electronics

MDPI

Article

Stochastic Cell- and Bit-Discard Technique to Improve Randomness of a TRNG

Jae-Won Nam [1], Jaewoo Kim [2] and Jong-Phil Hong [2,*]

[1] Department of Electronic Engineering, Seoul National University of Science and Technology, Seoul 01811, Korea; jaewon.nam@seoultech.ac.kr
[2] School of Electrical Engineering, Chungbuk National University, Cheongju 28644, Korea; kjw@cbnu.ac.kr
* Correspondence: jphong@cbnu.ac.kr

Abstract: This paper presents a post-processing algorithm for a true random number generator (TRNG). Once the randomness of security key generation deteriorates for any reason, the entire chain of the security system can be compromised, increasing the odds of it being exploited by an attacker to retrieve information. Considering the change in the distribution of the RNG output sequence due to variations in the operating environment or the occurrence of aging phenomena in silicon-integrated circuits, a robust post-processing algorithm must be applied to an intrinsic TRNG to ensure the sustainability of a security system. Targeting high-level cryptography systems complying with the NIST 800-22a requirements, the proposed algorithm significantly improves the Hamming weight (HW) and successfully passes the NIST criteria while sacrificing approximately 20% of the entire number of available bits. The proposed algorithm improves the randomness of the TRNG through a sequential cell- and bit-level discarding technique, a cell-discard method, and focuses on improving the overall HW of the TRNG while the subsequent bit- discard method performs a Chi-square (χ^2) test. To prove the concept, we programmed the proposed algorithm in a FPGA and configured the output of the manufactured TRNG chip to be post-processed and stored into on-board memory in real time. For five different ring-oscillator-based TRNG prototypes (fully custom designed in the 65 nm CMOS process), the failed intrinsic TRNG output sequences were respectively post-processed, resulting in all surpassing the NIST 800-22a requirements.

Keywords: cryptography; random number generator; TRNG; post-processing; NIST 800-22a

Citation: Nam, J.-W.; Kim, J.; Hong, J.-P. Stochastic Cell- and Bit-Discard Technique to Improve Randomness of a TRNG. *Electronics* **2022**, *11*, 1735. https://doi.org/10.3390/electronics11111735

Academic Editor: Cheng-Chi Lee

Received: 20 April 2022
Accepted: 27 May 2022
Published: 30 May 2022

1. Introduction

In our modern culture, random numbers are easily encountered in everyday life, such as when tossing a coin, dice, and playing the lottery. They are used to protect users' privacy and information in the digital information age. In addition, random numbers are also important for Monte Carlo simulations and in statistical analysis [1,2]. Information security means protecting information from various threats, and the types of threats include damage, alterations, and leakage during data transmission, reception, searches, storage, and processing. From the supplier's point of view, information security involves the safe protection and operation of hardware databases, such as networks and systems, as well as the intelligence assets of computer facilities. On the other hand, from the user's point of view, information security is meant to prevent the leakage and abuse of personal information. Random numbers are important for authentication. In the authentication process, random numbers are used to compare queries or are used as keys for encrypting and decrypting queries or messages [3].

A random number is a randomly generated sequence, and the distribution of numbers should not be stochastic. Reliable random numbers are characterized by randomness, which is the property of a sequence to be random without statistical bias, with unpredictability in that the next number given a previous sequence cannot be predicted, and with non-reproducibility, which is a property according to which a sequence once generated cannot

be reproduced. A random number applied to a high-level security system must be tested in terms of the aforementioned three properties of a random number and must pass all NIST-defined tests [4]. A random number generator (RNG) is a system that generates unpredictable random numbers, and there are true random number generators (TRNG) and pseudo random number generators (PRNG). The TRNG has a characteristic of generating random numbers through a naturally existing entropy source, and the PRNG has periodicity because it generates random numbers according to a mathematical formula. Generally, TRNGs have been implemented in a small-area and low-power CMOS process targeting specific hardware, thus are not easily scalable to programmable hardware.

Entropy refers to the Shannon entropy used in statistical mechanics and represents the average level of information or uncertainty of a variable [5]. Entropy is a quantitative indicator of uncertainty. Given a discrete random variable $X = x_1, x_2, x_3, \ldots, x_N$, which occur with probability $P(x_1), \ldots, P(x_N)$, the entropy of X, $H(X)$, can be expressed as follows:

$$\text{Entropy} \equiv H(X) = -\sum_{i=1}^{N} P(x_i)\ log_2 P(x_i).$$

In contrast, a PRNG is based on software, and the output is determined by the SEED. The SEED of PRNG is comparable to the key to a lock, meaning that if the SEED is leaked to the public, all pseudo-random numbers will be reproduced by the attacker, defeating the security system. Hence, for a PRNG to generate safe random numbers, it needs a SEED generator that generates unpredictable random numbers. A typical example of a PRNG is a stream cipher, which is one of the structures of symmetric key encryption. The stream cipher has a structure that combines the random number generated according to the SEED and the data to be encrypted [6]. The PRNG implemented in software requires a CPU and MCU as well as memory at the same time; hence, it has a disadvantage in that it is not safe for security requirements, and it is affected by the constraint of requiring a SEED generator that generates unpredictable random numbers. However, as the TRNG is implemented in hardware, there is no need to use any computing resources, such as a CPU or a MCU, nor any separate memory. Additionally, it does not require an external function block, such as a SEED generator because it uses a naturally existing entropy source [7]. Typically, the TRNG used in the CMOS process utilizes meta-stability, oscillator jitter, and other noise sources, such as gate oxide and chaos.

The remainder of this paper is organized in the following manner: Section 2 introduces existing issues with the conventional TRNG. Section 3 describes a NIST 800-22a test suite. Section 4 presents the proposed post-processing algorithms. Sections 5 and 6 discuss TRNG circuit details and measurement results associated with the proposed post-processing algorithms, and Section 6 provides the conclusion.

2. Existing Issues with the Conventional TRNG

As an ideal RNG, TRNGs must utilize unpredictable and high-performance entropy sources. However, the entropy source comes from the operating conditions or external disturbances; thus, the hardware-based TRNG is sensitive to changes in the process, voltage, and temperature (PVT) and external attacks that affect the quality of entropy [8,9]. Regarding these properties, there have been many active studies of TRNG post-processing techniques to improve intrinsic TRNG performance outcomes [10–14]. Among PVT changes, the process variation is the only random parameter physically defined during the chip fabrication phase and not under the actual operating conditions. It is an unpredictable manufacturing error and causes various process deviations. One of the well-known mismatches is caused by an inconsistent etching and doping process at different wafer positions. Even with an identical material with an identical amount of processing time, the degree of etching can differ, resulting in errors in the width and depth. Wafer mismatch affects the operating speed of chips manufactured at each location due to the mismatch of the wafer thickness that occurs during the process of cutting and polishing the ingot. If there

is process variation, the physical characteristics of the transistor, such as width (W) and length (L), change. The drain current of the MOSFET transistor is expressed as follows:

$$I_D = \frac{1}{2}\mu C_{ox}\frac{W}{L}(V_{GS} - V_{th})^2.$$

Considering W and L along with other variables associated with transistor parameter variations caused by manufacturing errors, we can statistically predict a certain range of the drain current. In other words, even circuits with an identical structure may have different performance outcomes. Figure 1 shows two extreme examples of a cell that is biased due to process variations. Process deviation can be divided into FF, SS, FS, and SF cases according to V_{th} of PMOS and V_{th} of NMOS. In an inverter fabricated with a SF wafer, because V_{th} of NMOS is larger than V_{th} of PMOS, the duty cycle shrinks to less than 50% and is biased to 0. On the other hand, in the inverter is fabricated with a FS wafer, because V_{th} of PMOS is larger than V_{th} of NMOS, the duty cycle exceeds 50% and is biased to 1.

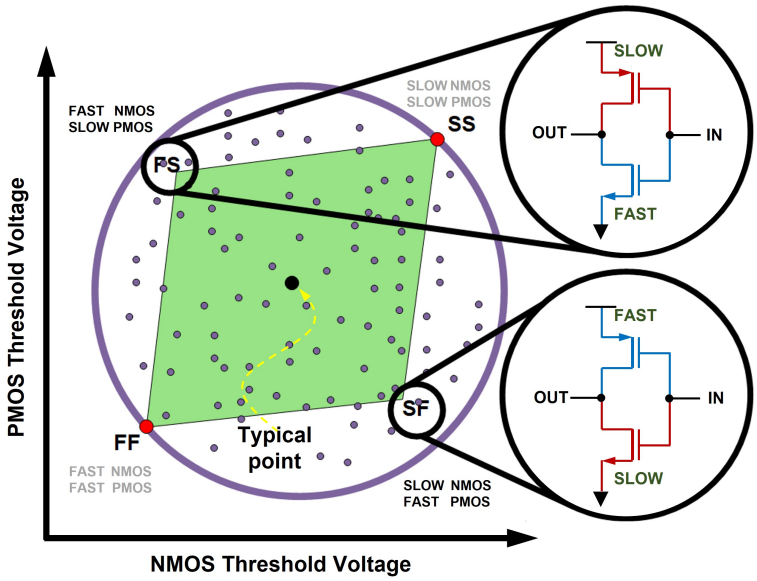

Figure 1. Example of process variations in an inverter.

In addition to this process, there is the aging of silicon as a factor that affects the performance of a TRNG. As silicon ages, the V_{th} of the transistor changes, which greatly affects the performance of the TRNG [10,15]. As the size of the process decreases, the length of the transistor channel also becomes shorter. In this case, electrons receive a high electric field due to HCIs (hot carrier injections), a short-channel effect, and deteriorate the quality of the insulating film of the gate [10]. Another phenomenon that arises during transistor operation is BTI (bias temperature instability). Given that the BTI characteristics change during the bias of the transistor, the V_{th} of the transistor continues to change, as shown in Figure 2.

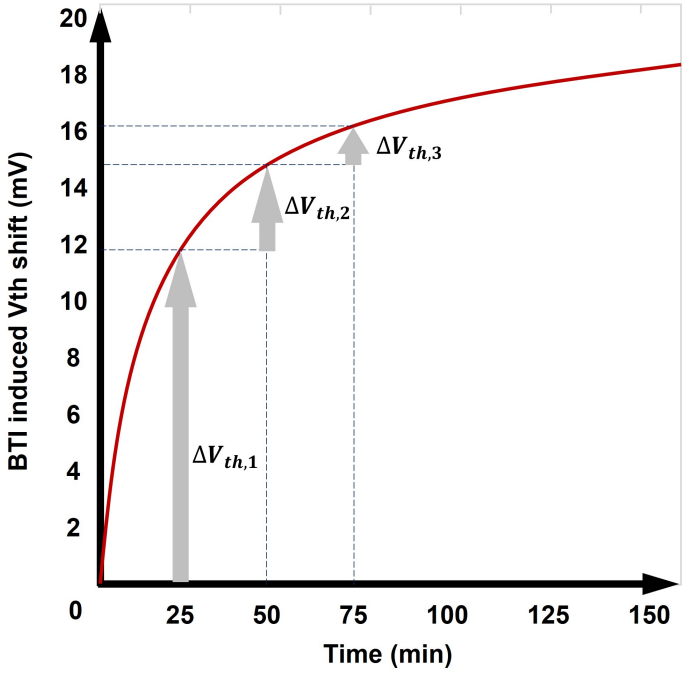

Figure 2. Bias temperature instability (BTI) effect.

3. NIST 800-22a Test Suite

In order to evaluate the performance of random numbers, tests by statistical methods have been widely used. Security Levels of Federal Information Processing Standard (FIPS) FIPS 140-2, a US federal information processing standard, addresses the security requirements that cryptographic modules must satisfy [16]. The Diehard test is a statistical test for measuring the quality of RNGs and has been developed by George Marsaglia for many years, thus far consisting of 12 sub-tests [17]. The document AIS.31 of the German Federal Information Technology Security Agency (BIS) (Bundesamt für Sicherheit in der Informationstechnik) describes the evaluation criteria for RNGs and consists of eight sub-tests [18]. Each of the analytical tests is contained in documents 800-22a of the National Institute of Standards and Technology (NIST). This document introduces a method to verify the randomness of a random number generator applicable to encryption.

NIST 800-22a introduces two approaches to the test results, the p-value and proportion. Table 1 shows an example of the NIST test results. According to the results, the p-value and the proportion values of each test are recorded, and pass or fail on the test is determined according to the authenticity of a specific condition. The p-value approach determines the uniformity of the p-value distribution. For each test, the minimum number of tests is determined from the value of α. If α is set to 0.01, the minimum number of tests is 100 in order to reflect a rejection ratio. Therefore, each test is performed with 100 sequences. Each of the 100 sequences results a corresponding p-value, and the test result is divided into the values of C1 to C10 according to the distribution. C1 is the number of cases in which each p-value in the test group exists between 0 and 0.1, C2 is for those existing between 0.1 and 0.2 and C10 denotes the number of cases in which each p-value in the test group is between 0.9 and 1. Therefore, the grand sum of the value from C1 to C10 becomes 100. The values from C1 to C10 are used for the Chi-square test, based on a statistical method (Chi-square distribution), and if the p-values are skewed to one side, as shown in Figure 3, the overall p-value becomes very low such that it can be rejected. In the proportion approach, the proportion value of passed sequences is calculated from the empirical results of statistical tests. The allowable range of the ratio can be defined as $p \pm 3\sqrt{p(1-p)/m}$, where $p = (1-\alpha)$. For example, if α is 0.01 and m is 100, the proportion value then becomes 0.96; likewise, if α is 0.001 and m is 1000, the range of the ratio is calculated as 0.98. The

range of the allowable ratio of 0.96 indicates that 96% of m tests have to pass the proportion approach. In this paper, we set α and m to 0.01 and 100, respectively.

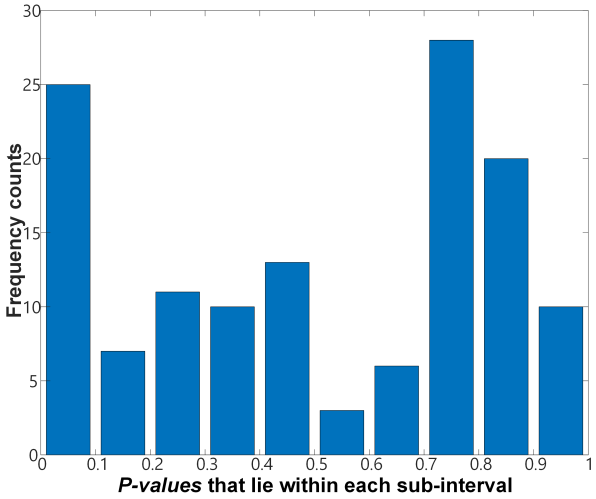

Figure 3. p-value distribution of the testing samples.

From a cryptographical perspective, SP 800-90A, B, and C documents written by US NIST clearly describe how a TRNG should be operated and evaluated [19–21]. According to the document SP 800-22a of US NIST, there are 15 verification tests concerning cryptographic stability perspectives [22]. Most of the verification tests in the test suite utilize a standard normal distribution and a Chi-square (χ^2) test as a reference distribution. Assuming that the RNG generates statistically random bits, the standard normal distribution is used to compare the test results from the RNG with the theoretically expected values. Moreover, the χ^2 distribution is used to check how well the observed frequencies of the measured samples match the expected frequencies of the hypothesized distribution.

Table 1. NIST 800-22a test examples.

C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	p-Value	Proportion			Statistical Test
50	17	7	5	2	5	4	2	3	5	0.000000	FAIL	81/100	FAIL	Frequency (monobit)
82	11	3	3	0	1	0	0	0	0	0.000000	FAIL	54/100	FAIL	Block Frequency (within a block)
14	12	13	11	6	10	7	6	10	11	0.616305	PASS	99/100	PASS	Runs
13	13	11	8	10	12	10	11	5	7	0.719747	PASS	99/100	PASS	Longest Run
10	7	11	14	9	4	13	13	8	11	0.474986	PASS	99/100	PASS	Rank
17	13	10	8	12	7	7	9	7	10	0.401199	PASS	97/100	PASS	FFT
6	10	15	11	12	10	5	11	10	10	0.616305	PASS	100/100	PASS	Linear Complexity
18	13	8	8	15	8	9	4	8	9	0.085587	PASS	96/100	PASS	Non-overlapping Template
17	10	10	15	4	9	10	6	11	8	0.153763	PASS	99/100	PASS	Overlapping Template
10	13	12	8	9	10	9	14	6	9	0.816537	PASS	99/100	PASS	Universal
100	0	0	0	0	0	0	0	0	0	0.000000	FAIL	1/100	FAIL	Serial
82	4	3	4	1	3	0	3	0	0	0.000000	FAIL	53/100	FAIL	Approximate Entropy
49	12	8	9	3	7	3	2	3	4	0.000000	FAIL	80/100	FAIL	Cumulative sums
1	4	4	5	7	4	6	4	6	2	0.559523	PASS	43/43	PASS	Random excursions
2	3	5	5	6	4	4	4	3	7	0.811993	PASS	43/43	PASS	Random excursions variants

Note: the minimum pass rate for each statistical test with the exception of the random excursion (variant) test is approximately = 0.96 for a sample size of 100 binary sequences, and the minimum pass rate for the random excursion (variant) test is approximately = 40 for a sample size of 43 binary sequences.

4. Proposed Post-Processing Algorithms

The post-processing algorithm proposed in this paper is targeted to supplement the randomness of the TRNG among the three characteristics of randomness, unpredictability, and non-reproducibility. The discarding-based post-processing technique was devised in

two steps, and it is largely divided into an algorithm that discards the cell with the worst performance and a bit with poor randomness, individually. The algorithm for discarding cells is mainly focused on solving the process variation issue, and bit discard is designed to eliminate bits with poor randomness due to aging of the TRNG. In the cell-discard algorithm, the cell with the worst performance is discarded by taking advantage of the characteristic that the performance is different for each manufactured cell, as illustrated in Figure 4. In the bit-discard process, bits generated by TRNG undergo a Hamming weight (HW) test and a χ^2 test, and bits that do not pass these two tests are discarded. Here, HW is the Hamming weight used in information theory, and it can be used as an index to determine the randomness of a TRNG. The HW can be expressed as follows:

$$HW = \sum_{i=1}^{N} \left(\frac{x_i}{N} \right)$$

where the i-th bit in an arbitrary bit string composed of N bits is x_i. For example, given that the number of non-zero components in the bit string listed as 0010111 is 4, the HW is 57.14% (=4/7). For an ideal TRNG, the HW converges to 50%.

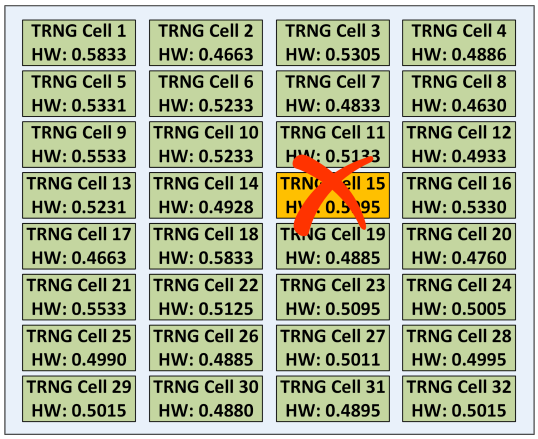

Figure 4. Cell-discard algorithm.

4.1. Cell-Discard Algorithm

A cell-discard algorithm is designed for removing biased cells. In this algorithm, N bits of data are accumulated for each TRNG cell. If the value of the total HW exists within the range of the condition, it is determined that there is no cell with sufficiently random or biased values of random numbers generated from all TRNG cells, and the algorithm for discarding cells is terminated. An example is shown in Figure 4. If the value of the HW does not exist within the range of the condition, the algorithm to discard the cell proceeds. In the algorithm for discarding cells, the HW is calculated for each cell of the TRNG composed of M cells. The cell with the worst HW is discarded, as it is considered to be a biased cell. The cell-discarding algorithm is repeated until the value of the HW satisfies the condition after discarding.

4.2. Bit-Discard Algorithm

An algorithm for enhancing the performance of a cell-discarded TRNG is an algorithm for discarding bits. In the bit-discard algorithm, there are two major bit-discarding processes: discard from a HW test and discard from a Chi-square test. In a bit discard from a HW test, bits that do not satisfy the condition among N bits generated from M TRNG cells are discarded. Bits generated from a TRNG follow the shape of a probability distribution. Therefore, these bits are composed of bits within the range of conditions selected in this paper and bits outside of the range of conditions. If this condition is not satisfied, the corresponding bit string is discarded. If the condition is satisfied, the χ^2 test, which is a uniformity test of the p-value, is conducted. During the χ^2 test, the p-value of the bits is

calculated to verify how evenly the p-value is distributed, and the χ^2 test proceeds in the following order (i.e., N = 10,000).

(1) All N bits are composed in the form of 100-by-100.
(2) Calculate the p-value for 100 bits corresponding to each bit string.
(3) Store the distribution values of each p-value from F_1 to F_{10} (number of samples that satisfy the p-value of the bit).
(4) Calculate χ^2, the p-value from the following function: (igamc: see Appendix A)

$$\chi^2 = \sum_{i=1}^{10} \frac{(F_i - s/10)^2}{s/10}, p\text{-value} = igamc\left(\frac{9}{2}, \frac{\chi^2}{2}\right).$$

(5) Bits with a p-value of less than 0.0001 are discarded, and bits with a value of 0.0001 or more are accepted as the final random number.

5. TRNG Topology

One of the simplest and most widely used TRNG circuits is a cross-coupled inverter. Here, the back-to-back connected inverter has meta-stability or bi-stability, resulting in an opposite digital code at each output terminal. According to the positive feedback paths of each inverter, any form of asymmetric noise during a power-up sequence contributes to generate random output bits [7]. The cross-coupled inverter-based TRNG has the advantage of a simple structure; however, the degree of entropy used by the circuit to generate a random number is too low, making this type vulnerable to the prediction of the security key by an external attacker. In order to increase the level of entropy while generating a random number, time-domain noise can be used along with a supply voltage disturbance during the start-up process. An oscillator-based TRNG is a typical topology which determines random output bits while comparing the frequency difference between two oscillators [23–26].

In this work, a ring-oscillator-based TRNG was designed using a general 65nm CMOS process. Figure 5 shows the basic ring oscillator structure. The three-stage ring oscillator has a structure in which the input and output are connected to each other such that the oscillator continues to oscillate after power is applied. During the oscillation of the ring oscillator, the unpredictable jitter component that arises in the real clock signal is accumulated, as shown in Figure 6. Jitter affects the oscillation frequency of the ring oscillator and is considered a source of entropy. A TRNG with a ring oscillator structure has a simple structure compared to a TRNG with other types of structures, making it easy to access and easy to implement.

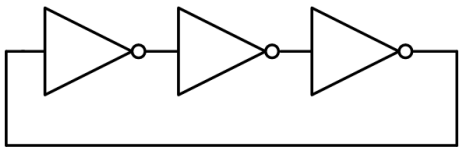

Figure 5. Ring oscillator circuit.

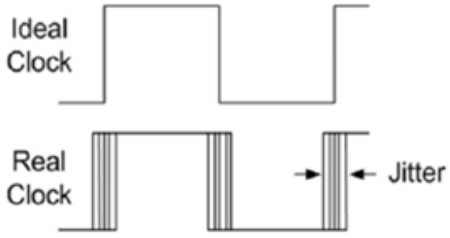

Figure 6. Clock jitter.

Figure 7 shows the entire schematic of the TRNG. Two ring oscillators with different oscillation frequencies enter the D flip-flop input and the clock (CLK) signal to sample the output of the rear stage, finally producing an output, such as a random number. The TRNG

is composed of a 32-bit array in this work. Because the TRNG uses noise existing in the CMOS process as an entropy source, the performance of the 32-bit TRNG structure is better than that of the 1-bit TRNG structure. However, implementing a high-bit TRNG incurs both greater power dissipation and a larger area required.

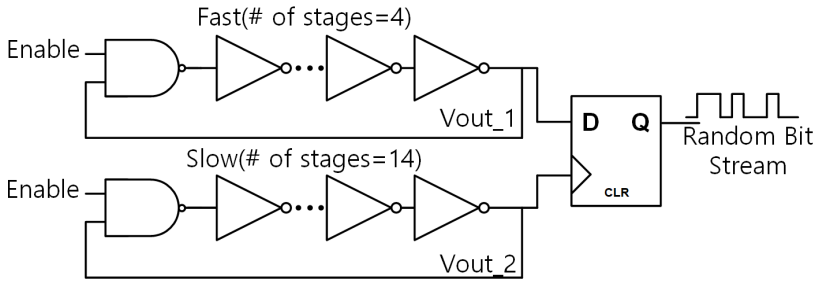

Figure 7. Ring-oscillator-based TRNG.

Figure 8 is a chip micrograph of the proposed TRNG prototype. It is fabricated in the 65nm CMOS process, and the RNG occupies an area of 0.015 μm^2 with a width of 190 μm and length of 80.87 μm.

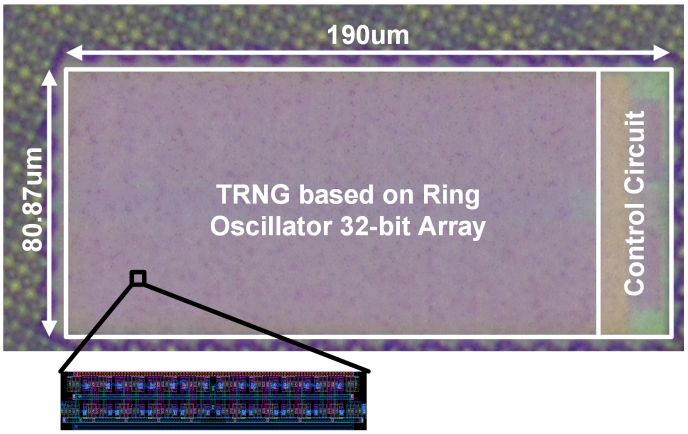

Figure 8. TRNG chip micrograph.

6. Measurements

Figure 9 shows a block diagram of the measurement process and testing environment. The equipment required for the measurements includes the manufactured chip, an interconnection between the evaluation board and external computing resources, an evaluation board to execute the TRNG function based on supervised control, a computer to store the output response, and a power supply unit. In order to perform real-time post-processing according to the initial TRNG bit sequences, MATLAB software is used with on-board memory and a Xilinx FPGA processor (XC351000). The FPGA is programmed by the ISE design suite, which implements a scan-chain function in the testing interface (i.e., the UART interface in this work).

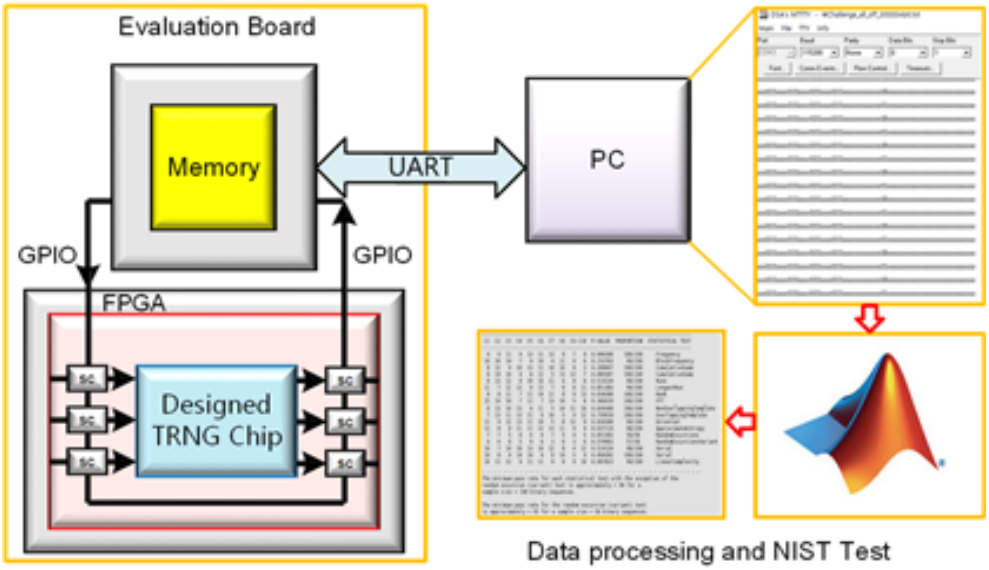

Figure 9. Testing environments.

A NIST 800-22a test was performed by acquiring random numbers from a TRNG prototype. The test was implemented in C language [16] and was conducted on the Linux operating system. The collected random number bit sequence is 10^8 bits, and the significance level of α was set to 0.01 (i.e., $N = 100$).

The test results indicate remarkable performance in terms of unpredictability and non-reproducibility of the random numbers, such as in Rank and FFT tests among 15 NIST 800-22a tests as summarized in Table 2. However, some random number tests, such as frequency and cumulative sums, could not meet the desired performance level. The goal of the discarding-based post-processing circuit method proposed in this paper is the passing of all tests by improving the randomness of random numbers by erasing bits that do not meet the test requirements.

Table 2. Test results of intrinsic TRNG bit sequences.

Test	*p*-Value	Proportion
Frequency (monobit)	0.000000(FAIL)	94/100 (FAIL)
Block Frequency	0.337692 (PASS)	96/100 (PASS)
Runs	0.599313 (PASS)	96/100 (PASS)
Longest Runs	0.019951 (PASS)	99/100 (PASS)
Rank	0.639667 (PASS)	99/100 (PASS)
FFT	0.256891 (PASS)	96/100 (PASS)
Linear Complexity	0.254129 (PASS)	94/100 (FAIL)
Non overlapping Template	FAIL (sub-test 148)	
Overlapping Template	0.153763 (PASS)	99/100 (PASS)
Universal	0.351900 (PASS)	96/100 (PASS)
Serial	0.315716 (PASS)	96/100 (PASS)
Approximate Entropy	0.329154 (PASS)	96/100 (PASS)
Cumulative sums	0.000000 (FAIL)	93/100 (FAIL)
Random excursions	PASS (sub-test 8)	
Random excursions variants	PASS (sub-test 18)	

Before applying the cell-discard algorithm, it is necessary to calculate the values required for the testing condition. Figure 10 presents a flow chart of the cell-discard algorithm. First, N-bit data are accumulated for each cell of the TRNG. According to the number of operating cycle (N) sweeps, the calculated HW for each cell becomes saturated at around 1000, as shown in Figure 11; thus, we decided to set N to 1000 in this work. Next, it is necessary to determine the condition value for determining the randomness of the bits

generated in each cell. The condition value is the minimum Hamming weight for all bits (M is the number of TRNG cells), and the first condition value of α can be calculated as 0.072 when $M = 32$ according to the following equation:

$$p\text{-value} = \text{erfc}\left(\frac{|2HW - 1| \times B}{\sqrt{2B}}\right),$$

where erfc is an error function and B is the length of the random number bit sequence. Assuming that the cell-discard algorithm has progressed and a single cell has been discarded, $M = 31$, and the second condition value β is calculated as 0.073. The cell-discard algorithm continues until the condition value of the entire HW is meets the requirements.

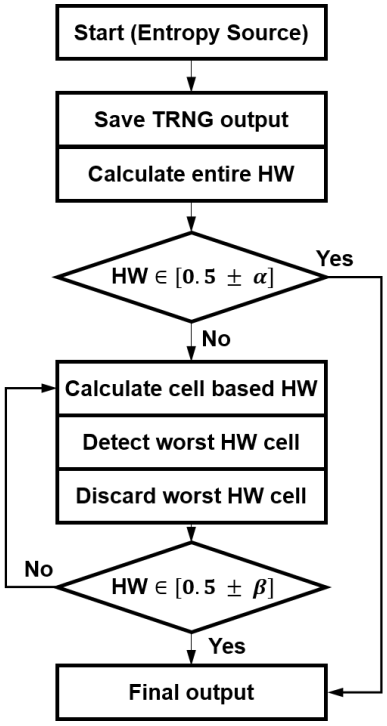

Figure 10. Cell-discard algorithm flow chart.

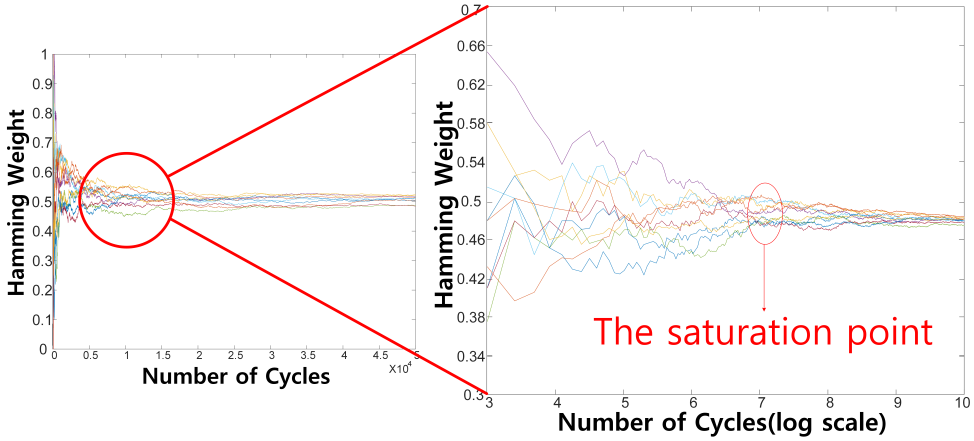

Figure 11. Hamming weight versus the operating cycle (N).

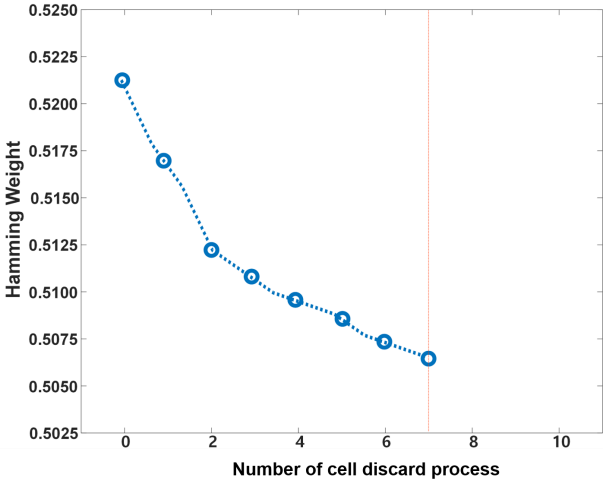

Figure 12. Hamming weight versus the number of cell-discard processes.

Figure 12 shows the result when applying the proposed cell-discard algorithm. The HW for each bit before applying the cell-discard algorithm reached the target range of the HW ($0.4928 \leq$ total HW ≤ 0.5070) after performing the cell-discard process seven times. Because seven cells were discarded, the total number of discarded bits is 2.1×10^7 bits, and approximately 21% of the bits were discarded.

Table 3 shows the results of the NIST test using bits after applying the cell-discard algorithm. The performance of the HW was improved compared to that before the discard algorithm was applied (see Table 2); however, the performance was not improved enough to pass all of the NIST tests.

Table 3. Test Results of Cell-discarded TRNG Bit Sequences.

Test	*p*-Value	Proportion
Frequency (monobit)	0.000000 (FAIL)	88/100 (FAIL)
Block Frequency	0.000000 (FAIL)	76/100 (FAIL)
Runs	0.816305 (PASS)	98/100 (PASS)
Longest Runs	0.611325 (PASS)	100/100 (PASS)
Rank	0.647689 (PASS)	98/100 (PASS)
FFT	0.509921 (PASS)	98/100 (PASS)
Linear Complexity	0.535311 (PASS)	100/100 (PASS)
Non overlapping Template	PASS (sub-test 148)	
Overlapping Template	0.256713 (PASS)	100/100 (PASS)
Universal	0.732523 (PASS)	99/100 (PASS)
Serial	0.000000 (FAIL)	66/100 (FAIL)
Approximate Entropy	0.000000 (FAIL)	72/100 (FAIL)
Cumulative sums	0.123551 (PASS)	98/100 (PASS)
Random excursions	PASS (sub-test 8)	
Random excursions variants	PASS (sub-test 18)	

Before applying the bit-discard algorithm, it is necessary to calculate the values required for the condition. Figure 13 presents the bit-discard algorithm. In order to proceed with the bit-discard algorithm, N bits must initially be obtained from the TRNG. The bit-discard algorithm is mainly divided into a HW test and a χ^2 test. According to Section 4.2, the minimum number of tests required for the χ^2 test is 100, and it is necessary to calculate the HW per test, as in the cell-discard algorithm. Because the minimum number of bits is 1000, the value of N is set to 100,000. When all 100,000 data instances are collected, the HW test is performed. The HW test requires a conditional value, γ, and γ for 100,000 bits is 0.0041. Therefore, if the condition ($0.4959 \leq HW \leq 0.5041$) is not satisfied, the generated random number sequence is discarded. If the condition is satisfied, the HW test of the

bit-discard process is finished, and the χ^2 test that determines the uniformity of the p-value is conducted. The χ^2 test proceeds in the order of (1) to (5), as introduced in Section 4.2. The bit string that has passed the χ^2 test is adopted as the final random number.

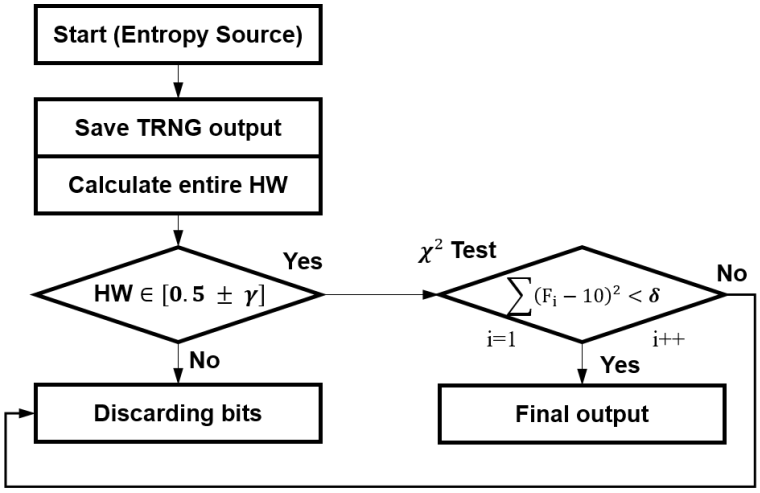

Figure 13. Bit-discard algorithm flow chart.

As a result of applying the bit-discard algorithm, the HW for all bits before applying the algorithm was improved from 0.5215 to 0.5025. The total number of discarded bits is 2.1×10^7, and approximately 21% of the bits was discarded. In the HW test, the overall HW was improved from 0.5215 to 0.5050, and about 15% of 1.5×10^7 bits was discarded. In the χ^2 test, the HW was improved from 0.5050 to 0.5025, and about 6% of 0.6×10^7 bits was discarded additionally. Table 4 shows the results of the NIST test using bits after applying the bit-discard algorithm. Compared to the initial NIST test (see Table 2), the performance of the HW was improved, and more tests were passed. However, the performance was still not improved enough to pass all of the NIST tests.

Table 4. Test results of bit-discarded TRNG bit sequences.

Test	p-Value	Proportion
Frequency (monobit)	0.000000 (FAIL)	92/100 (FAIL)
Block Frequency	0.000000 (FAIL)	88/100 (FAIL)
Runs	0.421350 (PASS)	99/100 (PASS)
Longest Runs	0.121661 (PASS)	98/100 (PASS)
Rank	0.511202 (PASS)	96/100 (PASS)
FFT	0.660332 (PASS)	100/100 (PASS)
Linear Complexity	0.115230 (PASS)	100/100 (PASS)
Non overlapping Template	PASS (sub-test 148)	
Overlapping Template	0.011225 (PASS)	98/100 (PASS)
Universal	0.672251 (PASS)	97/100 (PASS)
Serial	0.844125 (PASS)	96/100 (PASS)
Approximate Entropy	0.010203 (PASS)	97/100 (PASS)
Cumulative sums	0.886615 (PASS)	96/100 (PASS)
Random excursions	PASS (sub-test 8)	
Random excursions variants	PASS (sub-test 18)	

Figure 14 shows the algorithm when both the cell- and bit-discard algorithms are applied. The HW for 1E8 bits before application was improved from 0.5215 to 0.5003. The total number of discarded bits is 2.5×10^7, and about 25% of the bits were discarded. In the cell-discard algorithm, the overall HW was improved from 0.5215 to 0.5070, and about 21% of 2.1×10^7 bits was discarded. In the bit-discard algorithm, the HW was improved from 0.5070 to 0.5003, and about 4% of 0.4×10^7 bits was discarded. Table 5 shows the NIST test results for random numbers after applying both the cell- and bit-discard algorithms in the

proposed discarding-based post-processing technique. The p-value of the tested random number must be 0.0001 or higher, and the Proportion value must be 96/100 or higher to pass the test. After applying the post-processing technique, all 15 tests were passed.

Table 5. Test results of both cell- and bit-discarded TRNG bit sequences.

Test	p-Value	Proportion
Frequency (monobit)	0.946308 (PASS)	100/100 (PASS)
Block Frequency	0.153763 (PASS)	99/100 (PASS)
Runs	0.514124 (PASS)	99/100 (PASS)
Longest Runs	0.851383 (PASS)	99/100 (PASS)
Rank	0.834308 (PASS)	100/100 (PASS)
FFT	0.366918 (PASS)	100/100 (PASS)
Linear Complexity	0.997823 (PASS)	99/100 (PASS)
Non overlapping Template	PASS (sub-test 148)	
Overlapping Template	0.739918 (PASS)	100/100 (PASS)
Universal	0.834308 (PASS)	99/100 (PASS)
Serial	0.514124 (PASS)	98/100 (PASS)
Approximate Entropy	0.637119 (PASS)	98/100 (PASS)
Cumulative sums	0.289667 (PASS)	100/100 (PASS)
Random excursions	PASS (sub-test 8)	
Random excursions variants	PASS (sub-test 18)	

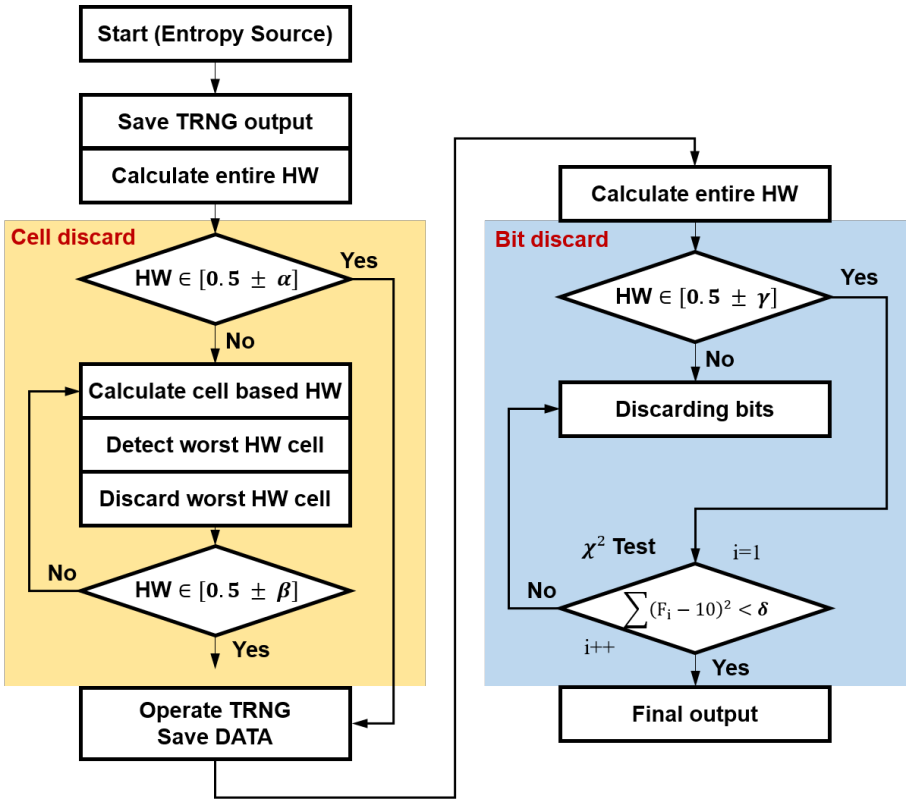

Figure 14. Flow chart of the entire proposed discard algorithm.

Table 6 is a comparison table of the results of the HW and NIST 800-22a tests before and after applying the proposed discarding-based post-processing algorithms to each chip manufactured via an identical process. When only the cell-discard algorithm is applied, the performance of the HW is improved; however, the post-processed bit sequences could not pass the NIST requirements, and the bit-discard algorithm also produces results identical to those in the stand-alone cell-discarded case. When both the cell and bit-discard algorithms are applied together, the Hamming weight approaches the ideal value and

the post-processed bit stream can pass all of the NIST tests. For chip-2, which had the best performance in terms of the Hamming weight, the Hamming weight was improved to 0.5003 after applying the post-processing technique. In the case of chip-5, where the performance of the Hamming weight without the post-processing technique was the worst, however, it was improved to 0.5008 after the application of post-processing. As a result, we satisfied all NIST tests performed on five different prototype chips.

Table 6. Performance comparison among five different TRNG chips.

Performance		Chip-1	Chip-2	Chip-3	Chip-4	Chip-5
Intrinsic TRNG	Hamming Weight	0.5278	0.5215	0.5370	0.5441	0.5445
	Number of entire bits	10^8	10^8	10^8	10^8	10^8
	NIST test results	5-Fail	5-Fail	5-Fail	5-Fail	5-Fail
Applying Cell discard Algorithm	Hamming Weight	0.5069	0.5070	0.5070	0.5065	0.5072
	Discarded bits	1.5×10^7	2.1×10^7	2.1×10^7	2.1×10^7	2.5×10^7
	NIST test results	4-Fail	4-Fail	4-Fail	4-Fail	4-Fail
Applying Bit discard Algorithm	Hamming Weight	0.5015	0.5025	0.5030	0.5026	0.5033
	Discarded bits	1.4×10^7	1.5×10^7	1.6×10^7	1.6×10^7	1.5×10^7
	NIST test results	2-Fail	2-Fail	2-Fail	3-Fail	3-Fail
Applying Cell and Bit discard Algorithm	Hamming Weight	0.5008	0.5003	0.5004	0.5006	0.5008
	Discarded bits	2.7×10^7 (27%)	2.5×10^7 (25%)	2.8×10^7 (28%)	2.8×10^7 (28%)	2.85×10^7 (28.5%)
NIST test results		PASS	PASS	PASS	PASS	PASS

7. Conclusions

We demonstrate a discarding-based post-processing technique to improve the performance of the intrinsic TRNG. The intrinsic TRNG adopts the conventional ring oscillator structure and is designed as a 32-bit array to increase randomness. The discarding-based post-processing technique consists of two major steps: cell- and bit-level discarding. In the cell-discard algorithm, the controller sequentially calculates the HW for each 32-bit structured intrinsic TRNG cell and discards cells that do not satisfy the target HW. The bit-discard algorithm is further divided into a HW test and a χ^2 test. In the HW evaluation, in case of the condition being satisfied by calculating the total HW for 10,000 generated bits in every TRNG cell, the process moves to the χ^2 test. On the other hand, when the condition is not satisfied, the corresponding bits are discarded. In the χ^2 test, bits are reconstructed in a 100×100 form to determine whether the condition is satisfied again. Bits satisfying the condition are adopted as the final random number. The random number performance was evaluated with five manufactured chips to verify the discarding-based post-processing algorithm using a FPGA processor. Owing to the cell and bit-discard algorithm, the randomness of the intrinsic TRNG becomes much stronger, thus resulting in the passing of all NIST criteria while sacrificing less than 30% of bits.

Author Contributions: Conceptualization, J.-P.H.; methodology, J.-P.H.; software, J.K.; validation, J.K.; formal analysis, J.-W.N.; investigation, J.W.-N.; resources, J.K. and J.-P.H.; data curation, J.K. and J.-P.H.; writing—original draft preparation, J.-W.N.; writing—review and editing, J.-W.N.; visualization, J.-W.N. and J.K.; supervision, J.-P.H.; project administration, J.-P.H.; funding acquisition, J.-P.H. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (2021R1A2C2005258) and the MSIT(Ministry of Science and ICT), Korea, under the Grand Information Technology Research Center support program (IITP-2022-2020-0-01462) supervised by the IITP(Institute for Information communications Technology Planning Evaluation). The EDA tool was supported by the IC Design Education Center(IDEC), Korea.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A

This appendix briefly introduces the functions used in this paper. The error function expresses the probability that an error of magnitude z occurs. In this paper, the complementary error function is defined as follows:

$$\operatorname{erfc}(z) = \frac{2}{\sqrt{\pi}} \int_z^{\infty} e^{u^2} du.$$

Unlike natural numbers, the gamma function is a generalized function for the number of regions that cannot be defined as a factorial function, such as integers, rational numbers, and real numbers, as follows:

$$P(s,x) = \Gamma(s,x) = \int_x^{\infty} t^{s-1} e^{-t} dt.$$

The incomplete gamma function is a special function that extends the gamma function, and the integration interval is changed from the original gamma function definition. The incomplete gamma function includes an upper incomplete gamma function and a lower incomplete gamma function. The upper incomplete gamma function is defined as

$$P(s,x) = \frac{\gamma(s,x)}{\Gamma(s)} = \frac{1}{\Gamma(s)} \int_0^{x} t^{s-1} e^{-t} dt,$$

whereas the lower incomplete gamma function is defined as

$$Q(s,x) = 1 - P(s,x) = \frac{\Gamma(s,x)}{\Gamma(s)} = \frac{1}{\Gamma(s)} \int_x^{\infty} t^{s-1} e^{-t} dt,$$

$$Q(s,0) = 1,\ Q(s,\infty) = 0,$$

where $s \in \mathbb{C}$ and $\mathrm{Re}(s) \leq 0$.

References

1. Liu, Z.; Jin, H.; Hu, Y.-C.; Bailey, M. Practical proactive DDoS-attack mitigation via endpoint-driven in-network traffic control. *IEEE/ACM Trans. Netw.* **2018**, *26*, 1948–1961. [CrossRef]
2. Cha, B.; Kim, K.; Na, H. Random password generation of OTP system using changed location and angle of fingerprint feature. In Proceedings of the 2008 8th IEEE International Conference on Computer and Information Technology, Sydney, Australia, 8–11 July 2008; pp. 420–425.
3. Gutterman, Z.; Pinkas, B.; Reinman, T. Analysis of the Linux RNG. In Proceedings of the 2006 IEEE Symposium on Security and Privacy (S&P'06), Berkeley/Oakland, CA, USA, 21–24 May 2006; pp. 371–385.
4. Forouzan, B.A.; Fegan, S.C. *Data Communications and Networking*; Huga Media: Berkeley, CA, USA, 2007.
5. Ziemer, R.E.; Tranter, W.H. *Principles of Communications*; John Wiley & Sons: Hoboken, NJ, USA, 2014.
6. Menezes, A.J.; Oorschot, P.C.; Vanstone, S.A. *Handbook of Applied Cryptography*; CRC Press: Boca Raton, FL, USA, 2016.
7. Mathew, S.K.; Srinivasan, S.; Anders, M.A.; Kaul, H.; Hsu, S.K.; Sheikh, F.; Agarwal, A.; Satpathy, S.; Krishnamurthy, R.K. 2.4 Gbps 7 mW All-Digital PVT-Variation Tolerant True Random Number Generator for 45nm CMOS High-Performance Microprocessors. *IEEE J. Solid-State Circuits* **2012**, *47*, 2807–2821. [CrossRef]
8. Poudel, B.; Munir, A. Design and Evaluation of a PVT Variation-Resistant TRNG Circuit. In Proceedings of the 2018 IEEE 36th International Conference on Computer Design (ICCD), Orlando, FL, USA, 7–10 October 2018; pp. 514–521.
9. Sreekumar, L.; Ramesh, P. Selection of an optimum entropy source design for a true random number generator. *Procedia Technol.* **2016**, *25*, 598–605. [CrossRef]
10. Muthukumar, A.; Sivasankari, N.; Rampriya, K. Anti-aging true random number generator for secured database storage. In Proceedings of the 2017 4th International Conference on Advanced Computing and Communication Systems (ICACCS), Coimbatore, India, 6–7 January 2017; pp. 1–7.
11. Cartagena, J.; Gomez, H.; Roa, E. A fully-synthesized TRNG with lightweight cellular-automata based post-processing stage in 130nm CMOS. In Proceedings of the IEEE Nordic Circuits and Systems Conference (NORCAS), Copenhagen, Denmark, 1–2 November 2016; pp. 1–5.

12. Nam, J.-W.; Ahn, J.-H.; Hong, J.-P. Compact SRAM-Based PUF chip employing body voltage control technique. *IEEE Access* **2022**, *10*, 22311–22319. [CrossRef]
13. Baek, S.; Yu, G.-H.; Kim, J.; Ngo, C.T.; Eshraghian, J.K.; Hong, J.-P. A reconfigurable SRAM based CMOS PUF with challenge to response pairs. *IEEE Access* **2021**, *9*, 79947–79960. [CrossRef]
14. Choi, K.-U.; Baek, S.; Heo, J.; Hong, J.-P. A 100% stable sense-amplifier-based physically unclonable function with individually embedded non-volatile memory. *IEEE Access* **2020**, *8*, 21857–21865. [CrossRef]
15. Kiamehr, S.; Golanbari, M.S.; Tahoori, M.B. Leveraging aging effect to improve SRAM-based true random number generators. In Proceedings of the Design, Automation & Test in Europe Conference & Exhibition (DATE), Lausanne, Switzerland, 27–31 March 2017; pp. 882–885.
16. Min, L.; Chen, T.; Zang, H. Analysis of fips 140-2 test and chaos-based pseudorandom number generator. *Chaotic Model. Simul. (CMSIM)* **2013**, *2*, 273–280.
17. Brown, R.G.; Eddelbuettel, D.; Bauer, D. *Dieharder*; Duke University Physics Department Durham: Durham, NC, USA, 2018.
18. *AIS 31: Functionality Classes and Evaluation Methodology for Physical Random Number Generators, Version 1*; Federal Office for Information Security: Bonn, Germany, 2001. Available online: http://www.bsi.bund.de/zerti.z/zert/interpr/ais31e.pdf (accessed on 10 October 2021).
19. *SP 800-90A Rev. 1*; Recommendation for Random Number Generation Using Deterministic Random Bit Generators. National Institute of Standards and Technology: Gaithersburg, MD, USA, 2015.
20. *SP 800-90B*; Recommendation for the Entropy Sources Used for Random Bit Generation. National Institute of Standards and Technology: Gaithersburg, MD, USA, 2018.
21. *SP 800-90C (Draft)*; Recommendation for Random Bit Generator (RBG) Constructions. National Institute of Standards and Technology: Gaithersburg, MD, USA, 2016.
22. *800-22*; A Statistical Test Suite for the Validation of Random Number Generators and Pseudo Random Number Generators for Cryptographic Applications. National Institute of Standards and Technology: Gaithersburg, MD, USA, 2010.
23. Dichtl, M.; Golic, J.D. High-speed true random number generation with logic gates only. In Proceedings of the International Workshop on Cryptographic Hardware and Embedded Systems, Vienna, Austria, 10–13 September 2007; Springer: Berlin/Heidelberg, Germany, 2007; pp. 45–62.
24. Yang, Y.; Jia, S.; Wang, Y.; Zhang, S.; Liu, C. A reliable true random number generator based on novel chaotic ring oscillator. In Proceedings of the 2017 IEEE International Symposium on Circuits and Systems (ISCAS), Baltimore, MD, USA, 28–31 May 2017; pp. 1–4.
25. Liu, X.; Jia, S.; Zhang, H. A novel high-speed fpga-based true random number generator based on chaotic ring oscillator. In Proceedings of the 2019 IEEE 13th International Conference on ASIC (ASICON), Chongqing, China, 29 October–1 November 2019; pp. 1–4.
26. Bucci, M.; Germani, L.; Luzzi, R.; Trifiletti, A.; Varanonuovo, M. A high-speed oscillator-based truly random number source for cryptographic applications on a smart card IC. *IEEE Trans. Comput.* **2003**, *52*, 403–409. [CrossRef]