Exploiting metastability and thermal noise to build a re-configurable hardware random number generator

Daihyun Lim[*a], Damith C. Ranasinghe[b], Srinivas Devadas[a], Behnam Jamali[b], Derek Abbott[b], Peter H. Cole[b]
[a] Massachusetts Institute of Technology, 77 Massachusetts Ave., Cambridge, MA, USA, 02139;
[b]School of Electrical & Electronic Engineering, Univ. of Adelaide, SA, Australia 5005

ABSTRACT

While pseudo random number generators based on computational complexity are widely used for most of cryptographic applications and probabilistic simulations, the generation of true random numbers based on physical randomness is required to guarantee the advanced security of cryptographic systems. In this paper we present a method to exploit manufacturing variations, metastablity, and thermal noise in integrated circuits to generate random numbers. This metastability based physical random number generator provides a compact and low-power solution which can be fabricated using standard IC manufacturing processes. Test-chips were fabricated in TSMC 0.18um process and experimental results show that the generated random bits pass standard randomness tests successfully. The operation of the proposed scheme is robust against environmental changes since it can be re-calibrated to new environmental conditions such as temperature and power supply voltage.

Keywords: PUFRNG, Metastability, Statistical testing, chaos theory, thermal noise

1. INTRODUCTION

Random number generators are important in a number of modelling and simulation applications. However, the most significant of which is their application in cryptography. The wide array of cryptographic applications employs secret keys that must be generated using a random process to ensure the security of the cryptographic system. Numerous cryptographic protocols also require random or pseudorandom inputs such as in the generation of digital signatures, or the generation of challenges in a challenge-response protocol [1]. Random numbers are also used in the selection of winning numbers for lotteries and the picking of premium bonds as in the United Kingdom. Cryptographic needs and large Monte Carlo computations continue to advance the development and research into truly random number generators.

Random number generators can be broadly categorised as follows:

1. Pseudorandom number generators
2. Physical random number generators.

The above types of generators are more specifically referred to as random bit generators when they are used to produce a stream of binary numbers. This stream can be subdivided into form blocks of random numbers of required block sizes such as 32 bits, 128 bits, or 1024 bits.

Pseudorandom number generators are based on a computational algorithm that receives a random sequence of length *l* as an input and outputs a binary sequence of length $x >> l$ which has the appearance of being random [1]. Such a generator employs the existence of a one-way function *f* that is based on the complexity of computations to make it irreversible. There are various algorithms for pseudorandom number generation (PRBG); ANSI X9.17 and FIPS 186 are examples of such generators [1,15]. The output of a PRBG is not random, but completely deterministic, while the number of possible output sequences is at most $2^l / 2^x$, of all possible bit sequences of length *x*.

[*] daihyn@MIT.EDU; phone (617) 225-9146

Noise in Devices and Circuits III, edited by Alexander A. Balandin, François Danneville,
M. Jamal Deen, Daniel M. Fleetwood, Proc. of SPIE Vol. 5844 (SPIE, Bellingham, WA, 2005)
0277-786X/05/$15 · doi: 10.1117/12.620306

While the output sequences of pseudo random number generators appear to be random, there is possibility for an adversary to predict random sequences by developing an equivalent algorithm or simply duplicating the generator hardware. To achieve unpredictable randomness, we can exploit the non-deterministic randomness in physical phenomena such as the decay of radioactive isotopes and laser scattering patterns through non-homogeneous materials to generate random numbers. Physical random number generators based on this physical randomness can be useful since there may not be an equivalent algorithm to simulate and predict the physical phenomena. However, exploiting this random source to produce a bit sequence that is both statistically independent and unbiased is not an effortless task. Random bit generators based on natural sources of randomness are exposed to environmental variations that can be sensitive to the generation of random sequences. Therefore, most physical random number generators should employ post processing units to compensate for the environmental variation and statistical defects of output sequences as shown in Figure 1.

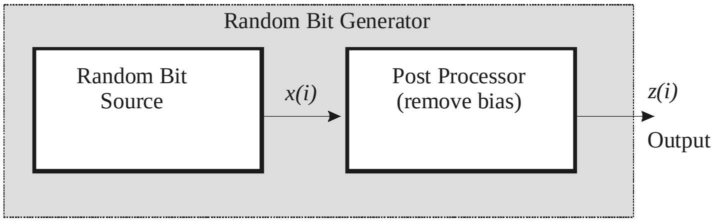

Figure 1: A random bit generator with a post processor

1.1. Sources of Randomness

Random sources can be constructed from dedicated hardware devices (Hardware based generators); such as oscillators with considerable jitter. A number of hardware random number generators can be found in [16], [17] and [18]. Random sources may also be extracted from software procedures using the platform on which the generator is implemented (Software based generators). For instance, event timing may provide sources for SWG, such as mouse clicks, key strokes, size of input and output buffers, or network access.

Software based generators (SWG) are not based on very ideal sources, and it is difficult to properly evaluate and assess the robustness of the sources in regards to observations and possible manipulations. Thus software based generators use a combination of sources to obtain protection from one of its sources being manipulated. Raw bits generated from SWGs most often need to be heavily processed before a random sequence is obtained.

Hardware based generators (HWG) have a number of advantages over software generators. Generally HWG can be implemented using common integrated technologies, they can be fabricated into tamper resistant devices to prevent an adversary from performing observations or manipulating the generator. Hardware based generators are also faster, and are capable of producing generators with high throughput. Commonly used sources for hardware generators are physical phenomena such as thermal noise, shot noise, avalanche noise, phase noise, cosmic radiation and atmospheric noise.

In this article we present an assessment of the possibility of using metastability and thermal noise as a source for a hardware random number generator. The HWG presented in this paper is fully integrable using standard CMOS technology.

1.2. Metastability

When a signal violates a device's signal setup and hold timing requirements of a latch, the output from the device becomes unstable [19]. In this case the observed output from a RS latch can be either high or low or it can even oscillate (see Figure 2). This widely undesirable phenomenon is known as metastability [22, 23] and circuit designers try to avoid this metastability as the final state of the device is unpredictable. A latch is such a bistable device that can enter into a metastable state.

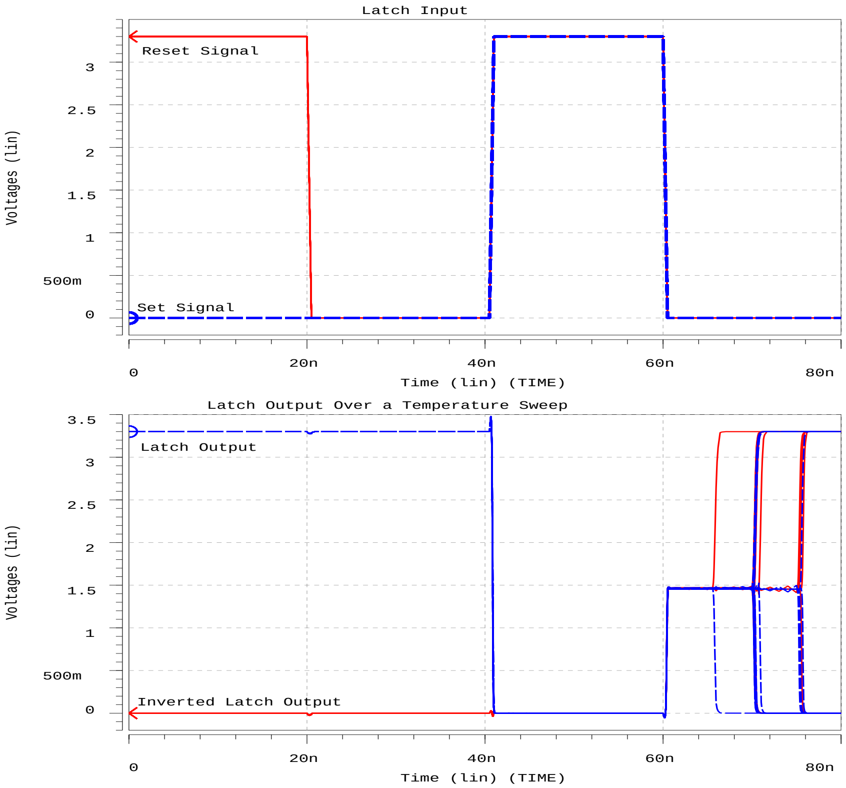

Figure 2: RS Latch under a metastable condition: The oscillations of the latch output can be seen here on the simulation. The final stable output value of the latch varies with the operating temperature. This indicates the role played by thermal noise in determining the final output value of the latch.

Figure 2 shows a HSPICE simulation of a metastable condition in a RS latch as the temperature is swept from -25 to 125^{o}C. The simulation results show that during the time period from 60 ns to 80 ns the latch is in a metastable condition where the output of the latch enters a state where it is neither a logic one nor a zero. Since the final output value can not be predicted due to thermal noise this metastability provides a source of randomness that can be used to construct a simple and efficient physical random number generator. There have been attempts to use this metastability as a source of randomness [25, 26, 27]. However, propagation delay variation by environmental changes such as temperature and power supply voltages makes the device difficult to stay in the metastable condition. In this paper, we propose an alternative method to keep the generator circuit in a metastable condition to produce random sequences in practical range of environmental changes.

2. RANDOM NUMBER GENERATOR DESIGN

A secret key extraction technique from the manufacturing variation in ICs [20,28] provides a suitable solution to create and keep the metastability on a recurring basis. The technique employs a PUF (Physically Unclonable Function) circuit which has an exponential number of delay path configurations determined by a challenge input. The observation of PUF results reveals that for certain challenges, the setup and hold time violation of an arbiter (D-latch) leads to unpredictable responses as the arbiter enters into a metastable condition. It is possible to identify and exploit the metastable challenges to obtain the metastability of the arbiter to produce random responses from the PUF. The transformed PUF will be referred to as PUF Random Number Generators (PUFRNGs) throughout this paper.

2.1. Circuit implementation

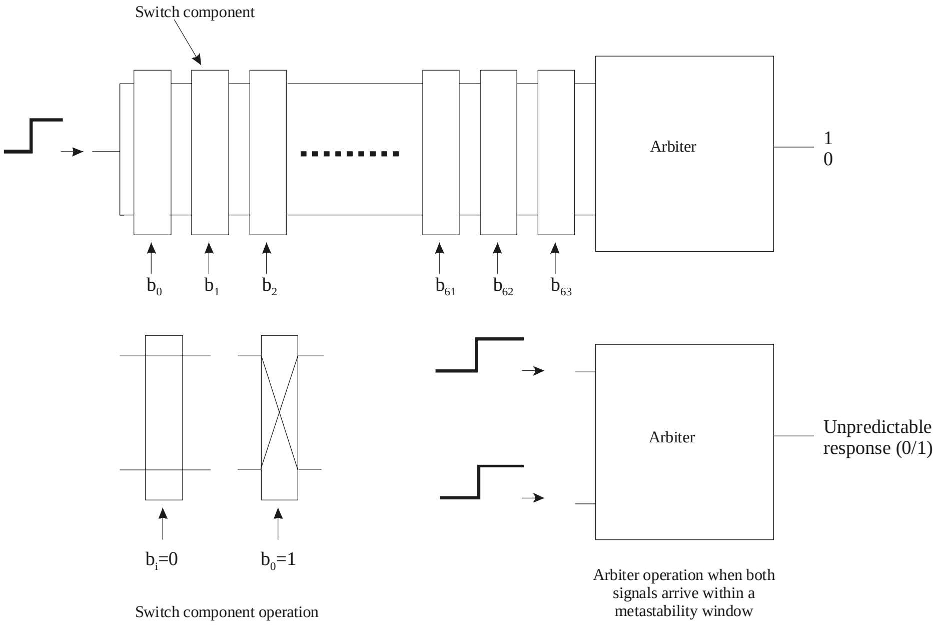
Switch component
Arbiter
1
0
b_0
b_1
b_2
b_{61}
b_{62}
b_{63}
Arbiter
Unpredictable response (0/1)
b_i=0
b_0=1
Switch component operation
Arbiter operation when both signals arrive within a metastability window

Figure 3: PUFRNG based on an arbiter-based PUF circuit.

The block diagram in Figure 3 depicts the structure of a PUFRNG circuit which is based on the arbiter-based PUF in [28]. The circuit accepts a *n* bit challenge b_0, b_2, b_3, …, b_n to form two delay paths in 2^n different configurations. In order to generate a response bit, two delay paths are excited simultaneously to allow the transitions to race against each other. The arbiter block at the end of the delay paths determines which rising edge arrives first and sets its output to 0 or 1. The actual implementation of arbiter-based PUFs in [28] uses 64 bit challenges. For some challenges, the delays in the two paths are approximately identical. When two transitions violates the setup time of the arbiter, the arbiter becomes metastable and generates random responses. The details of the switch component are given in Figures 5.

The switch component indicated in Figure 3 is implemented using a pair of two-to-one multiplexers (refer to Figure 4). Depending on the select bit C_i, the switch either allows the signal to travel straight through or swap the delay paths. The arbiter is constructed using a simple transparent latch with an active-low enable input. The arbiter favours the path to output zero since it is preset to zero and requires a setup time constraint to switch to a logic one. Fixing a small number of most significant challenge bits can compensate for this skew by effectively lengthen one delay path. The layout was carefully done to ensure that both paths are symmetrical and arbiter responses are not biased to 0 or 1.

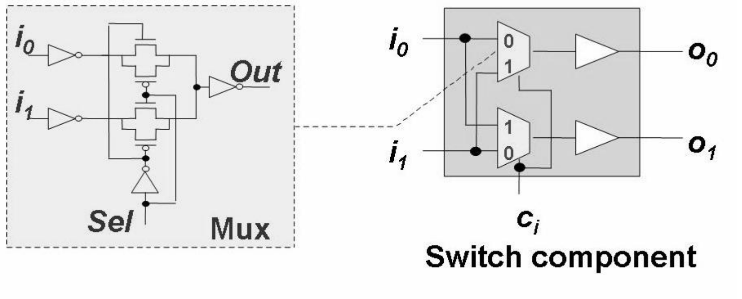

Figure 4: Switch component implemented using two-to-one multiplexers to swap two delay paths.

The chip used in testing was built in TSMC's 0.18 μm, single poly, 6-level metal process with standard cells. The chip contains eight sets of the arbiter-based PUF circuits capable of generating an 8 bit response for a given challenge and a JTAG-like serial interface for communication. The total area of the eight PUF circuits is 1212 μm x 1212 μm and the chip can be operated 100 MHz.

2.2. Design Analysis

The PUFRNG exploits the metastability of D-latch outputs cased by approximate identical timing of data and gate inputs. The output of the latch is largely determined by thermal noise. Figure 5 shows the probability density of the random variable *k*, which is the number of 1s in 200 repeated measurements for a given random challenge. In the middle of the density function, there exist the challenges whose responses consist of approximately 50% logic ones and 50% logic zeros. Within a temperature tolerance of ±5^{o}C from the operating temperature, the responses from these challenges can be used to generate a random bit stream.

The responses from the PUFRNG are sensitive to environmental conditions such as temperature and power supply voltage. In addition fabrication process variations will also influence the responses obtained from one PUFRNG to another for the same challenge. A challenge that generates an unreliable responses may not generate an unreliable responses if environmental conditions change beyond the tolerance level of 5^{o}C from the original temperature. Hence, each time a PUF is used as a source of randomness, a number of random challenges must be tested to select a challenge that produces unstable responses.

From experimental results, approximately 10 challenges out of 10,000 challenges (0.1%) produce unstable responses in a given environmental condition. Based on the performance of PUF circuits, it takes 0.5 seconds to test the randomness of 10,000 challenges by 1000 repeated measurements [20]. Hence it is possible to complete the initialization of a PUF random number generator within a second.

The input and output functions of the generator are responsible for most of the power consumption in the PUFRNG and the power consumption of the generator core is relatively small. The total power consumption of a PUFRNG circuit is about 130 μW in out implementation.

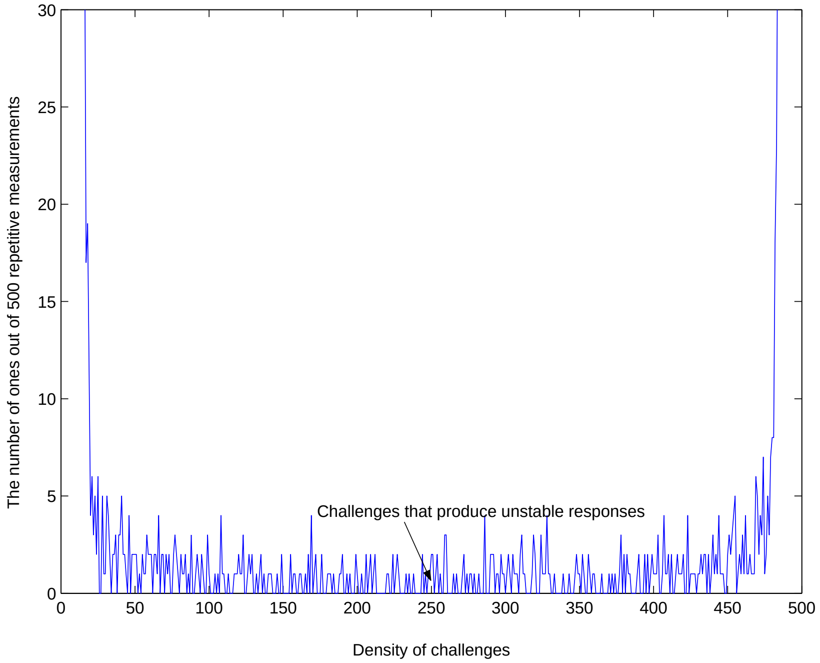

Figure 5: The density function of the random variable *k*, where *k* is the number of 1's out of 200 repetitive measurements [20].

2.3. Increasing the dynamic range of operation

The generator is sensitive to the power supply voltage and the temperature of the surrounding environment [20]. However problems caused by operational voltage changes can be minimised by the fabrication of a voltage regulator on the PUFRNG. In the testing stage of the PUFRNG a simple mechanism was used to allow the generator to function correctly over a temperature range of 80ºC by using eight different PUFRNG circuits each calibrated at intervals of approximately 10ºC, in view of the fact that each PUFRNG provided a temperature tolerance of ±5ºC. The experiments were run in an oven with thermostat control to provide cyclic temperature changes for the experimental collection of the random bit stream. The output bit stream was a result of an exclusive OR operation on each of the individual bit streams (as indicated in Figure 6).

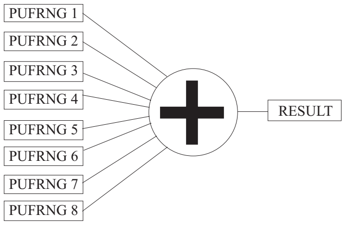

Figure 6: Using eight PUFRNGs to compensate for variation in operating temperature

The following section will examine the nature of the system used for generating random numbers and the randomness of generated bit sequence to evaluate the quality of randomness from the PUFRNG.

3. THE EVALUATION OF THE GENERATOR

In a true random number generator the probability of producing either a 1 or a 0 should be ½ where each bit is generated independently of any other bit in the bit stream. Hence it should not be possible to predict the value of a given bit with a probability greater than ½. These conditions form the framework of an ideal random number generator.

3.1. Chaos Theory (Dynamic system analysis)

It is possible to use chaos theory, a division of nonlinear system analysis, to analyse the complex system used for random number generation to investigate if the system exhibits behaviour that is neither random nor periodic. The chaos theory allows the characterization of the PUFRNG as a random, probabilistic (or chaotic) and deterministic systems. Furthermore the analysis can be used to discover underlying behaviour patterns, system information and dynamical system models that may render the generator deterministic without uncovering the laws and equations governing the dynamics of a given system.

A technique for the analysis of chaotic data is based on Taken’s Embedding Theorem [32], which allows the reconstruction of the phase space of the system dynamics. The reconstruction of the phase space can be performed from a finite time series of observed random numbers (observation of a single variable). This method relies on the appropriate selection of the delay time and the embedding dimension.

A phase space is a collection of possible states of a dynamical system. The elements of a phase space represent possible states of the system [34]. Implicit in the notion of phase space is that a particular state in phase space specifies the system completely; it is all the information needed to have complete knowledge of the immediate future of the system. Thus the phase space of the planar pendulum is two-dimensional, consisting of the position (angle) and velocity. According to Newton, specification of these two variables uniquely determines the subsequent motion of the pendulum.

Chaotic systems are deterministic and the exact system state can be expressed as

$$\mathrm{X}(t) = \left((\mathrm{x}(t), \mathrm{x}(t-\tau), \mathrm{x}(t-2\tau), \ldots, \ \mathrm{x}(t-(k-1)\tau) \right) . \qquad (1)$$

where t is a scalar index for the data series and τ is the interval of observations [35]. Let **F:** $\Re^k \rightarrow \Re^k$ be the nonlinear function governing the system. Then the future state of the system at time $t+\tau$ can be determined, such that

$$\mathbf{x}(t+\tau) = \mathbf{F}(\mathbf{X}(t)) + p(t) \tag{2}$$

There is relatively small, zero mean, probabilistic component *p(t)* added since real world systems are not completely deterministic. This term accounts for the random effects.

3.1.1. Attractors

An attractor is simply a state into which a system settles (thus dissipation is needed). Hence over time a dissipative dynamical system may settle into an attractor. The attractor may be a point, a closed path or a complex object on a phase space plot. The attractor is a geometric representation of Equation 1 for some large value of *t* where the effects of the transient have dissipated.

However a formal definition of an attractor is a set in the phase space that has a neighbourhood in which every point stays nearby and approaches the attractor as time tends to infinity. For the general chaotic system given by Equations 2, the *k*-dimensional system will have a non-intersecting attractor with a bounded path of infinite length. However this attractor will be encapsulated in a finite *k*-dimensional volume.

3.1.2. Phase space reconstruction

Employing Taken's theorem to reconstruct the phase space requires the determination of the delay τ and the embedding dimension *d*. Choice of τ should provide low correlations between adjacent elements in the embedded vector so that the original data series is not reiterated. The popular average mutual information algorithms can be used to evaluate the lag τ. Hence the first minimum of $I(\tau)$ (average mutual information function) which measures the average amount of information (bits) shared by two measurements is used as the lag τ.

Furthermore, the correct dimension *d* unfolds the attractor from the time series. The false nearest neighbour [35] algorithm can be used to evaluate the embedding dimension *d*. However [31] proposed a method for determining a good embedding dimension using the ideas behind false nearest neighbour algorithm. In [31] Cao proposed two metrics, $E_1(d)$ and $E_2(d)$. Here, $E_1(d)$ is used to discover a good embedding dimension, while $E_2(d)$ is used to determine whether the original data series is random. A suitable embedding dimension is given by the value of *d* where $E_1(d)$ stops changing. While random signals will exhibit a $E_2(d)$ that is close to unity for all values of *d*, while chaotic signal will have $E_2(d)$ values that is less than unity for small values of dimension *d*.

3.2. Statistical testing

Randomness is a property that can be characterized and described in terms of probability. The outcome of statistical tests applied to a random number sequence can be thus described in probabilistic terms. There is an array of statistical tests available to test the randomness of random and pseudorandom number generators. Even though these statistical tests do not provide definite results, it is possible to interpret these results with care and caution to determine the randomness of a generator. The general rule of thumb is "more tests the better". The generator bit stream was subjected to a battery of statistical tests for randomness used by The National Institute of Standard and Technology (NIST; an agency of the U.S. Commerce Department's Technology Administration [29]). A more detailed discussion of the tests and their interpretations can be found in [1]. It is however important to note that the test suite is suitable for identifying "deviations of binary sequences" from randomness. However factors contributing to these deviations are numerous and it is possible to expect a certain number of failures from a particular generator.

3.2.1. Hypothesis testing

Each NIST statistical test assesses a binary sequence to establish whether there is significant evidence to suggest that the null hypothesis (H_0) should be rejected in favour of the alternative hypothesis. Here the null hypothesis H_0 is that the sequence being tested is random, while the alternative hypothesis H_1, is that the sequence being tested is not random. Thus for each applied test a decision is made to accept or reject the null hypothesis based on statistical evidence.

A statistical hypothesis, commonly denoted as H_0 is an assertion about a distribution of one or more random variables [21]. This assertion can then be tested using a method based on the observations made on the random variables. The test can provide evidence to support or reject the hypothesis H_0.

Each test statistic obtained for each individual test is used to calculate a *P*-value that indicates the strength of the evidence against the null hypothesis. Thus for each test, the *P*-value is the probability that a perfect random number generator would have produced a sequence that is less random than the tested sequence, given the particular non-randomness being gauged by that particular test. Hence to reject the null hypothesis Ho (that is fail a test) at a 95% confidence level would require a *P*-value < 0.05. The possible outcomes of the conclusions from a statistical test are outlined in Table 1. It is clear from Table 1 that *P*-value only asses the relative incidence of Type I errors. Hence, it is important to note here that the test only provides a measure of the strength of the evidence provided by the data against the hypothesis and that the deduction derived from the test is not irrefutable but rather probabilistic.

Correct Result	**Test Result Decision**	
	Accept H_0	**Reject H_0**
H_0 True	Correct Decision	Type I error
H_1 True	Type II error	Correct

Table 1: Matrix of possible conclusion derived from a statistical test's hypothesis testing.

3.2.2. Statistical Test Suite

The statistical test suite employed (as detailed in [2]) is briefly described in Table 2.

No	Test	Description
1	The Frequency Test	The test aims to determine whether the proportion of 1's and 0's in a given sequence is that expected from for a random sequence. [1, 2]
2	Frequency Block Test	Similar to the above test but the focus is now the M-bit blocks within a given sequence. The block size used was 128 bits [2]
3	The Runs Test	The purpose of the test is to determine whether the number of runs (either 0's or 1's) of various lengths in the given sequence is as expected from a random sequence [1,2,3, 4]
4	Test for the Longest Run of Once in a Block	The test examines if the length of the longest run of ones within the given sequence is consistent with the length of the longest run of ones that would be expected in a random sequence [1, 2, 3, 4, 5].
5	The Binary Matrix Rank Test	The purpose of the test is to discover linear dependence among fixed length substrings of the original sequence [2, 6, 7].
6	The Discrete Fourier Transform Test	This test is used to examine the peak heights in the Discrete Fourier Transform of a given sequence. The test is able to depict periodic features in the tested sequence by examining the number of peaks exceeding the 95% peak height threshold value. The number of peaks exceeding this threshold peak height is less than 5% for a random sequence [2, 8].
7	The Non-overlapping Template Matching Test	This test is designed to search for the number of occurrences of pre-specified bit patterns. The test is aimed at detecting generators that produce large occurrences of a certain aperiodic bit pattern. The size of the template used was 9 bits in length, which resulted in a total of 148 templates being applied to each of the sequences. The results from this test are similar to having applied 148 different tests on the sequence of numbers provided [2].
8	The Overlapping Template Matching Test	Similar to the above test, with the exception that once a pattern is found the search window is now advanced only one bit instead of advancing the window to the end of the pattern as performed in the non-overlapping template matching test [2, 9].
9	The Serial Test	The purpose of this test is to establish whether 2^m m-bit overlapping patterns occurs as many times as expected from a random sequence. In a random sequence every m bit pattern has the same probability of appearing as every other m-bit pattern [1, 2, 10].
10	The Approximate Entropy Test	The purpose of the entropy test is to compare the frequency of overlapping bit patterns of two consecutive lengths of *m* and *m+1* bits with that expected from a random sequence [2, 11, 12, 13].

11	The Cumulative Sums Test	The Cumulative Sums (Cusum) test determines whether the cumulative sum of partial sequences occurring in a given bit string is that expected from a random sequence. The cumulative sum may be considered as a random walk and thus for a random sequence the deviation from the random walk should be near zero. This test is performed once going forward in the sequence and then going in the reverse direction [2, 5].

Table 2: Description of the tests used from the NIST test suite

4. ANALYSIS AND INTERPRETATION OF THE TEST RESULTS

Fabricated PUF generators were mounted on a circuit board and interfaced to a PC using a JTAG interface. Each PUF generator was initialised using 10,000 random challenges to select challenges that produced an unstable responses at different temperatures (10, 20, 30, 40, 50, 55, 60, 70 ˚C). The challenge was repeatedly used to obtain 4.5 million random bits after post processing of the output sequence.

4.1. Post Processing

In order to make the stream of bits emanating from the PUFRNG uniformly distributed, it was necessary to pass the bits through an entropy distillation process [1]. Post processing is required to remove bias from an original bit stream at the expense of reducing the overall size of the original bit stream. The method adopted is detailed in [1] and is that of von Neumann [33], and it involves the parsing of bits generated from the random number generator in pairs, and then transforming them according to the scheme outlined in Table 3. This method resulted in a typical reduction in the original PUFRNG bit stream in the range of 65-75%. However this is only a general estimate as it is a variable parameter that tends to depend on the environmental conditions such at the device operating temperature.

Input bits	**Output bit (transformation)**
10	1
01	0
11	Ignore (indicates that nothing is appended to the post processed bit stream)
00	Ignore (indicates that nothing is appended to the post processed bit stream)

Table 3: Post processing transformations. The original bit stream from the PUFRNG is obtained as non-overlapping pairs (input bits). The corresponding new output is then depicted in the 'output bit' column, where 'Ignore' System Analysis

4.2. System Analysis

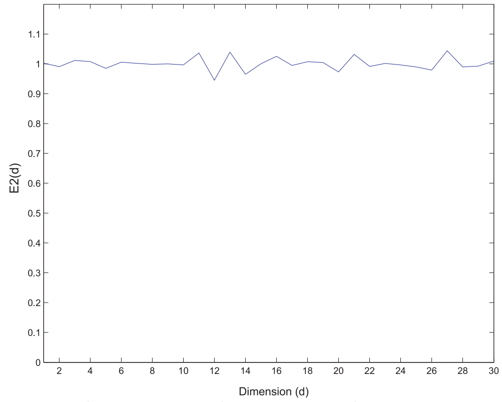

Figure 7 : Plot of *E2(d)* metric for the series of 32 bit random numbers

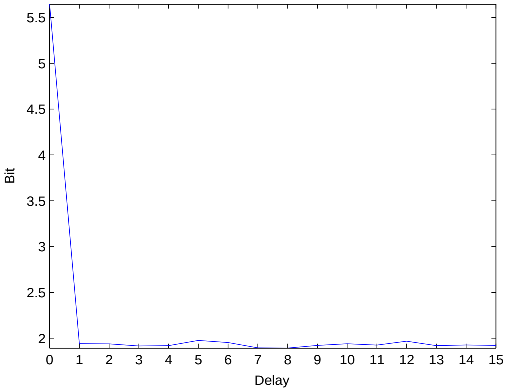

Figure 8: Mutual information function for the random data

The Discrete Fourier Transform spectrum in Figure 11 shows a broadband spectrum. Both chaotic and random systems will exhibit such a spectrum. However Section 3 described the features of a chaotic system and the reconstruction of a phase space. This section shows the results from the analysis. All the calculations depicted here were performed using the TSTool add-on program for Matlab. The bit stream subdivided into 32 bit blocks provided the random numbers for the following analysis.

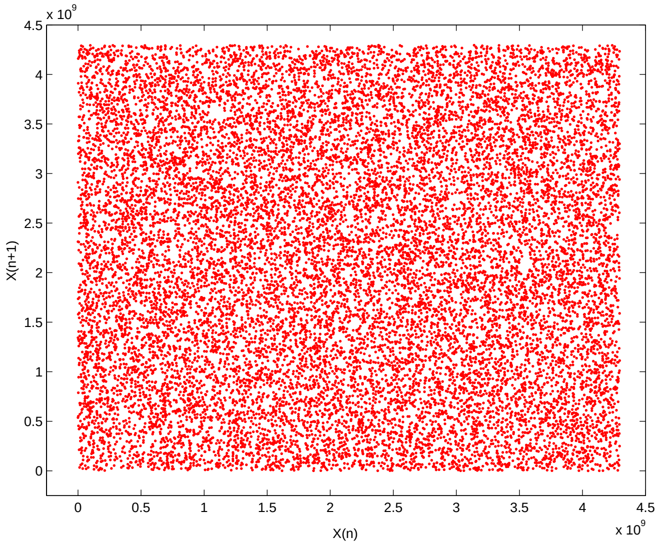

Figure 9: 2-D phase space plot of the random numbers using a delay of one estimated from the average mutual information algorithm. The plot does not show an attractor, however strange attractor of a square shape is only an artefact of the use of 32 bit blocks to generate random numbers.

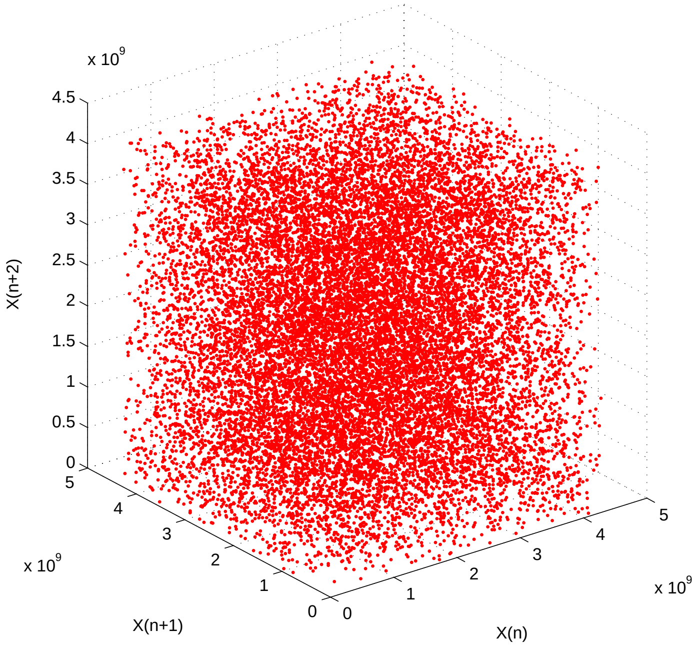

Figure 10: 3-D phase space plot of the random numbers using a delay variation of one estimated from the average mutual information algorithm. Again the plot does not depict an attractor; however the space of the plot is limited by the use of 32 bits of the random bit stream to generate data.

Cao's [31] $E_2(d)$ metric applied to identify whether the number sequence is random is shown in Figure 7. Here, $E_2(d)$ remains at unity for all values of d, suggesting a random system and hence the system is not chaotic. Nevertheless, this can be shown graphically for a two and a three dimensional phase space reconstruction by using the calculated lag.

Figure 8 shows the average mutual information plot, $I(\tau)$ to support the computation of lag time. The first minimum of the display is at lag one. This will be used as the delay coordinate in the phase space reconstruction. Figure 9 and Figure 10 depicts the two dimensional and three dimensional phase space reconstruction.

4.3. Statistical Testing

4.3.1. Parameters used in the test suite

Table 4 provides the test suite specific parameters for all the parameterised tests used during the analysis.

No	Test	Parameter value
2	Frequency Block Test	The block size used was 128 bits.
7	The Non-overlapping Template Matching Test	Template length used was 9 bits.
8	The Overlapping Template Matching Test	Template length used was 9 bits.
9	The Serial Test	Block length used was 16 bits.
10	The Approximate Entropy Test	Block length used was 10 bits.

Table 4: Test parameters used in the NIST test suite

4.3.2. Evaluation of test results

The guidelines in [2] can be used to interpret the test results. Details of these guidelines can be obtained from [2] and [1]. Table 5 gives a summary of the number of sequences that passed each of the tests performed using a *P*-value of 0.01 (that is $\alpha = 0.01$) as the significance level to reject or accept the null hypothesis. A pass in a test indicates that there is no significant evidence to reject the null hypothesis and thus the sequence can be considered to be random.

Number	Test	Description
1	The Frequency Tests	PASS
2	Frequency Block Test	PASS
3	The Runs Test	PASS
4	Test for the Longest-Run-of-Once in a Block	PASS
5	The Binary Matrix Rank Test	PASS
6	The Discrete Fourier Transform Test	PASS
7	The Non-overlapping Template Matching	PASS (except four templates failed)
8	The Overlapping Template Matching Test	PASS
9	The Serial Test	PASS
10	The Approximate Entropy Test	PASS
11	The Cumulative Sums Test	PASS

Table 5: Test result summary

Figure 11 shows the Discrete Fourier Transform of the random bit sequence. The sequence has a broadband spectrum and there is no significant frequency component in the spectrum. The spectrum shows that less than 5% of the peaks are below the 95% confidence level. The number of peaks exceeding this confidence level is less than 5% for a random sequence [2, 8].

The test results from the individual tests do not indicate a deviation from randomness. However, two recommended approaches by NIST to interpret the empirical results can be used to investigate the validity of the null hypothesis. The method adopted by NIST includes

1. the examination of the proportion of sequences that pass a given statistic test and
2. the estimation of the distribution of *P*-values to ensure that they are uniformly distributed.

If either of the above evaluations fails, the null hypothesis can be rejected. However, further testing on the generator should be performed to ensure that the conclusion was not due to a statistical irregularity.

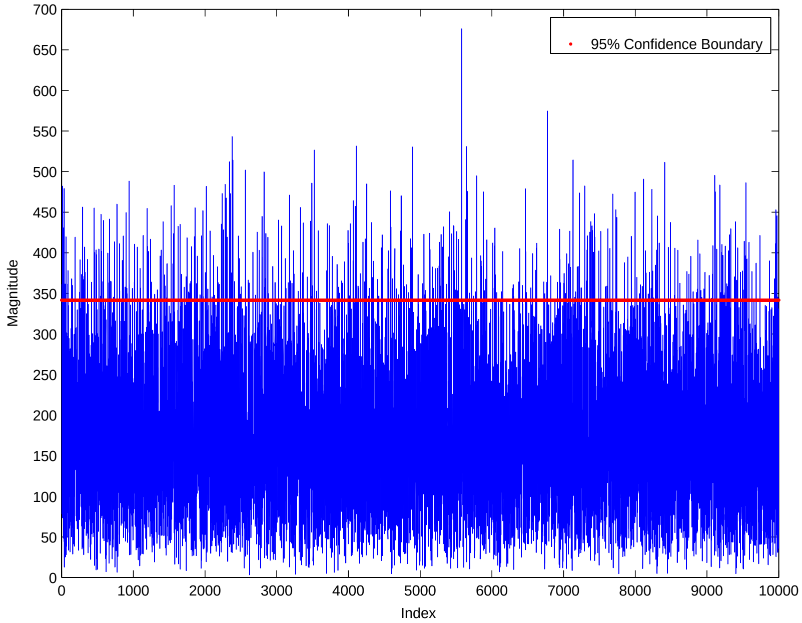

Figure 11: Discrete Fourier Transform test results of the random bit sequence. Measurement results show that only 4.5% of the spectral lines are above the 95% confidence boundary.

4.3.2.1. Proportion of sequences passing a Test

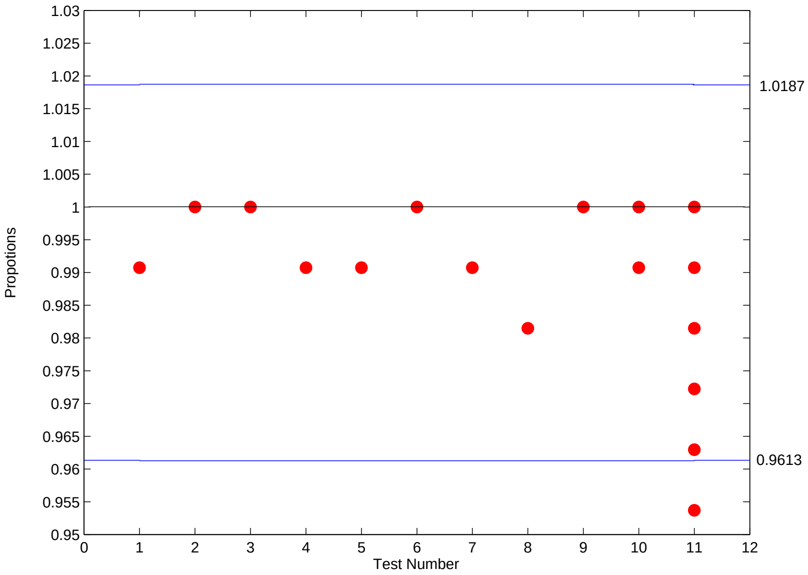

Figure 12: Proportion of sequences passing each test based on their *P*-value.

The empirical results can be used to calculate the proportion of sequences that passes a given test. The range of acceptable proportions is given in [2] and evaluated using the confidence interval defined as

$$\hat{p} \pm 3\sqrt{\frac{\hat{p}(1-\hat{p})}{m}}, \tag{3}$$

where $\hat{p} = 1 - \alpha$ and m is the number of sequences tested.

Figure 12 is a graph of the proportion of sequences that passed each test, with the confidence intervals marked with dotted lines. It can be seen that there is an outlier below the lower limit of the confidence interval due to four templates from the template matching test failing to pass the expected proportion of passing sequences.

4.3.2.2. Uniform distribution of *P*-values

Evaluation of the uniformity of the distribution of *P*-values is discussed in detail in [28]. A *P*-value calculated on the distribution of *P*-values of a statistical test can be used to accept the distribution of *P*-values as being uniform or non-uniform.

A χ^2 test and a determination of a *P*-value that corresponds to the goodness-of-fit distributional test of the P-values can be used to evaluate the so called '*P*-value of *P*-values'. Thus using ten bins to analyse the distribution of P-values the χ^2 statistic is given by [28]:

$$\chi^2 = \sum_{i=1}^{10} \frac{(F_i - s/10)^2}{s/10} \tag{4}$$

where F_i is the number of *P*-values in the bin *i* of Figure 16 and *s* is the sample size. The *P*-value of *P*-values is then given by the complemented incomplete gamma function:

$$1 - \Gamma(a, z) \tag{5}$$

where a = 9/2 and z = $\chi^2/2$ [28]. The resulting calculation yields a mean χ^2 value for the distributions in Figure 16.

A significance level of α = 0.0001 was used to assess the uniformity. Thus a *P*-value calculated on the distribution of *P*-values ≥ 0.0001 was considered to have a uniform distribution [28]. Table 6 summarises the assessment performed on the *P*-values obtained for each statistical test.

Number	**Test**	**Uniformity of *P*-value distribution**
1	The Frequency Tests	PASS
2	Frequency Block Test	PASS
3	The Runs Test	PASS
4	Test for the Longest-Run-of-Once in a Block	PASS
5	The Binary Matrix Rank Test	PASS
6	The Discrete Fourier Transform Test	PASS
7	The Non-overlapping Template Matching Test	PASS
8	The Overlapping Template Matching Test	PASS
9	The Serial Test	PASS
10	The Approximate Entropy Test	PASS
11	The Cumulative Sums Test	PASS

Table 6: Results evaluating the uniform distribution of *P*-values

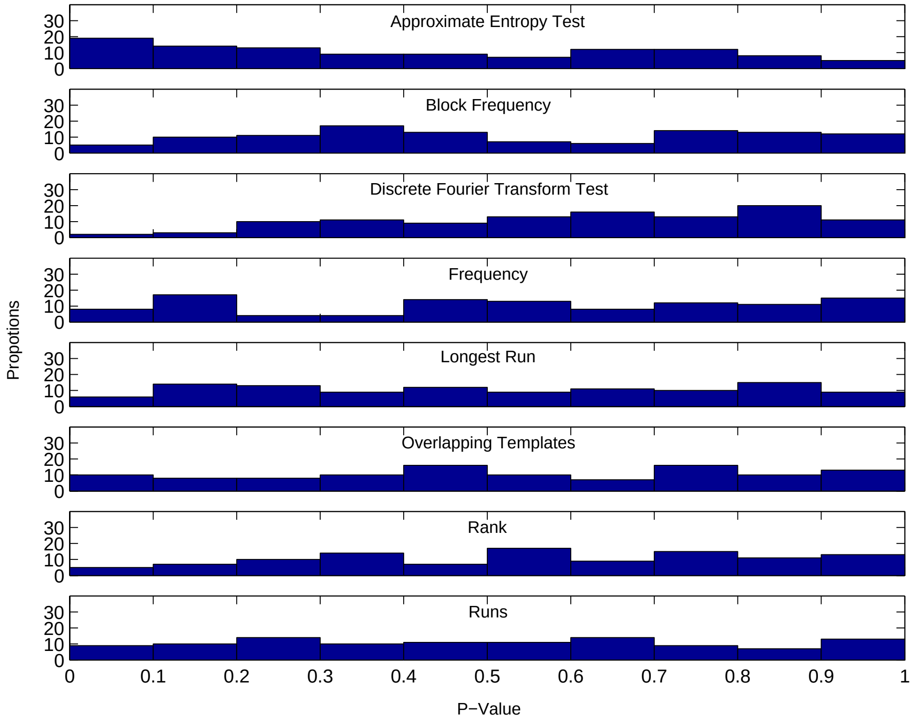

Figure 13: Histogram of the *P*-value distributions resulting from applying the eleven statistical tests from Table 2 to 108 bit stream of length 38912 bits. The graphed data show that all the distributions are approximately uniform

5. CONCLUSION

The system analysis showed that the random number generator based on an arbiter based PUF circuit is not deterministic but rather random. The analysis of the generated bit stream showed that the generator successfully passed the eleven tests used from the NIST test suite. This proves the quality of the randomness of the bit stream from a PUFRNG under varying environmental conditions.

The PUFRNG provides a cost effective solution to produce millions of random bits in a very short time by fabricating a number of PUFRNGs on a single IC. This generator can be easily constructed using standard digital gates and layout tools. However, the PUFRNG requires some overhead such as post processing and calibration prior to its use.

Nevertheless the ability to calibrate the PUFRNG very rapidly can be an advantage. This unique ability of the PUF random number generator allows the generator to adapt to external influences and to fine-tune the generator for greater performance. Compared to other physical random number generators, the PUF random number generator can be a compact and a low-power solution. A 64-stage PUF circuit costs only less than 1000 gates and the circuit can be implemented using standard IC manufacturing processes. Additionally, various kinds of low power techniques such as sub-threshold logic design and multi-thresholds CMOS design can be utilized to reduce the power consumption to make it suitable for use in devices sensitive to low power consumption.

The effects of environmental conditions on the measurements obtained from a PUF are documented in [20], and the symmetrical nature of the circuit counter acts to reduce much of the variation provided otherwise. Changing temperature affects the propagation delays and the metastability window and thus the same unstable response producing challenge may not produce acceptable results if the operating temperature of the device changes drastically. The tolerance of performance and the calibration of PUFRNG to operate at a range of temperature provides to compensate for varying conditions of temperature. However effects of power supply voltage still need to be investigated to discover practical performance boundaries such that the PUFRNGs do not need to be calibrated prior to its use on every occasion. Nevertheless it is possible to fabricate a voltage regulator onboard the PUFRNG to prevent effects from higher voltage variations, but it will not be able to counteract conditions induced by voltages below a calibrated power supply voltage.

Future work will also involve the investigations into the effects of voltage on the performance of the PUFRNG. It may also be possible to avoid calibration by evaluating the performance of the circuits under different environmental conditions of varying voltage. It is also left to investigate whether the generator throughput can be improved. Although avoiding post processing would be the best means of improving the throughput of the generator.

6. ACKNOWLEDGEMENTS

This work used the NIST statistical test suite. The authors would like to thank NIST for allowing public access to the software and allowing modification of the software, the developers of TSTools for providing numerous routines used in system analysis and Matthew J. Berryman for improving the clarity of this work. The work presented here would not have been possible without the invaluable support from Zheng Zhu.

REFERENCES

1. A. Menezes, P. van Oorschot and S. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1996.
2. *Cryptography Theory and Practice,* CRC Press, 1995.
3. W.T. Holman, J.A. Connelly and A. B. Downlatabadi, *An Integrated Analog/Digital Random Noise Source,* IEEE Trans. Circuits and Systems I, vol. 44, no. 6, pp. 521-528, June 1997.
4. M. Dichtl and N. Janssen, *A High Quality Physical Random Number Generator*, Proc. Sophia Antipolis Forum Microelectronics, pp. 48-53, 2000.
5. C. S. Petrie and J.A. Connelly, *A Noise-Based IC Random Number Generator for Applications in Cryptography*, IEEE Trans. Circutis and Systems I, vol. 47, no. 5, pp.615-621, May 2000.
6. Phillips Semiconductors, *A metastability primer,* Application Note, AN219.
7. T.J. Chaney, *Measured Flip-flop Responses to Marginal Triggering,* IEEE Trans. On Comp., vol. 32, no. 12, pp. 1207-1209, December 1983.
8. G.R. Couranz and D.F. Wann, *Theoretical and Experimental Behaviour of Synchronizers Operating in the Metastable Region,* IEEE Trans. on Comp., vol 24, no. 6, pp. 604-616, June 1975.
9. M. J. Bellido, A. J. Acosta, et al., A Simple *Binary Random Number Generator: New Approaches for CMOS VLSI*,35th Midwets Symposium on circuits and Systems, August 1992.
10. M. J. Bellido, A. J. Acosta, M. Valencia, A. Barriga, and J.L. Huertas, *A Simple Binary Random Number Generator*, Electronic Letters, vol. 28, no. 7, pp. 617-618, March 1992.
11. S. Walker and S. Foo, *Evaluating Metastability in electronic Circuits for Random Number Generation,* IEEE Computer Society Workshop on VLSI, pp. 99-102, April 2001.
12. D. Lim, *Extracting Secret Keys from Integrated Circuits*, Master thesis, Massachusetts Institute of Technology, May 2004.
13. J.W. Lee, D. Lim, B. Gassend, G.E. Suh, M. van Dijk, S. Devadas, *A Technique to Build a Secret Key in Integrated Circuits for Identification and Authentication Applications*, 2004 Symposium on VLSI circuits, pp 176-179, 2004.
14. F. Spitzer, *Principles of Random Walk*. Princeton: Van Nostrand, 1964.
15. N. Boccara, *Modeling Complex System*s, Springer-Verlag, New York, 2004.
16. H. Abarbanel, *Analysis of Observed Chaotic Data*, Springer-Verlag, New York, 1996.
17. L. Cao, *Practical method for determining the minimum embedding dimension of a scalar time series*", Physical D, 110, pp. 43-50, 1997.
18. NIST home page, http://www.nist.gov.
19. A. Rukhin, J. Soto, J. Nechvatal, M. Smid, E. Barker, S. Leigh, M. Levenson, M. Vangel, D. Banks, A. Heckert, J. Dray and S. Vo, *A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications*, NIST Special Publication 800-22, 2001.
20. J. D. Gibbons, *Nonparametric Statistical Inference*, 2nd edition, New York: Marcel Dekker, 1985.
21. A. P. Godbole and S. G. Papastavridis, (ed), *Runs and patterns in probability*: Selected papers. Dordrecht: Kluwer Academic, 1994.
22. P. Revesz, Random *Walk in Random and Non-Random Environments*, Singapore, World Scientific, 1990.

23. I. N. Kovalenko, *Distribution of the linear rank of a random matrix*, Theory of Probability and its Applications., vol.17, pp. 342-346. 1972.
24. G. Marsaglia and L. H. Tsay, *Matrices and the structure of random number sequences*, Linear Algebra and its Applications, vol. 67, pp. 147-156, 1985.
25. R. N. Bracewell, *The Fourier Transform and Its Applications*., McGraw-Hill, 1986.
26. O. Chrysaphinou and S. Papastavridis, *A Limit Theorem on the Number of Overlapping Appearances of a Pattern in a Sequence of Independent Trials*, Probability Theory and Related Fields, vol. 79 pp. 129-143, 1988.
27. I. J. Good, *The serial test for sampling numbers and other tests for randomness*, Proc. Cambridge Philos. Soc., vol. 47, pp. 276-284, 1953.
28. S. Pincus and B. H. Singer, *Randomness and degrees of irregularity*, Proc. Natl. Acad. Sci. USA, vol. 93, pp. 2083-2088, March 1996.
29. S. Pincus and R. E. Kalman, *Not all (possibly) Random Sequences are created equal*, Proc. Natl. Acad. Sci. USA, vol. 94, , pp. 3513-3518, April 1997.
30. A. Rukhin , *Approximate entropy for testing randomness*, Journal of Applied Probability, vol. 37, 2000.
31. J. von Neumann, *Various techniques used in connection with random digits*, von Neumann’s collected works, vol. 5, 1963.
32. L. Cao, *Practical method for determining the minimum embedding dimension of a scalar time series*", Physical D, 110, pp. 43-50, 1997.