International Journal of Networking and Computing – www.ijnc.org, ISSN 2185-2847
Volume 12, Number 2, pages 372–386, July 2022

Implementation and Evaluation of Ring Oscillator-based True Random Number Generator

Naoya Torii, Ryuichi Minagawa, Hideaki Kevin Omae, and Kotaro Hayashi
SOKA University,
Hachiouji, Tokyo, 192-8577, Japan

Received: February 14, 2022
Revised: April 29, 2022
Accepted: June 5, 2022
Communicated by Yasuyuki Nogami

Abstract

A true random number generator (TRNG) is suitable for generating secure keys and nonces. TRNGs implemented in IoT devices must be small in scale, have low power consumption, and be feasible. The random number sequence generated by TRNG needs to have high entropy immediately after startup and a stable state. This paper implements three types of ring oscillator type TRNGs, TERO-based, COSO-based, and STR-based TRNG, on Zynq-7010. When these TRNGs are implemented as a single entropy source, it is challenging to implement them because evaluating the layout and wiring for each FPGA is necessary. This paper evaluates a TRNG configuration, which exclusively ORs the outputs of multiple entropy sources. We show that this configuration can reduce the implementing difficulty and realize high entropy. For the random number sequence evaluation, we use the statistical test of NIST SP800-90B, SP800-22, and BSI AIS 20/31. In addition, the random number sequence immediately after the startup is also statistically evaluated. As a result, our evaluated TRNGs generate high entropy random numbers. They are easy to implement on FPGA when we implement TRNGs with eight single entropy sources for TERO-based TRNG, 48 for COSO-based TRNG, and two for STR-based TRNG, respectively.

Keywords: true random number generator, TRNG, health test, SP800-90B, SP800-22, AIS 20/31

1 Introduction

The Internet of Things (IoT) system is attracting attention as a system that provides new services. Security is one of the important requirements for IoT systems. In addition, sensors on IoT systems need to realize security with small-scale and low-power resources.

A random number sequence with high entropy is required to generate nonces and keys for cryptographic processing in IoT devices. There are three requirements for random number sequences used in security: randomness, non-reproducibility, and unpredictability.

There are two types of random number generators. One is a pseudo-random number generator (PRNG) that generates random numbers using an algorithm. The other is a true random number generator (TRNG) that uses analog noise from physical phenomena. TRNG meets the above three requirements.

Two ways of implementing TRNG are used: One is to use an analog circuit and retrieve an entropy source [1, 2], such as thermal noise. The other is to use a digital circuit and retrieve an entropy source [3, 4, 5, 6, 7, 8, 9], such as its clock jitter and its metastability. The digital circuit type TRNG is relatively low-cost. Therefore, this type is suitable for IoT devices.

In addition, TRNG implemented in IoT devices is expected to generate random numbers with high entropy immediately after startup with low power consumption and small scale.

Petura et al.[10] reported that they implemented five types of Ring Oscillator (RO) based TRNG on Field Programmable Gate Array (FPGA) and evaluated the random number using statistical tests in German standard AIS 20/31 [11]. Among five TRNGs, they describe that the entropies vary greatly depending on the layout and wiring. Especially for the Transition Effect RO (TERO) based, the Coherent Sampling RO (COSO) based, and the Self-timed Ring (STR) based TRNG. Great labor needs to find suitable to find suitable parameters to get high entropy random number sequences. We consider that the feasibility of TRNG in IoT devices is also one of the essential factors because IoT devices will be implemented in large numbers.

There are two types of countermeasures against the problem. The first measure is to analyze the TRNG output and adjust the TRNG parameters to absorb individual differences [12, 13]. The second measure is to prepare multiple TRNG entropy sources and combine the output data to obtain stable and high entropy [3, 7]. Multiple TRNG entropy sources increase the possibility of including good entropy sources because they are implemented in various layouts and wirings.

This paper applies the second measure. We evaluate a TRNG configuration that exclusively ORs the outputs of multiple entropy sources, TERO-based, COSO-based, and STR-based TRNG. This configuration enables lightening the labor to find suitable parameters and increasing the feasibility, whereas the hardware size increase. The contribution of this paper is that we clarify the number of random sources in this configuration required to obtain a high entropy and evaluate whether the increase of the hardware is suitable for IoT devices.

We implemented these TRNGs on FPGA and estimated their statistical properties. For evaluation in a stable state, we use the AIS 20/31, which includes statistical tests for TRNG and the NIST SP800-90B [14], which includes independent identically distributed (IID) random sequences tests for TRNG. In addition, we use the NIST SP800-22 [15], which is statistical package consist of 15 tests. For evaluation at startup, we use health test and restart the test by SP800-90B, pattern counting health test used by Intel [16], autocorrelation coefficient, and frequency test.

The structure of this paper is as follows. In Section 2, we describe the evaluated configuration and implementation of TRNG. Section 3 describes the statistical tests to perform randomness evaluation. In Section 4, we evaluate the performance of TERO-based, COSO-based, and STR-based TRNG. In Section 5, we discuss the performance evaluated and future studies. Section 6 concludes the paper.

2 TRNG configuration

2.1 Basic components of TRNG

Figure 1 shows a general TRNG block diagram [14]. It consists of three blocks: entropy source, conditioning, and health tests. The entropy source generates raw random numbers. If there is a correlation or bias in them, the conditioning block reduces the bias of the raw data by compression processing and increases the entropy. If the original random number generated by the entropy has sufficient entropy, the conditioning circuit is unnecessary; therefore, it is optional. The health tests block outputs an error message when the entropy source cannot generate a random number with sufficient entropy due to failure or other reasons.

2.2 TRNG Principle and Implementation Method

This section describes the principle and structure of the TERO-based, COSO-based, and STR-based TRNGs. It is reported that the entropy of these TRNGs changes greatly depending on the circuit characteristics [10]. Therefore, implementing TRNG on FPGA, it is necessary to specify the layout and wiring to generate a high entropy random number, and it requires great labor.

We evaluate a TRNG configuration for TERO-based, COSO-based, and STR-based TRNG. This configuration prepares plural single entropy sources (SES), and the outputs are exclusively ORed (XORed) to generate random numbers, as shown in Figure 2. Compared to the single SES

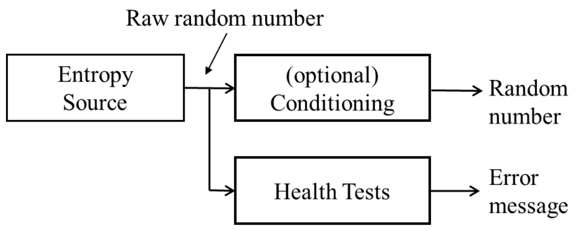

Figure 1: TRNG Block diagram [14]

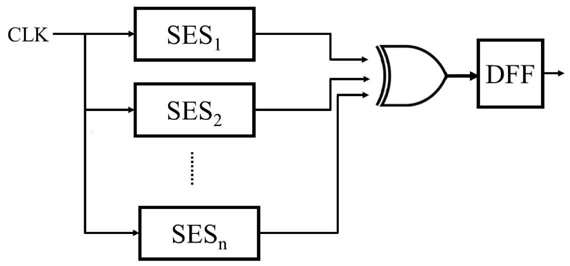

Figure 2: Multiple Entropy Sources (MES) configuration

implementation, multiple SES increases the possibility of including good SES that generates high entropy random numbers because of various layouts and wirings.

This configuration was applied to RO-based TRNG [3] and latch-based TRNG [7]. However, it does not apply to TERO-based, COSO-based, and STR-based TRNG. In the following, this configuration is called the Multiple Entropy Sources (MES) configuration.

We determine n_{ses}, which means the number of SES in the MES by AIS 20/31 and SP800-90B statistical tests. First, we implement MES-TRNG with a relatively large n_{ses} on the FPGA. Using the TRNG, we test random numbers from the TRNG with n_{ses} SES from a small number to larger to decide the minimum number to pass the statistical test.

2.2.1 TERO-based TRNG

The TERO-based TRNG [5] uses metastability as an entropy source. It consists of two NANDs and an even number of inverters or buffers.

Figure 3 shows the SES block of the TERO-based TRNG. By inputting signals to the two NAND gates, the circuit starts oscillating and enters the metastable state. After that, due to various factors such as variations in the gate elements and wiring capacitance, the circuit will eventually transition from the metastable state to the stable state. At this time, the number of oscillations until the transition to the stable state is random. Therefore, a 1-bit random number can be obtained by determining whether this is an even number or an odd number using T flip-flop (TFF). The output is sampled at the system clock divided using a D flip-flop (DFF).

In our implementation, the number of buffers is set to 3 and 5, respectively. This implementation, proposed by Fujieda [12], enables the number of oscillations not to become too large by intentionally increasing the propagation delay of the circuit.

In the SES design, the relative positions were specified to place each element close to the other. The MES was implemented with eight SES ($n_{ses} = 8$). The placement of each SES was automatically wired in the FPGA development environment.

2.2.2 COSO-based TRNG

The COSO-based TRNG [6] uses coherent sampling and enables low power consumption on a small circuit scale. It consists of two independent RO, DFF and TFF, and generates random numbers by

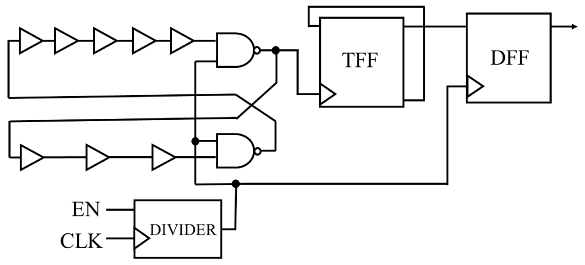

Figure 3: TERO-based TRNG [10]

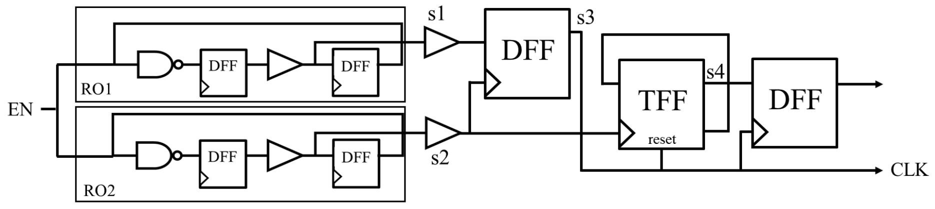

Figure 4: COSO-based TRNG [6] [10]

extracting the jitter in the signal oscillated by the two ROs.

Figure 4 shows the SES block diagram of the COSO-based TRNG. The $s1$ clock by RO1 is sampled by DFF using the $s2$ clock by RO2 to extract the beat signal $s3$. Then, the $s3$ signal is used to reset the output $s4$ of the 1-bit counter (TFF) for $s2$ clock and latches the $s4$ to output a random bit.

The following conditions must be met to generate random numbers [10].

$$\Delta_T < \sqrt[3]{\sigma_T^2 T} = \Delta_{T_{max}} \tag{1}$$

where the difference between the cycles of the two ROs is Δ_T, the cycle of RO1 is T, and the variance of the cycles is σ_T^2. To satisfy this condition, it is necessary to optimize the layout and wiring for each chip manually, and these processes are laborious.

In our implementation, MES consists of 48 SES ($n_{ses} = 48$). In the SES design, we specified relative positions of the RO and the latch to be placed in one slice. Then the automatic wiring of the development environment is used for the layout of each SES.

2.2.3 STR-based TRNG

STR-based TRNG [8, 9] uses a self-timed ring (STR). STR is a multi-event oscillator with no signal collision. Each of the L stages of STR consists of a 2-input Muller gate and an inverter. Figure 5 shows the SES block diagram of STR-based TRNG.

Each stage communicates using the two-phase handshake protocol. In addition, E out of L stages is initialized when the Muller gate output becomes one and propagation starts. Regardless of the initial number of events, the Charlie effect and the drafting effect eventually shift to a steady-state. There are two modes of oscillation. The first is a burst oscillation mode in which multiple events form a cluster. The second is an evenly spaced oscillation mode in which events spread evenly and propagate at regular time intervals. Generally, when creating a random number generator, it is desirable to use the evenly spaced oscillation mode.

If the number of stages and the number of events are relatively prime, STR shows the same number of phases at different intervals. Assuming that the number of stages is L and the oscillation period of STR is T, the phase decomposition can be expressed by the following equation [9].

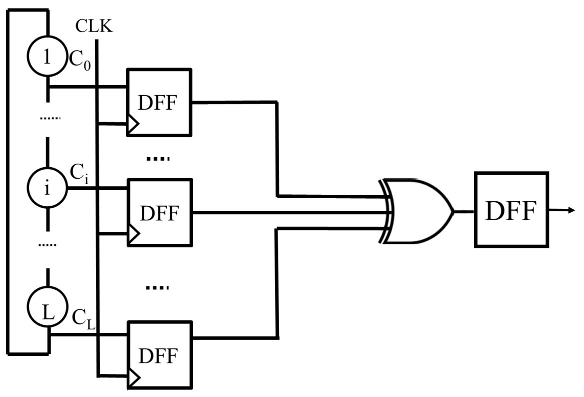

Figure 5: STR-based TRNG[8, 10]

$$\Delta\phi = \frac{T}{2L} \tag{2}$$

The STR stage is sampled by DFF. Their outputs are XORed and sampled at the same clock to generate random numbers. The following equation must be satisfied to sufficiently guarantee the entropy rate. σ_{acc} is the standard deviation of jitter.

$$\Delta\phi < \sigma_{acc} \tag{3}$$

The Muller gate was implemented using LUT, and the relative position was specified so that the Muller gate and DFF were set in a slice. The STR-based TRNG was implemented at $L = 63$. However, a single STR-based TRNG could not generate a random number that would pass the statistical test. Therefore, we implemented MES consists of two STR-based TRNGs ($n_{ses} = 2$) with $L = 63$. For the placement of each SES, automatic wiring in the development environment was used. The oscillation frequency of STR was about 400 MHz.

3 Random Number Evaluation Method

3.1 Stable state Evaluation

This paper evaluates the random number sequence at a stable state using AIS 20/31, SP800-90B and SP800-22.

3.1.1 Statistical Tests by AIS 20/31

AIS20/31 includes eight statistical tests from T0 to T8. We evaluated the random numbers in accordance with test procedure A and B specified in AIS 20/31. In test procedure A, the random numbers are evaluated by statistical tests from T0 to T5. In test procedure B, the random numbers are evaluated by statistical tests from T6 to T8. Especially, Test T8 is entropy estimation, and it outputs entropy of the tested sequence. If the tested random number is ideal, the probability of passing these tests is almost 0.9998, while the probability of failing more than one test is almost 0. Passing the statistical tests is one of the requirements for the random number generator class PTG.2 in AIS 20/31. The class PTG.2 random number is used for generating cryptographic keys, nonces, and the seed of pseudo-random numbers generators.

3.1.2 IID Test by SP800-90B

SP800-90B specifies the design principles and requirements for TRNG and the tests for the validation of TRNG. We use the tests for the IID (independently and uniformly distributed) track in SP800-90B. IID is defined as "a sequence of random variables for which each element of the sequence has the same probability distribution as the other values, and all values are mutually independent" in SP800-90B [14]. IID validation tests include shuffling tests on independence and stability (six tests) and specific statistical tests (two tests). We used the SP800-90B Entropy Assessment library [17]. The IID test outputs the min-entropy of the random number.

3.1.3 SP800-22

SP800-22 describes statistical test suite to test random and pseudorandom number generators for cryptographic application [15]. We evaluated the random numbers by using this suite for reference because these tests are said to be useful for analyzing random number sources in AIS 20/31. It is a statistical test package to test the randomness of sequences that consists of 15 tests, including Frequency test, BlockFrequency test, etc. We used NIST statistical test suite called "sts-2.1.2." [18] and the recommended parameter settings specified by AIS 20/31. The size of each data set was about 2^{30} bits.

3.2 Startup Evaluation

To evaluate the random numbers immediately after startup, we used five statistical tests; the health test and restart test shown by SP800-90B, the pattern counting health test used by Intel, the autocorrelation coefficient, and the frequency test.

3.2.1 SP800-90B Health Test

The two health tests are described in SP800-90B. One is the Repetition Count Test, and the other is the Adaptive Count Test. These tests are for watching the rapid decrease of the entropy source of TRNG and for a startup test.

Repetition Count Test The goal of the repetition count test is to quickly detect a catastrophic failure that causes the noise source to generate a single output value for a long time. The test procedure is as follows. If the same value (0 or 1) appears consecutively c times or more in a sequence of random numbers, the random numbers are a failure, where $c = 1 + \lceil 30/(min - entropy) \rceil$ [14]. In this paper, min-entropy is 0.99 described in Section 3.1.2 and c is calculated as 32.

Adaptive proportion Test The goal of the adaptive proportion test is to detect a large loss of entropy, such as might occur as a result of some physical failure or environmental change affecting the noise source. The number of consecutive 0s or 1s must be less than or equal to the cutoff value C_1 for random numbers separated by 1024 bits [14].

3.2.2 SP800-90B restart test

In the restart test, the random number sequence immediately after startup has the same distribution as others. The random number should be independent of bit position, and hard to predict the subsequent startup output. These characteristics prevent an attacker from predicting the following sequence. At first, the sanity test is performed, and if it is passed, the random number entropy is evaluated

First, run SanityTest. We start up the TRNG 1000 times and measure 1000 bits for each startup. Using this measurement data, create a matrix with 1000 rows and 1000 columns. The row data set consists of 1M bits by connecting each row, and the column data set consists of 1M by concatenating each column. Using this information, we perform the SanityCheck test. SanityChek is a test using a binomial test. In the case of binary sequences, the test is to find the frequency of 0 or 1. And check if the frequency is greater than that expected by the initial entropy HI (the entropy output from the evaluation of the IID). The error probability α for Type I is set to 0.01 for the entire SanityCheck. A test is performed for every 1000 bits of the row and column datasets, and if the test is not passed, the SanityTest fails. If any test is not passed, the SanityTest fails, and the restart test also fails.

Next, the entropy is evaluated. The entropy H_r and H_c of the row and column datasets are calculated, respectively. If either Hr or Hc is less than half of the initial entropy, it fails; otherwise, the smallest of HI, H_r, and H_c is the entropy of this source.

3.2.3 Pattern Counting Health Test

The Pattern Counting Health Test is a test implemented in the Intel CPU [16] and is used in place of the above two tests of SP800-90B. The test is performed every 256 bits. The six data patterns (1, 01, 101, 010, 0110, 1001) are counted in the sliding window. Type I errors (false-positive errors) are set as 1 %. In our evaluation, the test condition was the same as it in the [16].

3.2.4 Autocorrelation Coefficient and Frequency Test

A frequency test and an autocorrelation coefficient are evaluated as examples of evaluating a random number series by a health circuit.

Perform a χ^2 test on the bit frequency. Let the error probability of type I be $\alpha = 10^{-3}$; the number of 1s in the 256-bit series should be from 103 to 159. However, we use from 96 to 159 referring [16].

For the autocorrelation function, we investigated 256-bit Serial Correlation Coefficient (SCC). SCC is the correlation coefficient of Lag-1.

Considering that the bit width is short, and the fluctuation of the SCC value becomes large, min-entropy evaluated it with a pass of 0.6 or more [16].

4 TRNG Performance Evaluation

4.1 Evaluation Environment

Figure 6 shows the evaluation environment. TRNG is implemented on the Diligent Z7 Zynq-7010 evaluation board [19]. The Zynq-7010 is equipped with the XC7Z010-1CLG400C and the dual-core ARM Cortex-A9. The FPGA development environment is Xilinx Vivado 2019.1 [20].

A command is set to command register from the PC via UART to control the TRNG core. When measuring, random numbers generated by the TRNG core are written to RAM and sent to thse PC via UART. RAM size is $4K \times 32$ bit. Therefore, this operation is repeated forty times to obtain a random sequence of 5 Mbits in total. The clock to TRNG is generated by dividing the system clock. The random sequences start collecting immediately after enabling signal (EN) to the TRNG is ON for startup tests.

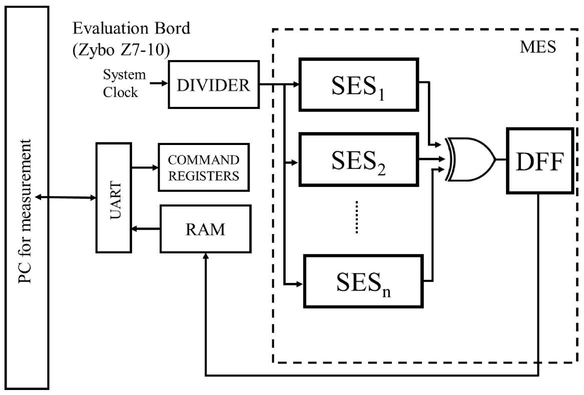

Figure 6: Evaluation bord block diagram

4.2 Throughput

The throughput of TERO-based, COSO-based, and STR-based TRNG are 2 Mbit/s, 12.5 Mbit/s, and 200 Mbit/s, respectively.

4.3 Statistical test at stable state

The stable state evaluation results of TERO-based, COSO-based, and STR-based TRNG are shown in Table 1, Table 2, and Table 3, respectively. The evaluation used the statistical tests of AIS 20/31 and the IID test of SP800-90B. We executed the tests ten times for each of the five boards and used 5M bits for each test. All TRNGs passed the test. The entropy of SP 800-90B in the tables is the initial entropy estimation, and the restart test calculates the entropy estimation [14].

Table 4 shows the results of the SP800-22 tests for TERO-based, COSO-based and STR-based TRNG. P-value and proportion is shown. The minimum pass rate for each statistical test 1052 for a sample size = 1073 binary sequences except the random excursion test and for the random excursion test is 645 for a sample size = 660 binary sequences. If the p-value obtained for each test is less than 0.0001, we conclude that the bit data is not random. The asterisk denotes that a test consisted of several sub-tests and that the minimum p value is shown. TERO-based, COSO-based passed all tests. STR-based TRNG passed tests except for one out of 148 non-overlapping template matching. The values of the failed test are shown in bold type.

4.4 Statistical test at startup

4.4.1 SP800-90B Health Test

100 sets of random numbers were tested immediately after the TRNG started. The health test passes when both the reputation count test and adaptive count test pass. In this evaluation, TERO-based,

Table 1: TERO-based TRNG evaluation result at stable state

Board number	AIS20/31		SP800-90B	
	#pass	entropy	#pass	entropy
1	10/10	0.999	10/10	0.995
2	10/10	0.999	10/10	0.998
3	10/10	0.999	10/10	0.998
4	10/10	0.999	10/10	0.998
5	10/10	0.999	10/10	0.998

Table 2: COSO-based TRNG evaluation result at stable state

Board number	AIS20/31		SP800-90B	
	#pass	entropy	#pass	entropy
1	10/10	0.999	10/10	0.995
2	10/10	0.999	8/10	0.994
3	10/10	0.999	10/10	0.994
4	10/10	0.999	9/10	0.993
5	10/10	0.999	10/10	0.992

Table 3: STR-based TRNG evaluation result at stable state

Board number	AIS20/31		SP800-90B	
	#pass	entropy	#pass	entropy
1	10/10	0.999	10/10	0.987
2	10/10	0.999	9/10	0.996
3	10/10	0.999	10/10	0.998
4	10/10	0.999	10/10	0.998
5	10/10	0.999	9/10	0.994

Table 4: Reult of SP800-22 test

Test name	TERO		COSO		STR	
	P-value	Proportion	P-value	Proportion	P-value	Proportion
Frequency	0.784731	1065/1073	0.294073	1062/1073	0.074035	1056/1073
BlockFrequency	0.674040	1064/1073	0.406814	1063/1073	0.298120	1055/1073
CumulativeSums ∗	0.222415	1063/1073	0.123997	1062/1073	0.011614	1059/1073
Runs	0.958070	1063/1073	0.945742	1062/1073	0.795339	1064/1073
LongestRun	0.901946	1056/1073	0.757550	1061/1073	0.483286	1059/1073
Rank	0.294073	1060/1073	1062/1073	0.812633	0.051725	1063/1073
FFT	0.292733	1053/1073	0.565500	1062/1073	0.124671	1061/1073
NonOverlappingTemplate ∗	0.000200	1053/1073	0.036610	1059/1073	0.689501	**1051/1073**
OverlappingTemplate	0.093535	1058/1073	0.292733	1063/1073	0.388813	1066/1073
Universal	0.961257	1066/1073	0.028034	1064/1073	0.577019	1056/1073
ApproximateEntropy	0.915665	1058/1073	0.619610	1063/1073	0.895373	1063/1073
RandomExcursions ∗	0.106093	652/660	0.004888	663/667	0.132419	619/625
RandomExcursionsVariant∗	0.008689	653/660	0.012854	658/667	0.047468	618/625
Serial ∗	0.159252	1062/1073	0.559762	1061/1073	0.520083	1060/1073
LinearComplexity	0.840776	1065/1073	0.555945	1067/1073	0.842378	1061/1073

Table 5: Result of SP800-90B Health test

Board number	TERO	COSO	STR
1	100/100	100/100	100/100
2	100/100	100/100	100/100
3	100/100	100/100	100/100
4	100/100	100/100	100/100
5	100/100	100/100	100/100

Table 6: Results of restart test

	TERO	COSO	STR
Board number	entropy	entropy	entropy
1	0.994	0.994	0.987
2	0.994	0.993	0.995
3	0.995	0.993	0.995
4	0.995	0.993	0.993
5	0.995	0.990	0.994

Table 7: Results of pattern counting test (Failures per 800 tests)

Board number	TERO	COSO	STR
1	1	0	0
2	1	0	0
3	1	1	0
4	1	2	0
5	1	0	0

Table 8: Results of autocorrelation coefficient and frequency test (Failures per 8000 tests)

	TERO		COSO		STR	
Board number	ACC	freq	ACC	freq	ACC	freq
1	0	0	0	0	2	0
2	0	0	0	0	0	0
3	0	0	0	3	0	0
4	0	0	0	0	2	0
5	0	1	0	0	2	0

COSO-based, and STR-based TRNG were evaluated with $\alpha = 2^{-20}, H = 0.997$, $C = 21$, and $C_1 = 589$. The results are shown in Table 5. All data of TERO-based, COSO-based, and STR-based TRNG have passed the health test from the startup.

4.4.2 SP800-90B restart test

The restart test used the first 1000 bits from 2048 bits immediately after the TRNG started. TERO-based, COSO-based, and STR-based TRNG passed the test. Table 6 shows the entropy estimation of the first 1000 bits. The average entropies of these are 0.997, 0.993, and 0.993, respectively. It can be seen that TERO-based, COSO-based, and STR-based TRNG have high entropy.

4.4.3 Pattern counting health test

100 sets of 2048-bit data were used immediately after the TRNG started. The evaluation results are shown in Table 7. It can be seen that there are almost no failures in all of the TERO-based, COSO-based, and STR-based TRNG.

4.4.4 Autocorrelation Coefficient and Frequency Test

The autocorrelation coefficient and frequency test use 1000 sets of 2048-bit data immediately after the TRNG started on five boards. Table 8 shows the evaluation results. It can be seen that TERO-type, COSO-type, and STR-type TRNG have almost no failure numbers in both the autocorrelation coefficient and the frequency test.

5 Consideration

5.1 Comparison of feasibility

Table 9 shows the evaluation results of the TRNG type, method, circuit scale, power consumption, bit rate, efficiency, entropy, AIS 20/31 statistical test, SP800-90B IID test, startup test, and feasibility. For comparison, it includes the Latch-based TRNG [21, 22], ERO-based TRNG, MURO-based TRNG [23] and implementations by the Petura et al.[10]. The power consumption in our implementation is an evaluation in the development environment, not an actual measurement value.

The evaluation of the feasibility, that is, the implementation difficulty, follows [10]. It is divided into six scores. Score 5 is a configuration that can be easily implemented on different models. Score 4 can be implemented with a simple setup if it is the same model. Score 3 requires parameter optimization and settings that are difficult for some models. Score 2 requires complicated settings (wiring optimization) depending on the model. The same model can be configured as is if the settings are made. Score 1 is the configurations that must be optimized by each device, even for the same model. Score 0 is that random numbers may not be generated even after optimization. Since we deal with only one model of the Zynq-7010 board, the highest score is 4.

Table 9: Summary of TRNG implementation and evaluation

TRNG type	Method	Area LUT/Reg	Power cons. mW	Bit rate Mbit/s	Efficiency bits/μWs	Entropy per bit	AIS20/31	SP800-90B IID	Startup	feasibility
TERO	This paper: MES (8 SES)	93/16	4	2	500	0.999	○	○	○	4
	Petura[10]	39/12	3.312	0.625	188.7	0.999	○	-	-	1
COSO	This paper: MES (48 SES)	162/295	2	12.5	6250	0.999	○	○	○	4
	Petura	18/3	1.22	0.54	442.6	0.999	○	-	-	1
STR	This paper: MES (2 SES)	126/153	27	200	7407	0.999	○	○	○	4
	Petura	346/256	65.9	154	2343.2	0.999	○	-	-	2
Latch	Fujieda[21]: MES (320 SES)	716/974	-	20.0	-	-	-	-	-	4
	Torii[22]: MES (256 SES)	660/768	3	15.6	520	0.999	○	○	○	4
ERO	Petura	46/19	2.16	0.0042	1.94	0.999	○	-	-	5
	Hayashi[23]	12/19	1	0.125	125	0.999	○	○	○	4
MURO	Petura: MES (120 SES)	521/131	54.72	2.57	46.9	0.999	○	-	-	4
	Hayashi: MES (2 SES)	20/21	1	0.125	125	0.999	○	○	○	4

5.1.1 TRNG configuration considering the small scale and low power consumption

When a TRNG is implemented in IoT devices, it is desirable to have a small scale, low power consumption, and high entropy. Furthermore, the high throughput would be desirable. From this point of view, it is better to implement the COSO-based TRNG as a single entropy source. However, it is Score 1 and requires optimization of each device's layout and wiring, which makes it challenging to implement. The next candidate is implementing TERO-based TRNG as a single entropy source. However, it is Score 1 and hard to implement as COSO-based TRNG.

From the aspect of the implementing difficulty, it is desirable to implement ERO-type TRNG. The difficulty for implementation is score 4, so it can implement without considering device differences. In addition, both the circuit scale and power consumption of the ERO-based TRNG are low. However, the throughput of ERO-based TRNG is very low. For applications that require throughput, the MES implementations of TERO-based or STR-based TRNG are a good candidate in terms of throughput and efficiency despite the circuit scale and power consumption will be relatively large.

5.1.2 TRNG configuration considering throughput

Considering throughput, our MES of STR-based TRNG using two SES with $L = 63$ is good. It is Score 4 and can be implemented without being aware of individual device differences in layout and routing. In addition, the circuit scale and power consumption are kept lower than STR-based TRNGs of Petura et al. because the configuration of the SES is different between them. The SES of Petura et al. has 255 stages ($L = 255$), while our SES constituting the MES is 63 ($L = 63$).

5.2 TRNG Characteristics

5.2.1 Number of SES and entropy

To evaluate whether the random numbers generated by TRNG have high entropy, we decided the condition that the statistical test is performed multiple times and 90 % or more of the tests are passed. We found n_{ses} that satisfies the condition for three types of TRNG.

For example, in COSO-based TRNG, Figure 7 showed the pass rate for AIS 20/31 and SP800-90B when n_{ses} is from 8 to 48 in increments of 8. The results from measuring and evaluating 50 random numbers, ten times each, on five FPGA boards. From this result, $n_{ses} \geq 24$ is enough to meet the condition. In this paper, we choose $n_{ses} = 48$ for the COSO-based TRNG because there needs a sufficient margin to clear the condition on different FPGA models.

In TERO-based TRNG, Figure 8 showed the pass rate for AIS 20/31 and SP800-90B when n_{ses} is from 2 to 8 in increments of 2. The results from measuring and evaluating 50 random numbers, ten times each, on five FPGA boards. From this result, $n_{ses} \geq 6$ is enough to meet the condition. In this paper, we choose $n_{ses} = 8$ for the TERO-based TRNG because there needs a sufficient margin to clear the condition on different FPGA models.

It is expected that n_{ses} will change if the experimental platform changes, due to different electrical characteristics and wiring capacitance. In this paper, n_{ses} is selected with enough margin, however evaluation n_{ses} depending on the platform is a future issue. On the other hand, we consider that the MES configuration TRNG generates high-entropy random numbers because the three SES in this paper are known to generate high-entropy random numbers on three types of FPGA; Spartan, Cyclone, and Smart Fusion 2 [10].

5.2.2 Parameters of STR-based TRNG

Figure 9 shows the results of the statistical evaluation for the number of stages (L) and the number of SES (n_{ses}) in the MES implementation of STR-based TRNG. We evaluated STR-based TRNG when L= 31, 63, 123 and 255 and n_{ses} is one and two. In the evaluation, we measured 5 Mbit data ten times for each of the five boards. We also measured at startup state on the same five boards, and the results are the same as the stable state.

As shown in Figure 9, the SES implementation of the STR-based TRNG is difficult. That is, it does not pass the condition that 90 % or more of the statistical tests even when the number of

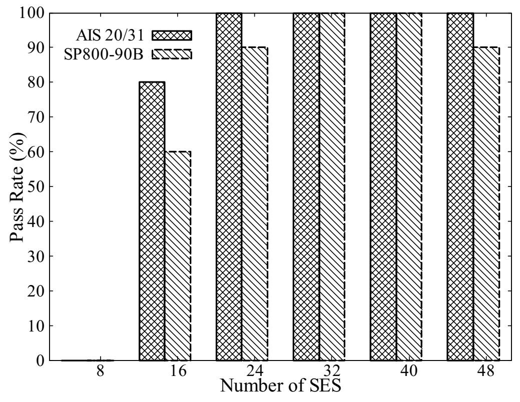

Figure 7: Number of SES evaluations for COSO-based TRNG

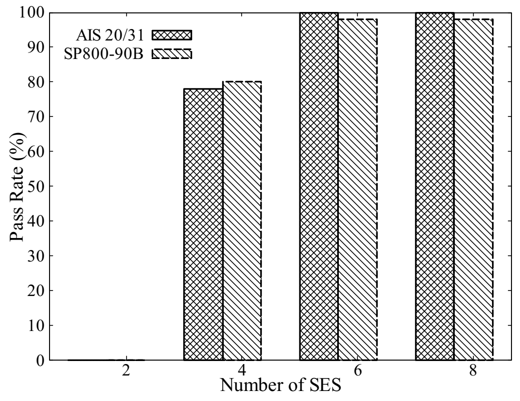

Figure 8: Number of SES evaluations for TERO-based TRNG

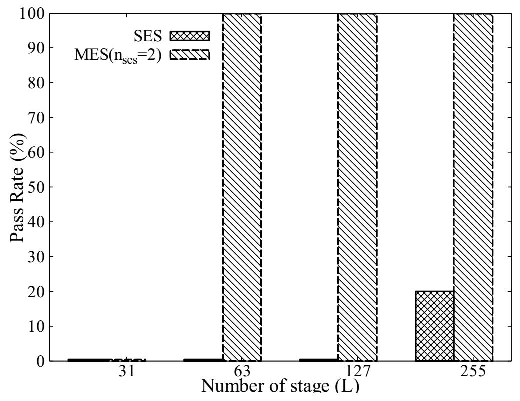

Figure 9: Number of stages (L) and SES evaluations for STR-based TRNG

stage L=255. On the other hand, the MES implementation (n_{ses}=2) passes the statistical test when $L \geq 63$. Therefore, we select L =63 in this paper.

5.2.3 Entropy

Table 9 shows the entropy evaluated by AIS 20/31 for TERO-based, COSO-based, and STR type TRNG because the previous TRNGs evaluated by it. The entropy evaluated by SP800-90B is shown in Table 6.

5.3 Clock generation

In this evaluation, we use the system clock to generate a random number. Using the system clock, generated random numbers may be affected by the global noise jitter of the entire board. Therefore, to increase the attack resistance of TRNG, it is desirable to use the output of RO as a clock [24]. In the future, we plan to evaluate TRNG using the clock signal generated by RO.

5.4 Clock synchronization

In the MES configuration, we must consider synchronization between SES and D-FF clock because set-up time violation may happen by sampling XORed SES outputs by D-FF. For TERO- and STR-based TRNGs, the TRNG clock is the system clock, and in MES configuration, the output DFF clock is the system clock. Therefore, there is little chance of a set-up violation. However, for the COSO-based TRNG, each SES in MES uses a different clock signal, and we need a circuit to synchronize the clock with the system clock of the output DFF, although it decreases the throughput of TRNG. The evaluation in Table 9 includes this circuit for COSO-based TRNG.

5.5 Temperature Characteristics

For TERO-based and STR-based TRNGs, temperature evaluations in the SES configuration are reported [25, 26]. They say that the frequency of RO varies when the temperature changes. However, these TRNGs pass the statistical tests. For COSO-based TRNGs, the report in [27] states that the two ROs cancel out the frequency changes when the temperature changes. Therefore, COSO-based TRNG is resistant to temperature changes. We consider the TRNG in the MES configuration generates high-entropy random numbers when the temperature changes because the three SES in this paper is robust against temperature changes.

5.6 Possibility of attack

Ring oscillator-based TRNGs are known to have reduced entropy due to injection attacks [28]. Also, according to [16], it is reported that when multiple oscillators are used, they are susceptible to injection attacks because there are multiple modes in which the oscillators lock each other. MES configuration uses multiple SES. Therefore, it may be vulnerable to injection attacks. In addition, the attack on TERO-based TRNG [29] and the attack on STR-based TRNG [25] have not been evaluated for MES-based. Evaluation of resistance to these attacks is future work.

6 Conclusion

We evaluated the MES-type TRNGs for TERO-based, COSO-based and STR-based TRNG, and implemented them on FPGA using the Zybo Z7 Zynq-7010 evaluation board. We performed random number evaluation by statistical tests following SP800-90B, SP800-22 and AIS 20/31, and some tests at start-up. Based on these evaluations, we proposed the optimum configuration for MES-type TRNG. Our evaluated TRNGs generate high entropy random numbers when we implement TRNGs with eight single entropy sources for TERO-based TRNG, 48 for COSO-based TRNG, and two for STR-based TRNG, respectively. In addition, the difficulty of implementation can be reduced by implementing MES configuration. We compared the previously proposed TRNGs with our evaluated TRNGs. From the viewpoint of embedding TRNG for IoT devices, our evaluated TRNGs are helpful in some applications.

Various attack methods have been proposed against TRNGs. However, evaluating the resistance of MES-type TRNG is future work.

Acknowledgment

This work was partially supported by JSPS KAKENHI Grant Number JP19K11973.

References

[1] A. Menezes, P. van Oorschot, and S. Vanstone. Handbook of applied cryptography. *Boca Raston, FL CRC FL*, 1977.

[2] W. Holman, J. Connelly, and A. Dowlatabadi. An integrated analog digital random noise source. *IEEE Transactions on Circuits and Systems I Fundamental Theory and Applications*, 44(6):521–528, 1997.

[3] W.J. Martin B.Sunar and D.R.Stinson. A provably secure true random number generator with built-in tolerance to active attacks. *IEEE Transactions on Computers*, 56(1):109–119, 2007.

[4] M. Bellido. Simple binary random number generator. *Electronics Letters*, 28(7):617–618, 1992.

[5] M. Varchola and M. Drutarovsky. New high entropy element for fpga based true random number generators. In *Cryptographic Hardware and Embedded Systems, CHES 2010, LNCS*, volume 6225, pages 351–365. Springer-Verlag, 2010.

[6] P. Kohlbrenner and K. Gaj. An embedded true random number generator for fpgas. *in Proceedings of the 2004 ACM/SIGDA 12th international symposium on Field programmable gate arrays. ACM*, pages 71–78, 2004.

[7] H. Hata and S. Ichikawa. Fpga implementation of metastability based true random number generato. *IEICE Transaction on Information and Systems*, E95-D(2):426–436, 2012.

[8] A. Cherkaoui, V. Fischer, A. Aubert, and L. Fesquet. A self-timed ring based true random number generator. In *2013 IEEE 19th International Symposium on Asynchronous Circuits and Systems*, pages 99–106, 2013.

[9] A. Cherkaoui, V. Fischer, L. Fesque, and A. Aubert. A very high speed true random number generator with entropy assessment. In *Cryptographic Hardware and Embedded System, CHES 2013, LNCS*, volume 8086, pages 179–196. Springer, 2013.

[10] O. Petura, U. Mureddu, N. Bochard, V. Fischer, and L. Bossuet. A survey of ais-20/31 compliant trng cores suitable for fpga devices. In *2016 26th International Conference on Field Programmable Logic and Applications (FPL)*, pages 1–10, 2016.

[11] W. Killmann and W. Schindler. Ais20/ais31. a proposal for: Functionality classes for random number generators, version 2.0. 2011.

[12] N. Fujieda. On the feasibiliy of tero-based true random number generator on xilinx fpgas. *30th International Conference on Field Programmable Logic and Applications*, pages 103–108, 2020.

[13] V. Rozic A. Peetermans and I. Verbauwhede. A highly-portable true random number generator based on coherent sampling. In *2019 29th International Conference on Field Programmable Logic and Applications (FPL)*, pages 218–224, 2019.

[14] M. S. Turan, E. Barker, J. Kelsey, K. A McKay, M. L Baish, M. Boyle, et al. Recommendation for the entropy sources used for random bit generation. *NIST Special Publication*, 800(90B):102, 2018.

[15] L. E. Bassham, A. L. Rukhin, J. Soto, J. R. Nechvatal, M. E. Smid, E. B. Barker, S. D. Leigh, M. Levenson, M. Vangel, D. L. Banks, N. A. Heckert, J. F. Dray, and S. Vo. Sp 800-22 rev. 1a. a statistical test suite for random and pseudorandom number generators for cryptographic applications. Technical report, Gaithersburg, MD, USA, 2010.

[16] D.Johnston. *RANDOM NUMBER GENERATORS - PRICIPLES AND PRACTICES A guide for Engineers and Programmers*. DeG Press, 2018.

[17] C. Celi. Sp800-90b_entropyassessment,. https://github.com/usnistgov/SP800-90B_EntropyAssessment, 2018.

[18] NIST. Nist statistical test suite. https://csrc.nist.gov/CSRC/media/Projects/Random-Bit-Generation/documents/sts-2_1_2.zip, 2014.

[19] Diligent. Zybo z7 reference manual. https://digilent.com/reference/programmable-logic/zybo-z7/reference-manual, 2018.

[20] Xilinx. Vivado. https://www.xilinx.com/products/design-tools/vivado.html, 2022.

[21] N. Fujieda and S. Ichikawa. A latch-latch composition of metastability-based true random number generator for xilinx fpgas. *IEICE Electronics Express*, 15(10):20180386–20180386, 2018.

[22] N. Torii and H. Omae. Performance evaluation of true random number generator using latches. *IEICE Technical Report HWS0*, pages HWS2020–17–2020, 2020.

[23] K. Hayashi and N. Torii. Implementation and evaluation of a ring oscillator type true random number generator. volume D2-3, 2021.

[24] V. Fischer, F. Bernard, N. Bochard, and M. Varchola. Enhancing security of ring oscillator-based trng implemented in fpga. In *2008 International Conference on Field Programmable Logic and Applications*, pages 245–250, 2008.

[25] M. Honorio, K. Thomas, M. Enrique San, and H. Michael. Fault attacks on strngs: Impact of glitches, temperature, and underpowering on randomness. *IEEE Transactions on Information Forensics and Security*, 10(2):266–277, 2015.

[26] O. Petura. *True random number generators for cryptography: Design, securing and evaluation.* PhD thesis, Université de Lyon, 2019.

[27] V. Fischer. D2.1- report on selected trng and puf principles. executive summary. random number generators (rngs) and physical unclonable functions (pufs). Technical Report ICT-644052-D2.1, European Union's Horizon 2020 research and innovation programme, 2016.

[28] T.Markettos and S.W.Moore. The frequency injection attack on ring-oscillator-based true random number generators. *Cryptographic Hardware and Embedded Systems, CHES 2009, LNCS*, 5747:317–331, 2009.

[29] D. Fujimoto S. Osuka and Y. Hayashi. Fundamental study on an estimation method of output bits by observing em leakage from tero-based trng. *The 37th Symposium on Cryptography and Information Security (SCIS2020)*, 3E3-4, 2020.