A Highly Flexible Lightweight and High Speed True Random Number Generator on FPGA

Mei, F., Zhang, L., Gu, C., Cao, Y., Wang, C., & Liu, W. (2018). A Highly Flexible Lightweight and High Speed True Random Number Generator on FPGA. In *IEEE Computer Society Annual Symposium on VLSI* Institute of Electrical and Electronics Engineers Inc.. Advance online publication. https://doi.org/10.1109/ISVLSI.2018.00079

Published in:
IEEE Computer Society Annual Symposium on VLSI

Document Version:
Peer reviewed version

Queen's University Belfast - Research Portal:
Link to publication record in Queen's University Belfast Research Portal

Publisher rights
© 2018 IEEE.
This work is made available online in accordance with the publisher's policies. Please refer to any applicable terms of use of the publisher.

General rights
Copyright for the publications made accessible via the Queen's University Belfast Research Portal is retained by the author(s) and / or other copyright owners and it is a condition of accessing these publications that users recognise and abide by the legal requirements associated with these rights.

Take down policy
The Research Portal is Queen's institutional repository that provides access to Queen's research output. Every effort has been made to ensure that content in the Research Portal does not infringe any person's rights, or applicable UK laws. If you discover content in the Research Portal that you believe breaches copyright or violates any law, please contact openaccess@qub.ac.uk.

Open Access
This research has been made openly available by Queen's academics and its Open Research team. We would love to hear how access to this research benefits you. – Share your feedback with us: http://go.qub.ac.uk/oa-feedback

Download date:20. Apr. 2024

A Highly Flexible Lightweight and High Speed True Random Number Generator on FPGA

Faqiang Mei[1], Lei Zhang[1], Chongyan Gu[2], Yuan Cao[3], Chenghua Wang[1] and Weiqiang Liu[1*]
[1]*College of EIE, Nanjing University of Aeronautics and Astronautics, Nanjing, China*
[2]*CSIT, ECIT, Queen's University Belfast, Belfast, UK*
[3]*College of IoT Engineering, Hohai University, Changzhou, China*
E-mail: {meifaqiang, zhanglei_1993, chwang, liuweiqiang}@nuaa.edu.cn, cgu01@qub.ac.uk, caoyuan0908@gmail.com

***Abstract*—True random number generator (TRNG), plays an important role in information security systems. Conventional TRNGs use natural physical stochastic processes including thermal noise, chaos-based circuit and so on to generate the random numbers. These analog circuit based TRNG structures often consume lots of hardware resources, and are not easy to be integrated in digital systems. In this paper, a low-cost and high-speed TRNG has been proposed by using mixed oscillation generated from XOR gates nested multiple ring oscillators (ROs). Multi-group mixed oscillation XOR operation is applied to obtain high-speed output. The proposed TRNG design is implemented on Xilinx Artix-7 XC7A35T-1FTG256C FPGA. It achieves a high performance with throughput up to 160 Mbps and with a usage of 37 FFs and 25 look up tables (LUTs) in the FPGA. The results show that the proposed TRNG design has successfully passed the testing standards of NIST SP800-22 and AIS31. Compared with previous designs, the proposed TRNG design achieves lower hardware resource consumption and higher speed.**

***Keywords*-TRNG; Mixed Oscillation; FPGA;**

I. INTRODUCTION

The information security has extensively influenced modern communication and computing systems. The random number plays an important role in cryptography and it is used in almost all security protocols and cryptographic algorithms [1]. The security of the whole system relies on the efficiency and quality of the random number sequences. Therefore, high speed and high quality are two essential requirements of random numbers in security system.

Both pseudo random number generator (PRNG) and true random number generator (TRNG) are usually used to generate random sequences that are used in practical applications. In critical security applications, the random numbers should be truly unpredictable and random. PRNG is generally based on a seed, through a certain mathematical algorithm to generate random sequences. The pseudo random number is predictable when the seed is revealed, and the security of the whole system will be vulnerable to attacks. Compared with PRNGs, TRNG is desirable in terms of security level [2] as it can produce unpredictable random number sequences that utilize various random differences in the physical process. For a TRNG, it is difficult to predict the random sequences even if the attacker has unlimited computing power and collects a large number of random sequences. Therefore, TRNG has been a highly demanded security primitive.

Conventional TRNGs employ natural physical stochastic processes such as resistance thermal noise and chaos to generate random sources. Although the statistical distribution of these entropy sources is ideal, they are mainly analog circuit based TRNG structures that often consume a lot of hardware resources. Moreover, they are difficult to be integrated in digital systems.

Due to the ubiquitous nature of the IoT, lightweight and high speed digital TRNG design are required for low cost devices. FPGA based TRNGs have been studied quite extensively [3]. To address the above limitations of the conventional TRNGs, a novel high-speed TRNG design is proposed and implemented on FPGA in this work. It also has significant advantages in improving the speed and reducing the hardware cost. The proposed TRNG, which successfully passed two commonly used testing standards of AIS31 and NIST SP800-22, achieves high reliability and demonstrates feasibility for practical applications.

This paper is organized as follows. Section II reviews the existing digital TRNGs. Section III presents the proposed TRNG design and its analysis. Section IV provides the testing results. Hardware resource analysis and performance comparison with previous designs are given in Section V. Section VI concludes this paper.

II. RELATED WORKS

There are two main approaches to construct true random sources using digital circuits: one is oscillator sampling [4] and the other is metastability [5].

The oscillator sampling method [4] has been proposed to sample a high frequency oscillator by utilizing a low frequency oscillator and the sampling result was used as the output random data. It utilizes the phase noise of the oscillator as a random source. The output rate is determined by the low frequency oscillation. A TRNG circuit proposed by [6], as shown in Fig. 1, combines several oscillation signals with an XOR gate to exploit randomness of phase jitter. [3]

presented an enhanced structure by adding an extra D flip-flop after each ring to improve the overall randomness for the output. The enhanced TRNG, implemented on an Altera Cyclone II FPGA, passed NIST and Diehard statistical tests at a throughput of 100 Mbps. However, this design uses 167 LEs (logic element) to implement 50 ring oscillators (ROs), which is quite expensive in terms of hardware cost.

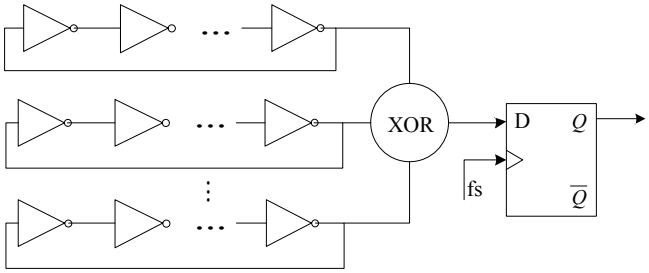

Fig. 1. The structure of digital TRNG proposed in [6].

Metastability-based structure is another typical approach for TRNG. It utilizes instability states that caused by the competition in logic gates, latches or flip flop (FF) triggers to produce an uncertain output. The final state is unable to predict since it depends on the electrical noise in the circuit. [7] proposed a RS latch-based TRNG and implemented on a Xilinx Virtex-4 FPGA. An open-loop TRNG design has been implemented on a Xilinx Virtex-5 FPGA in [8]. Two coarse delay chains were adopted on data and clock signals. It consumes a few hardware resources and generates random bits at a throughput of 20 Mbps. However, the issue for metastability-based TRNG is the low throughput, which requires very complex strategies of placement and routing to achieve balanced signal transmission paths.

III. Proposed TRNG Structure

To address these restrictions mentioned above, we propose a novel low cost and high speed TRNG based on digital circuit.

A. The Overall TRNG Architecutre

The proposed TRNG consists of three parts, a chained oscillation ring (COR), a FF array and an XOR array, as shown in Fig. 2. The overall TRNG architecture combines two random sources, the oscillating signal jitter and the metastable state of the FF. The oscillated ring accumulates jitter during the sampling time. The COR uses XOR gates to combine the jitter signal of the adjacent oscillation ring. By using the FF array, it produces a metastable state and guarantees that the sampling point include one or more oscillation regions. Finally, an XOR array is applied so that all the signals collected by FFs are mixed with XOR gates to produce high speed random sequences.

B. A Model for Jitter Sources in ROs

The random source of an ring oscillator (RO) is the jitter of gate transmission. Suppose $d_{i,j}$ is the delay of the *i*-th transmission gate in the *j*-th half clock. It can be obtained from [15]:

$$d_{i,j} = D_i + \Delta d_{i,j} = D_i + \Delta d_{Li,j} + \Delta d_{Gi,j} \qquad (1)$$

where, D_i is the constant delay of the *i*-th gate, which includes an interconnection delay between the *i*-th and the *(i+1)*-th gate, $\Delta d_{i,j}$ is the delay variation caused by the individual local delay $\Delta d_{Li,j}$ and common global delay $\Delta d_{Gi,j}$. The local jitter of the *i*-th gate during the half-period *j* can be expressed as

$$\Delta d_{Li,j} = \Delta d_{LGi,j} + \Delta d_{LDi,j} \qquad (2)$$

where, $\Delta d_{LGi,j}$ is the delay of Gaussian jitter from independent local sources, and $\Delta d_{LDi,j}$ is the local deterministic jitter. The global delay can be represented by

$$\Delta d_{Gi,j} = K_i(\Delta D + \Delta d_{GGj} + \Delta d_{GDj}) \qquad (3)$$

where, ΔD represents slow delay variations due to temperature and/or power supply deviations. Δd_{GGj} is the delay of the Gaussian noise from the power supply, and Δd_{GDj} is the delay of the global deterministic jitter from fast power supply variations. K_i is a coefficient defining the proportion of the global jitter on the jitter of gate *i*.

A strategy proposed in [15] simplifies the model and removes the signal that never contributes to the jitter. The simplified model is shown in Eq. (4).

$$d_{i,j} = D_i + \Delta d_{i,j} = D_i + \Delta d_{LGi,j} + K_i \Delta d_{GDj} \qquad (4)$$

C. The Contribution of the XOR Gates between the ROs

The common RO is composed of an odd number of inverters. An example of the RO consisting three inverters is shown in Fig. 3.

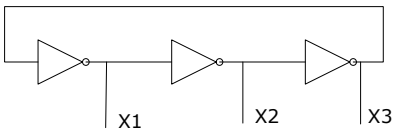

Fig. 3. A RO consisting of three inverters.

Suppose that the signal X_3 is '1' in the *j*-th half clock. When the signal reaches the position of the signal X_3 again, the time is:

$$t = D_1 + D_2 + D_3 + (\Delta d_{1,j+1} + \Delta d_{2,j+2} + \Delta d_{3,j+3}) \qquad (5)$$

When $\Delta d_{1,j+1} + \Delta d_{2,j+2} + \Delta d_{3,j+3}$ is equal to 0, the value of X_3 is determined by the propagation time ($D_1 + D_2 + D_3$) after oscillation, as shown in Fig. 4(a). When $\Delta d_{1,j+1} + \Delta d_{2,j+2} + \Delta d_{3,j+3}$ is not 0, the value of X_3 is dependent on two conditions as shown in Fig. 4(b). The

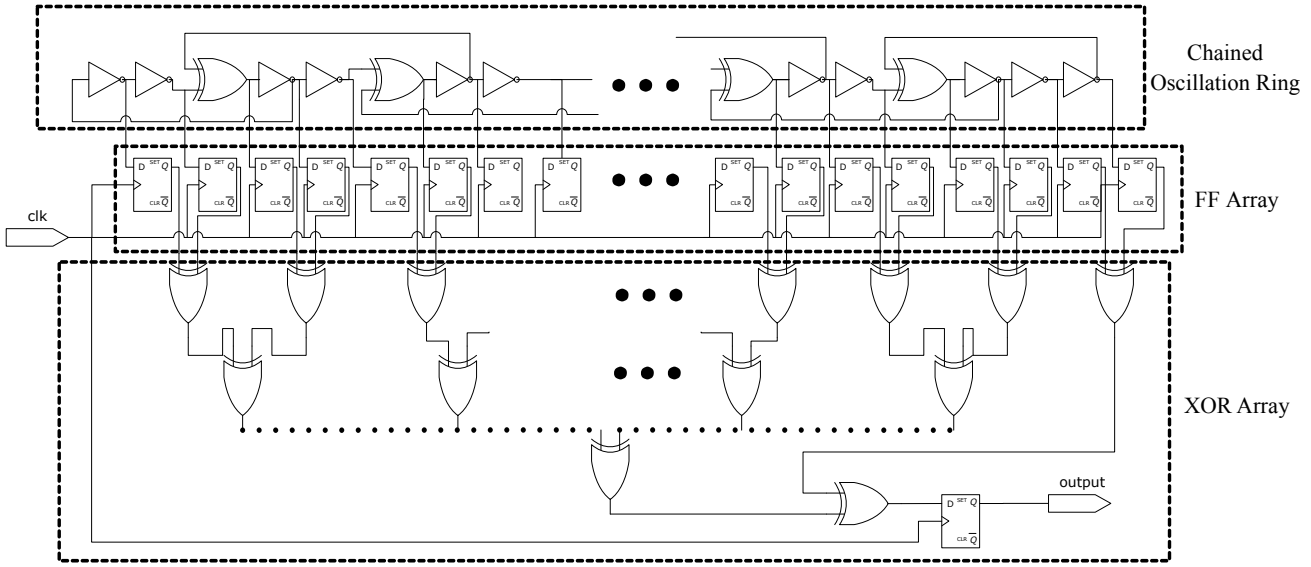

Fig. 2. The structure of proposed TRNG design.

shaded region represents the jitter range, where T_1 and T_2 represents two adjacent sampling points. If T_1 is known, T_2 can be easily obtained as shown in Fig. 4(a). If T_2 falls into the jitter range as shown in Fig. 4(b), the value of X_3 is determined by $\Delta d_{1,j+1} + \Delta d_{2,j+2} + \Delta d_{3,j+3}$. [15] has shown that increasing the time interval between T_1 and T_2 to accumulate more jitter can improve its randomness.

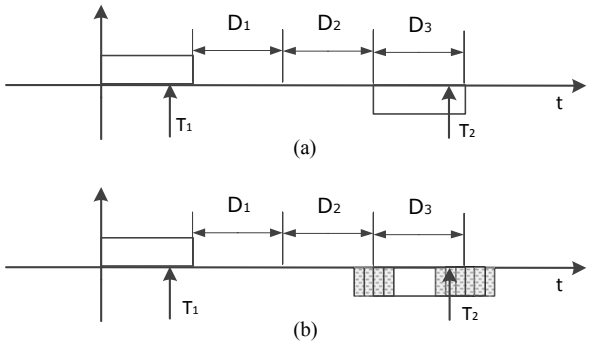

Fig. 4. The waveform of X_3 (a) when there is no jitter, and (b) when there is any jitter.

The entropy of random number can be increased by increasing the jitter range. This design adopts the COR, which can be used to connect the adjacent RO by XORs, to increase the jitter range. The COR in four oscillated rings is shown in Fig. 5.

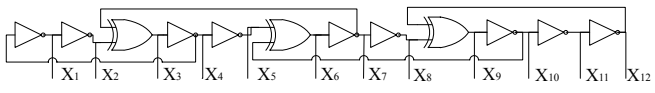

Fig. 5. The COR with 4 nested ROs.

Consistent with the above analysis, it is assumed that the signal is known in the *j*-th half clock.

$$X_4 = 0, X_5 = 1, X_{10} = 0. \tag{6}$$

After a clock cycle, the waveform of X_2 is shown in Fig. 6 (a). The waveform of X_7 is shown in Fig. 6(b). After the half-clock cycle, the waveform of the XOR is shown in Fig. 6(c). Due to the phase difference between X_4 and X_5, it can be seen from the Fig. 6(c) that the jitter regions in the two signals (Fig. 6(a) and Fig. 6(b)) are superimposed. Its jitter region is larger compared with one OR. So the time of cumulative jitter can be shortened by different or gate.

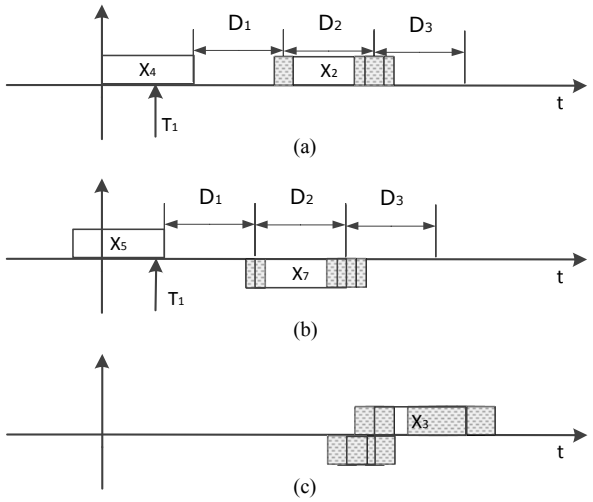

Fig. 6. The waveform of (a) X_2 after a clock cycle, (b) X_7 after a clock cycle, and (c) the XOR gate.

D. The Metastability of the TRNG

Assuming that the width of one edge collision is ε. As long as the time difference between the edge of the sampling clock and the edge of the input signal is less than ε, it can be considered that the edge collision occurs as shown in Fig. 7. The probability of the metastability at the *i*-th flip flop can be expressed as

$$P_m(i) = 2\varepsilon f \tag{7}$$

where, f represents the frequency of signal.

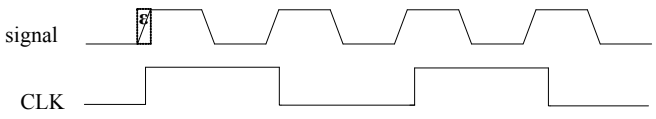

Fig. 7. The metastability state during sampling.

Assume that there is no metastability. The FF array is sampled under the control of the same sampling clock and there are *k* triggers. When sampling on the *i*-th trigger, the probability of getting a data of '1' is p_{oi1}. So the probability that the sample gets '0' is $1 - p_{oi1}$. According to the definition of information entropy, the entropy can be obtained:

$$H(i) = -p_{oi1} \times \log_2 p_{oi1} - (1 - p_{oi1}) \times \log_2(1 - p_{oi1}) \tag{8}$$

As metastability exists, it can assume that the probability of getting '1' is p_{oi1} on the *i*-th trigger when the metastability occur. Because of metastability, the probability of changing '1' to '0' is $p_{oi1} \times (1 - p_{mi1}) \times p_m(i)$ and the probability of changing '0' to '1' is $(1-p_{oi1}) \times p_{mi1} \times p_m(i)$. So the probability of sampling '1' can be expressed as

$$p_{i1} = p_{oi1} + (1-p_{oi1}) \times p_{mi1} \times p_m(i) - p_{oi1} \times (1-p_{mi1}) \times p_m(i) \tag{9}$$

The probability of sampling '0' is

$$p_{i0} = 1 - p_{oi1} + (p_{oi1} \times (1-p_{mi1}) - (1-p_{oi1}) \times p_{mi1}) \times p_m(i) \tag{10}$$

In order to simplify the model, it can be assumed that $p_{mi1} = \frac{1}{2}$. So p_{i1} and p_{i0} can be expressed as:

$$p_{i1} = p_{oi1} + (\frac{1}{2} - p_{oi1}) \times p_m(i) \tag{11}$$

$$p_{i0} = (1 - p_{oi1}) + (p_{oi1} - \frac{1}{2}) \times p_m(i) \tag{12}$$

The entropy is calculated according to p_{i1} and p_{i0}

$$H(i) = -p_{i1} \times \log_2 p_{i1} - p_{i0} \times \log_2 p_{i0} \tag{13}$$

When p_{i1} is closer to $\frac{1}{2}$, its entropy is closer to '1', and the randomness of the sampling is higher. Assume $\frac{1}{2} < p_{oi1}$, we have

$$(\frac{1}{2} - p_{oi1}) < (\frac{1}{2} - p_{oi1}) \times p_m(i) < 0 \tag{14}$$

$$\frac{1}{2} < p_{i1} < p_{oi1} \tag{15}$$

It can be seen from Eq. (15) that p_{i1} is closer to $\frac{1}{2}$ compared with p_{oi1}. Therefore, the metastability of FF array can increase the entropy of the random number.

Since each phase of the inverter has a different delay, the phase of the signal after each inverter is different. [15] has shown that the jitter obeys the normal distribution. Therefore phase difference distribution of the signal after each inverter can be considered as independent. Assuming that the expectation of every signal sampled from flip-flop is *u*. The expected value of XOR of all these bits is as follows [3]:

$$\mathrm{E} = \frac{1}{2} + (-2)^{K-1} \times (u - \frac{1}{2})^K = \frac{1}{2} \times (1 + (-2\varepsilon)^{\mathrm{K}}) \tag{16}$$

where, $\varepsilon = u - \frac{1}{2}$. Because $u \in (0, 1)$,

$$|2\varepsilon| < 1 \tag{17}$$

The larger the *K* is, *E* is closer to $\frac{1}{2}$. The probability of '1' and '0' are represented by P_1 and P_0, respectively.

$$E = 0 \times p_0 + 1 \times p_1 = p_1 \tag{18}$$

So

$$p_1 = \frac{1}{2} \times (1 + (-2\varepsilon)^{\mathrm{K}}) \tag{19}$$

The entropy of the TRNG can be expressed as

$$\begin{aligned} H(i) = \frac{1}{2} - \frac{1}{2} \times (1 + (-2\varepsilon)^{\mathrm{K}}) \times \log_2(1 + (-2\varepsilon)^{\mathrm{K}}) \\ -\frac{1}{2} \times (1 - (-2\varepsilon)^{\mathrm{K}}) \times \log_2(1 - (-2\varepsilon)^{\mathrm{K}}) \end{aligned} \tag{20}$$

As *K* increases, *E* approches to $\frac{1}{2}$, p_1 also approches to $\frac{1}{2}$ and entropy tends to be '1'. Therefore, larger *K* leads to larger entropy of the TRNG output.

E. The Characteristic of COR

The COR uses XOR gates nest with multiple ring oscillators (ROs) to achieve mixed oscillation signals. The schematic diagram of a simple COR is shown in Fig. 8. The COR can be activated when the number of nested ROs is equal to $3 \times N + 1$ ($N = 0, 1, 2...$). For example, the number of nested ROs, equal to 2, cannot oscillate and the intermediate signals are fixed to states 010101, as shown in Fig. 8.

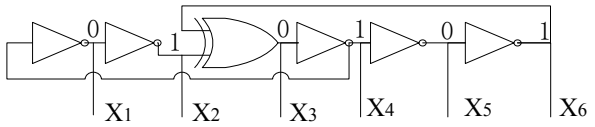

Fig. 8. An example of COR with two nested ROs.

When $N = 1$, the number of nested ROs is 4 and the COR can be oscillated as shown in Fig. 5. The processes of oscillation is as follows. Assume that the COR can be

settled down to a stable state. Then X_7 and X_8 must be '1' and '0', respectively, to activate the last RO. Therefore, X_{10} and X_6 are in a metastable state which results in X_7 changed accordingly. Finally, the COR remains oscillating.

IV. EXPERIMENT

To evaluate the effects of the FF array on the entropy of the proposed TRNG, the proposed TRNG is implemented on FPGA with and without the FF array. Then the proposed TRNG is tested with both NIST SP800-22 and AIS31 standards. A comparison with previous works is also provided in this section.

A. Verification on FPGA

To verify the random resources, we implement the proposed TRNG with and without the FF array, respectively, and compared the two designs. The results obtained at the sampling frequency of 160MHz are shown in the Table I after the NIST test. This test results confirm that the FF array can improve the entropy of the proposed TRNG.

Table I: NIST Test Results between TRNG with and without FF Array.

	without FF Array		with FF Array	
Test Item	P-value	Proportion	P-value	Proportion
Frequency	0.000000	0.00	0.474986	0.97
Block Frequency	0.000000	0.00	0.514124	0.99
Cumulative Sums	0.000000	0.00	0.616305	0.99
Runs	0.000000	0.00	0.924076	0.99
Longest Run of Ones	0.000000	0.44	0.955835	0.98
Rank	0.935716	0.99	0.494392	0.99
Discrete Fourier Transform	0.000406	0.92	0.971699	1
Nonperiodic Template Matchings*	0.001455	0.17	0.495490	0.99
Overlapping Template Matchings	0.000000	0.00	0.366918	0.99
Universal Excursions	0.000000	0.64	0.739918	0.98
Approximate Entropy	0.000000	0.00	0.319084	0.99
Random Excursion*	–	–	0.131175	0.99
Random Excursions Variant*	–	–	0.295078	0.99
Serial*	0.366918	1	0.243884	1
Linear Complexity	0.867692	0.99	0.102526	1

B. NIST SP800-22

The NIST SP800-22 [12] includes 16 tests. In order to ensure the reliability of the test results, we set the length of each test sequence as 10^6 bits. Then the number of data required for the tests is 3×10^8 bits. We collected 300 sets of random number from the proposed TRNG design under the clock frequency of 160MHz. The test results are shown in Table II. The proposed TRNG design passes the test when P-value is larger than 0.01 and the pass ratio is larger than 0.95. We can see that the random sequences that generated by the proposed TRNG design have passed all the NIST SP800-22 test items.

Table II: The Testing Result of NIST SP800-22.

Test Item	P-value	Proportion
Frequency	0.096578	0.98
Block Frequency	0.779188	1
Cumulative Sums	0.627038	0.985
Runs	0.350485	0.99
Longest Run of Ones	0.834308	0.99
Rank	0.202268	0.99
Discrete Fourier Transform	0.554420	0.99
Nonperiodic Template Matchings*	0.495490	0.99
Overlapping Template Matchings	0.574903	0.96
Universal Excursions	0.171867	0.99
Approximate Entropy	0.383827	0.99
Random Excursion*	0.295663	0.99
Random Excursions Variant*	0.295078	0.99
Serial*	0.515420	0.995
Linear Complexity	0.534146	0.99

C. AIS31

The AIS31 standard [13] includes two functional stages: P1 (T0∼T5) and P2 (T0∼T8). P1 is used to test the output of the post-processing part of the TRNG, while P2 is used to test the output of the noise source.

Three data sets are collected under the clock frequency of 160MHz. The test results are shown in Table III, and the item without * in the table is the pass rate. It can be seen that all the data passed the test items included in AIS31.

Hence, the random sequences generated by the proposed TRNG design have passed two main test standards, *i.e.*, NIST SP800-22 and AIS31.

D. Analysis and Comparison

The proposed TRNG design is implemented on an Xilinx Aritx-7 FPGA. The hardware resource consumption is

Table III: The Testing Result of AIS31

Data	RS 1	RS 2	RS 3
disjointness test (T0)*	Pass	Pass	Pass
monobit tests (T1)	100	100	100
poker tests (T2)	100	100	100
run tests (T3)	100	100	100
long run test (T4)	100	100	100
autocorrelation test (T5)	100	100	100
uniform distribution test (T6)*	Pass	Pass	Pass
multinomial distributions (T7)*	Pass	Pass	Pass
entropy test (T8)*	Pass	Pass	Pass

shown in Table IV. The proposed design, only consumes 37 FFs and 25 LUTs, which uses approximate 0.1% of the overall resources. The throughput of the proposed digital TRNG is 160 Mbit/s. The proposed TRNG design is compared with conventional TRNGs in terms of speed, cost and source of randomness in Table V. The comparison of the hardware consumption is performed by converting the hardware consumption to the equivalent number of inverters and FFs. It can be seen that the proposed TRNG achieves higher speed and lower resource consumption than the previous works from [3][7][9][14].

Table IV: The Hardware Resource Consumption in FPGA.

Resource	Utilization	Available	Utilization %
FF	37	41600	0.09
LUT	25	20800	0.12
I/O	3	106	2.38
BUFG	1	32	3.12

Table V: The Comparison With Previous Designs.

Designs	[3]	[7]	[9]	[14]	This work
Speed(Mbps)	100	12.5	6.25	4	160
Resource (gate, FF)	(150, 50)	(256, -)	—	(128, 48)	(103, 37)
Source of Randomness	Gate Delay Instability	Metastability	PLL	self-timed rings	Gate Delay Instability& Metastability

V. Conclusion

In this paper, a new lightweight and high-speed TRNG is proposed by using the XOR gates nested multiple ROs to achieve mixed oscillation signals. Multi-group mixed oscillation signal with XOR operation is applied to obtain high-speed outputs. Furthermore, the proposed TRNG design exploits the metastablility of FFs as a random source to efficiently improve the randomness of output sequences. The proposed TRNG design is implemented on an Xilinx Artix-7 XC7A35T FPGA, which can achieve a throughout of 160 Mbit/s. The generated random number set has passed the commonly employed testing standards, *i.e.*, NIST SP800-22 and AIS31. The proposed TRNG achieves low-cost hardware resource usage, high speed performance and flexible implementation over conventional digital TRNGs.

References

[1] W. Schindler, Random Number Generators for Cryptographic Applications, Springer, 2009, pp. 5-23.

[2] B. Sunar, True Random Number Generators for Cryptography, Springer, 2009, pp. 55-73.

[3] K. Wold and C. H. Tan, "Analysis and enhancement of random number generator in FPGA based on oscillator rings," *Proc. Int. Conf. Reconfigurable Computing & FPGAs*, 2009, pp. 385-390.

[4] R. C. Fairfield, R. L. Mortenson, and K. B. Coulthart, "An LSI random number generator (RNG)," *Proc. CRYPTO*, 1985, pp. 203-230.

[5] D. J. Kinniment, and E. G. Chester, "Design of an on-chip random number generator using metastability," *Proc. European Solid-State Circuits Conf*, 2002, pp. 595-598.

[6] D. J. Kinniment and E. G. Chester, "A provably secure true random number generator with built-in tolerance to active attacks," *IEEE Trans. Computers*, vol. 56, pp. 109-119, 2006.

[7] H. Hata and S. Ichikawa, "FPGA implementation of metastability-based true random number generator," *IEICE Trans. Information & Systems,* vol. 95-D, pp. 426-436, 2012.

[8] F. Lozach, M. Ben-Romdhane and T. Graba, "FPGA design of an open-loop true random number generator," *Proc. Euromicro Conf. Digital System Design*, 2013, pp. 615-622.

[9] N. Deàk, T. Györfi, K. Màrton, L. Vacariu, and O. Cret, "Highly efficient true random number generator in FPGA devices using phase-locked loops," *Proc. Int. Conf. Control Systems & Computer Science*, pp. 453-458, 2015.

[10] D. Chen D, D. Singh and J. Chromczak, "A comprehensive approach to modeling, characterizing and optimizing for metastability in FPGAs," *Proc. ACM Int. Symp. FPGAs*, 2010, pp. 167-176.

[11] Viktor Fischer, A Closer Look at Security in Random Number Generators Design, *Constructive Side-Channel Analysis and Secure Design. Springer Berlin Heidelberg*, 2012, pp. 167-182.

[12] T. Lange, D. Lubicz and A. Weigl, "Random numbers generation and testing," *Handbook of Elliptic & Hyperelliptic Curve Cryptography*, pp. 715-735, 2005.

[13] W. Killmann, A proposal for: Functionality classes and evaluation methodology for true (physical) random number generators, Version 3.1, 2001, www.bsi.bund.de/zertifiz/zert/interpr/trngk31e.pdf.

[14] H. Martin, P. Peris-Lopez, JE. Tapiador and ES. Millan, "A new TRNG based on coherent sampling with self-timed rings," *IEEE Transactions on Industrial Informatics*, pp. 91-100, 2016.

[15] Fischer, V and Bernard, F and Bochard, N and Varchola, M, "Enhancing security of ring oscillator-based trng implemented in FPGA," *Field Programmable Logic and Applications*, 2008, pp. 245-250.