Review of Scientific Instruments ARTICLE scitation.org/journal/rsi

Efficient FPGA implementation of high-speed true random number generator

Cite as: Rev. Sci. Instrum. 92, 024706 (2021); doi: 10.1063/5.0035519
Submitted: 29 October 2020 • Accepted: 19 January 2021 •
Published Online: 10 February 2021

Zhenguo Lu, Shenshen Yang, Jianqiang Liu, Xuyang Wang, and Yongmin Li[a)]

AFFILIATIONS

State Key Laboratory of Quantum Optics and Quantum Optics Devices, Institute of Opto-Electronics, Shanxi University, Taiyuan 030006, China and Collaborative Innovation Center of Extreme Optics, Shanxi University, Taiyuan 030006, China

[a)]Author to whom correspondence should be addressed: yongmin@sxu.edu.cn

ABSTRACT

High-speed true random number generator is a building block in the modern information security system. We propose and demonstrate an efficient high-speed true random number generator based on multiple parallel self-timed rings (STRs). To improve the security, we evaluate the randomness of the entropy source by min-entropy and exploit the information-theoretically provable Toeplitz-hashing extractor. To minimize the consumption of hardware resources of a field programmable gate array at a predetermined high throughput and maximize the throughput with the limited hardware resources, we systematically derive and investigate the dependence of the data throughput and the total consumed resources of the random number generator on the system parameters. On this basis, we make a joint optimization for the degree of parallelism of the STRs and the extraction efficiency of the extractor. A 10-Gbps true random number generator is implemented efficiently, so that the output random bits can pass all the National Institute of Standards and Technology (NIST) and Dieharder test suites.

I. INTRODUCTION

True random number generators (TRNGs) play a crucial role in the modern information security field such as key preparation in symmetric algorithms, challenges generation in authentication,[1] and blinding values.[2] In certain cryptographic systems, TRNGs are required to generate keys for information encryption and decryption to achieve private communication tasks. The throughput of a random bit sequence required for a high speed cryptographic system is usually close to the order of Gbps. Therefore, it is necessary to design TRNGs with features of both high throughput and high quality randomness.

Over the past few years, a variety of TRNGs employing the random behavior of electronic noise have been proposed and implemented, such as chaotic mapping,[3–5] sampling oscillator jitter,[6–8] and metastable circuit.[9–11] They can be implemented on digital platforms, such as application specific integrated circuit (ASIC) and field programmable gate array (FPGA) device,[12–14] which can attain a high throughput and is more suitable for integrated systems. However, it has been shown that the realization of digital chaos mapping may lead to pseudo-randomness due to limited calculation accuracy.[15] The TRNGs based on metastable events are very sensitive to the manufacturing process and variations of operating conditions; in addition, precise delay control is also a challenge for metastable circuits.[16] Sampling oscillator jitter[17–19] is a preferred method to generate random numbers because of its simple structure and has been widely used in FPGAs. Among various implementations of sampling oscillator jitter, self-timed rings (STRs) are a suitable source of entropy and more robust to environmental fluctuations and process variability.[19] In the work of Cherkaoui *et al.*,[20] an entropy assessment model of the STR random generator was proposed, and a throughput of 200 Mbps on the FPGA was demonstrated. Recently, a staged-running STR architecture was proposed to achieve better performance with a large number of nodes and a complex mapping method,[21] which consume many of hardware resources of the FPGA. Note that most of the proposed TRNGs focus on improving the randomness of the generation, and the effective use of large-scale resources is still lacking of systematic analysis. This is critical for practical applications of the high-speed TRNG based on the FPGA in which small area, low power consumption, and cost are the main concerns.

For a realistic TRNG, the original random bitstream inevitably mixes with deterministic noises, such as power supply noise, and usually does not have satisfactory statistical characteristics. More importantly, from a security point of view, these deterministic noises may be known or manipulated by malicious attackers. A reliable post-processing process is crucial to ensure the quality of the random number generator. A Von Neumann corrector can eliminate the bias of the bit sequence but at the cost of a high loss of the raw bit sequence.[22] Some other attractive correctors based on linear functions have also been used in post-processing such as linear feedback shift register[7] or resilient corrector.[17] It was shown that the generator polynomial with the BCH code and the parity-check polynomial[20] are effective in lowering the adversary bias. The above-mentioned methods can improve the randomness of an initial bit sequence in some specific designs; however, the generated randomness based on these extractors is not information-theoretically provable and produces potentially weak security.

In this paper, we designed and implemented an efficient, high quality, and high-speed TRNG based on multiple parallel STRs in the FPGA. To ensure the high throughput of the generated random numbers, we proposed a parallel STR structure as a source of entropy of our TRNG. We also evaluated the randomness of entropy source by min-entropy instead of Shannon-entropy and exploited the information-theoretically provable Toeplitz-hashing extractor[23] to derive the lower bound of the randomness. We proposed an approach to optimize the structure of the entropy source and post-processing to achieve a hardware-efficient architecture that is suitable for compact implementations of the TRNG. To this end, the degree of parallelism of the STRs and the extraction efficiency of the extractor are carefully balanced to obtain the given throughput with the lowest resources assumption and to maximize the throughput with the limited hardware resources.

This article is organized as follows. In Sec. II, we provide background information of the STRs. In Sec. III, we present the random number generator based on the STR and analyze the security of each part theoretically in which min-entropy and the Toeplitz-hashing extractor are used in post-processing to quantify the randomness and distill the random bitstream. In Sec. IV, we propose a hardware-efficient architecture based on multiple parallel STRs and present the experimental results. Section V gives the statistical test results of the random number sequence and the comparison of our TRNG with other FPGA-based TRNGs. In Secs. VI and VII, a discussion and a conclusion are given, respectively.

II. SELF-TIMED RING OSCILLATOR

The schematic of the STR is shown in Fig. 1. It is composed of L elemental stages, and each stage has four ports: the forward input port F, the reverse input port R, the initialize input port S, and the output port C. Each stage consists of a Muller gate and an inverter and can be implemented by the Look-Up-Table (LUT) in the FPGA. The output port C also acts as a feedback loop in LUT to hold the value of the stage. As shown in Fig. 1, different stages of the STR are successively connected, and the output signal C_i of the stage i is not only connected to the forward input port F_{i+1} of the stage $i+1$ but also connected to the reverse input port R_{i-1} of the stage $i-1$. In this way, a handshake protocol is realized by interconnecting L stages, and a stable periodic signal is generated.

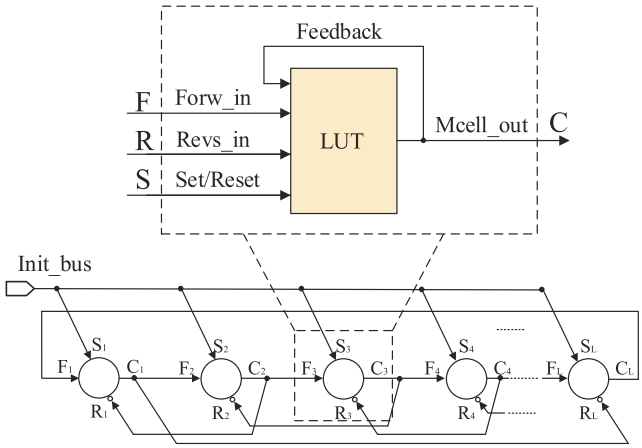

FIG. 1. STR architecture.

The output behavior of the STR can be described by bubbles and tokens. If C_i of the stage i is not equal to C_{i-1} of the stage $i-1$, then the stage i contains a token. Conversely, if C_i is equal to C_{i-1}, then the stage i contains a bubble. When C_i is not equal to C_{i-1} and C_i is equal to C_{i+1}, the token will propagate from the stage i to the stage $i+1$ and the bubble will propagate in the opposite direction. The STR operation process includes two phases: initialization and generation. During the initialization phase, the number of tokens (N_T) and the number of bubbles (N_B) in the STR are set by configuring the initial values of each stage. In the generation phase, each stage of the STR outputs a stable periodic signal. The ratio of N_T/N_B can affect the propagation mode of the event and the frequency of the output signal in the STR. It is also related to the ratio of D_{ff}/D_{rr}, where D_{ff} is the forward propagation delay and D_{rr} is the reverse propagation delay. Two modes exist for the events, a burst oscillation mode and an evenly spaced oscillation mode. When N_T/N_B satisfies

$$\frac{N_T}{N_B} \approx \frac{D_{ff}}{D_{rr}}, \tag{1}$$

the events run in an evenly spaced oscillation mode.[24] The difference between N_B and N_T is smaller, and the frequency of the output periodic oscillation signal is higher. Compared with a burst oscillation mode, the STR running in an evenly spaced propagation mode is more suitable for generating random numbers because of its higher oscillation frequency and good random quality.

III. STR-BASED TRUE RANDOM NUMBER GENERATOR

To achieve a reliable TRNG, we propose a STR-based scheme, in which a jitter acquisition unit and an information-theoretically provable post-processing unit are exploited. We use the phase jitter of the periodic signal of the STR as the source of entropy. The jitter noise of the period signal C_i is discretized into raw binary random bits by D flip–flop (DFF). Then, an XOR logic gate combines all the jitter regions that are contributed by each STR stage. Since all output signals of the STR propagate with the same intervals (phase resolution), there will be no overlapping areas of jitters after performing the XOR operation, which improves the utilization of entropy.

Therefore, jitter boundary and phase resolution are two key parameters that determine the quality of the randomness for the STR-based TRNG. The raw random bits synchronized with the system clock by synchronization (SYNC) unit are fed into the post-processing unit, which produces an almost uniformly distributed random sequence.

A. Jitter and phase resolution

The phase jitter of the periodic signal in the STR serves as the source of entropy. The sources of the jitter are mainly caused by local noises, such as flicker noise, thermal noise, and shot noise, and global noises caused by the operating environment such as power supply noise and electromagnetic environment noise.[25] The jitter that produces true randomness only depends on the local noise. Global deterministic noise is detrimental to random number generators, and attackers can predict and manipulate them to launch an attack. For the STR, different events propagate simultaneously, and the deterministic noise affects each event in the same way.[26] In this case, the impact of the global deterministic noise on the jitter of the output signal will be greatly suppressed. Therefore, the STR structure is robust to withstand the deterministic noise in the system.

In the evenly spaced propagation mode, all the output C_i are synchronous signals with the same period T. The phase resolution $\Delta\varphi$ refers to the interval between the entropy sources. Larger $\Delta\varphi$ means a bigger interval between the entropy sources, which is not conducive to the extraction of random numbers. When N_T and L are co-prime, the phase resolution between two adjacent output signals is determined by $\Delta\varphi = T/2L$, where the output period T of the STR is a function of N_T/L. Without changing the frequency of the STR, larger L can improve $\Delta\varphi$. For a given L, the smaller the oscillation period T of the STR, the higher the phase resolution $\Delta\varphi$, which is beneficial to extract random numbers. However, larger L requires more resources consumption. In Sec. IV, we will investigate the tradeoff between the resources consumption and generation rate of the random number generator.

B. Entropy acquisition

In the process of data acquisition from the STR jitter, the sampling clock is a critical factor that should been considered. The sensitivity of the sampling clock to temperature and power supply will affect the correlation and stability of the statistical characteristics of the TRNG's output sequence. To reliably harvest the entropy from the STR jitter, one of the STR outputs is selected to serve as the sampling clock in our scheme due to its robustness to the fluctuations of the temperature and voltage. Moreover, since the power frequency can also affect the deterministic jitter noise, when the sampling clock of the TRNG is generated by theSTR, the risk of power frequency attacks can be significantly reduced. The STR-based TRNG allows sampling frequencies f_s up to the oscillating frequency of the STR.[20] Here, we use one of the STR outputs to realize trigger acquisition of the phase jitter. In this case, the sampling clock is synchronized with the STR, which suppresses the pseudo-randomness as well as maximizing the throughput of the random numbers.[27]

C. Post-processing

1. Entropy assessment

The objective of entropy assessment is to provide a lower bound of randomness as a function of STR-based TRNG's characteristics. We quantify the randomness by min-entropy instead of Shannon-entropy, which represents the average information of a random distribution in Ref. 20. The min-entropy for a binary variable X with distribution $P(X)$ is directly related to the maximum probability of any measurement results and is defined by

$$H_{\min} = -\log_2[P_{\max}(X = \mu)], \quad \mu = 1 \text{ or } 0. \tag{2}$$

Based on the theoretical model of the entropy source,[20] the following assumptions are made:

(1) Since the effective entropy source mainly comes from the local Gaussian jitter, the jitter of the STR output periodic signal follows a Gaussian distribution $N(\frac{\Delta\varphi}{2}, \sigma^2)$, where $\Delta\varphi$ is the phase resolution and σ is the jitter variance.
(2) Sampling clock is an ideal clock without jitter.
(3) The entropy of non-adjacent events after sampling and XOR is approximately 0 so that $H(\psi) \approx H(R_{j-1} \otimes R_j)$.

The sampling bits R_j are combined by XOR operation to generate the random bit ψ. The probability of the random bit ψ can be determined by

$$P(\psi = 0) = p_1(1 - p_2) + p_2(1 - p_2), \tag{3}$$

where $p_1 = p(t_{Xj} \le t_s)$ and $p_2 = p(t_{Xj+1} \le t_s)$.

From (3), we can see that $P(\psi = 0)$ depends on the relative positions of t_{Xj}, t_{Xj-1}, and t_s, where t_{Xj} is the time of the event X_j. Equation (3) can also be expressed by the cumulative distribution function Φ of the standard normal distribution $N(0, 1)$ as follows:

$$P(\psi = 0) = \Phi\left(\frac{t_s - \frac{\Delta\varphi}{2}}{\sigma}\right) + \Phi\left(\frac{t_s + \frac{\Delta\varphi}{2}}{\sigma}\right) - 2\Phi\left(\frac{t_s - \frac{\Delta\varphi}{2}}{\sigma}\right)\Phi\left(\frac{t_s + \frac{\Delta\varphi}{2}}{\sigma}\right). \tag{4}$$

Combine (2) and (4), the min-entropy of the random sequence is given by

$$H_{\min} = -\log_2[P_{\max}(\psi = 0)]. \tag{5}$$

From (4) and (5), it is known that the min-entropy $H_{\min}$ depends on the jitter variance σ, the phase resolution $\Delta\varphi$, and the sampling time t_s. Notice that the sampling time t_s cannot be accurately controlled. To obtain a lower bound on the min-entropy $H_{\min}$, we illustrate the relationship between the sampling time t_s and the min-entropy under different σ. As shown in Fig. 2, the min-entropy reaches its lower bound value at $t_s = 0$. Hereafter, we use the lower bound of the min-entropy for the random number generator, which corresponds to the worst-case scenario.

To evaluate the effect of the length of STR stages L on the min-entropy, the dependence of the min-entropy $H_{\min}$ on L is plotted, where $k = \sigma/T$. As shown in Fig. 3, when the stages-length L is small, the extractable entropy is limited, particularly for the case of small jitter variance. For large L, sufficient entropy can be extracted

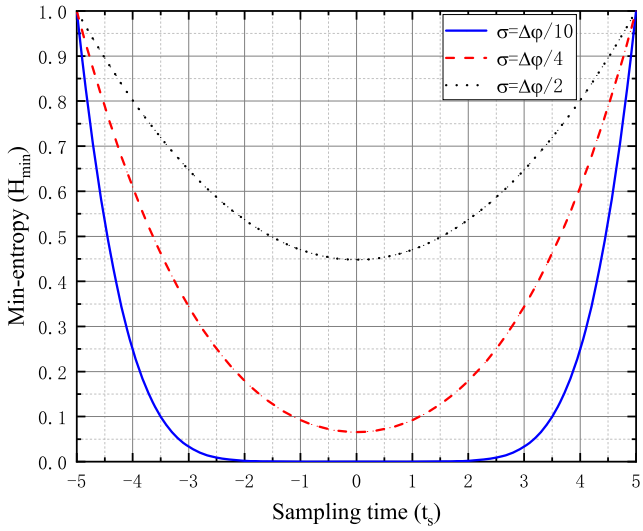

FIG. 2. The min-entropy as a function of the sampling time t_s under different jitter variance σ, $\Delta\varphi = 10$.

even if the jitter variance is small. Therefore, the min-entropy can be effectively improved by increasing L.

2. Toeplitz-hashing extractor

In general, the raw random bit sequence from the STR has a slight bias phenomenon and a weak correlation is existed between the bits, which deteriorate the randomness of the output bit sequence. To solve this problem, an extractor is usually used in the post-processing process to distill the randomness. The Toeplitz-hashing extractor, one of the universal hash functions, has been proved to be a strong extractor by leftover hash lemma.[28] One of the advantages of this strong extractor is that it can reuse the random seed. Therefore, the outputs of multiple STRs can be processed simultaneously by multiple extractors with the same seed.[23] Another advantage is the simple construction of the Toeplitz matrix. Only the elements of the first row and the first column of the matrix need to be initialized, and other elements in the matrix can be obtained from the elements of the first row and the first column. Compared with the conventional universal hash function that requires $m \times n$ random bits to construct, the Toeplitz matrix greatly reduces the consumed random bits to $m + n - 1$.

To further improve the randomness, we consider the security parameter ε, which represents the statistical distance between two probability distributions of the generated random string and a perfectly random string. If the statistical distance of two probability distributions A and B on the same domain T is bounded by ε, they are ε-close,

$$\|A - B\| = \frac{1}{2}\sum_{u\in T}|P(A = u) - P(B = u)| \le \varepsilon. \tag{6}$$

Considering the security parameter ε, the output length n of a Toeplitz-hashing extractor is given by

$$n = mH_{\min} - \log\frac{1}{\varepsilon^2}, \tag{7}$$

where m is the length of the raw random bit sequence. From (7), we can see that m affects the extraction rate $\gamma = n/m$ of a random number extractor. We define the extraction efficiency η as the ratio of extraction rate to the min-entropy,

$$\eta = \frac{\gamma}{H_{\min}}. \tag{8}$$

Given a security parameter $\varepsilon = 2^{-50}$, the extraction efficiency η under different input lengths m can be calculated using (7) and (8). As shown in Fig. 4, given the min-entropy, the extraction efficiency increases with the input length and gradually reaches its

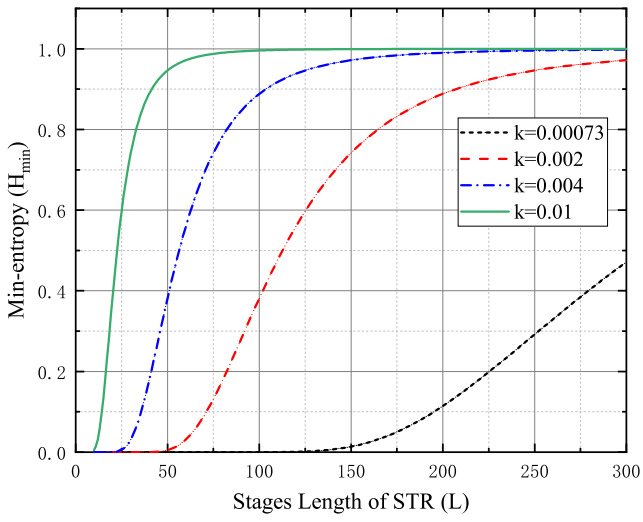

FIG. 3. The min-entropy per output bit vs the STR stage L under different k.

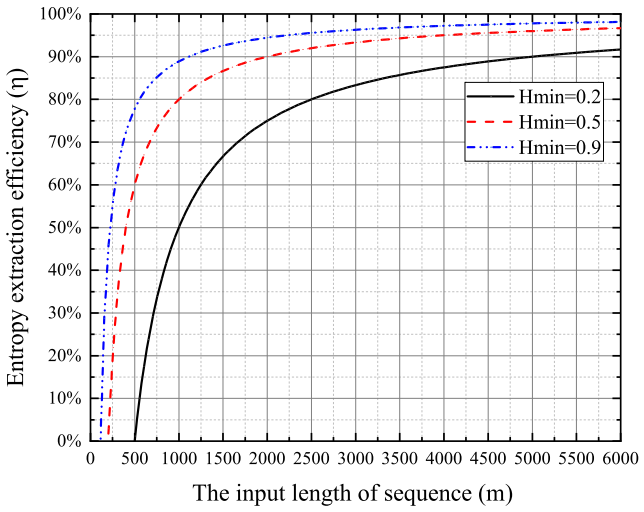

FIG. 4. The extraction efficiency γ of the extractor vs the input length of the random sequence m under different min-entropy. The security parameter is set to $\varepsilon = 2^{-50}$.

extreme value. When the min-entropy is small and the length of the input random sequence is not long enough, there will be no extractable random numbers. For large min-entropy, high extraction efficiency can be achieved at a small input length. Notice that the input length m of the random sequence is equal to the number of column of the Toeplitz matrix, and a small m means less use of hardware resources. Therefore, it is necessary to balance the extraction efficiency and resources consumption during the design of hardware-based Toeplitz-hashing extractor.

IV. PROPOSED ARCHITECTURES AND IMPLEMENTATION

Due to the limited bandwidth of the output signal from the STR, it is difficult to achieve Gbps throughput with a single STR-based TRNG. To realize a high-speed random number generator, we propose a TRNG scheme based on multiple parallel STRs. As shown in Fig. 5, a number of parallel STR channels run simultaneously, and each channel consists of a L-stage STR, an entropy acquisition unit, and a synchronization unit. The configuration unit is used to initialize the STRs. The random number generated by all the channels uses time division multiplexing technology to transmit data to the post-processing unit through the Multiplexer (MUX) unit, which is controlled by the data control unit. In addition, to test the statistical properties of the generated random numbers, a portion of the random numbers are transferred to a personal computer through the Gigabit Ethernet port, where a statistical characteristics analysis is performed with National Institute of Standards and Technology (NIST) and Dieharder test suites.

The proposed scheme is implemented on a Xilinx Virtex-7 FPGA device XC7VX485T, which is a high-performance FPGA device manufactured using the 28 nm process. Each stage of the STR is built in the FPGA with a four-port LUT. Since the concrete layout of STRs in the FPGA has an influence on the quality of randomness, we adopt a manual layout strategy to place the LUTs close to each other and form a ring topology. In this way, the delay lines between the connecting STR units are minimized, which is helpful to balance delays between the ring stages and realize events distributed evenly.

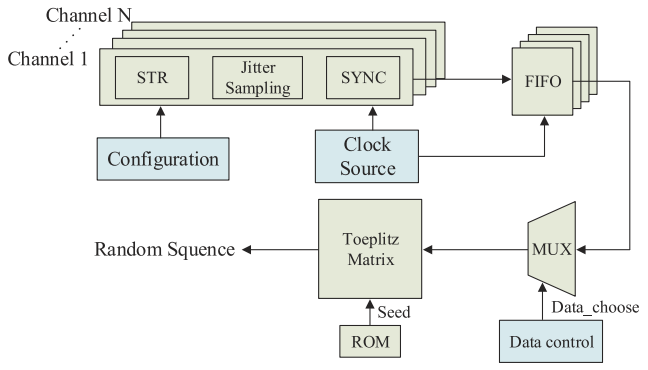

FIG. 5. Multiple parallel STRs-based random number generator.

A. Analysis of throughput and hardware resources

The total throughput T_h is an important indicator to evaluate the random number generator, which can be expressed as

$$T_h = N f_s \eta H_{\min}, \tag{9}$$

where N is the degree of parallelism of the channels and f_s is the sampling frequency. From (9), the total throughput is closely related to four parameters. Among them, N, f_s, and $H_{\min}$ depend on the entropy source, whereas the extraction efficiency η is subject to the post-processing process. Given the target throughput T_h, the entropy source and post-processing unit should be jointly optimized to minimize the consumption of the hardware resources.

The total LUT resource S_T consumed by the TRNG consists of two parts: the entropy source part S_S and the post-processing part S_P,

$$\begin{aligned} S_T &= S_S + S_P, \\ S_S &= N\upsilon L, \\ S_P &= \tau mn, \end{aligned} \tag{10}$$

where υ and τ are the resources consumption coefficients of the entropy source and post-processing, respectively. υ and τ are used to quantify the amount of resources consumed by each unit of entropy source and post-processing. Since each stage of STR can be realized by one LUT, the resources consumption coefficient υ is 1.

To illustrate the optimization method in detail, here we choose four STRs with typical lengths (31, 63, 127, and 255) as a fundamental element to construct a parallel random number generator. According to (4), the min-entropy $H_{\min}$ is not only related to L but also related to the period T and jitter variance σ. The jitter variance σ in STRs is independent of the length of stages L and can be estimated according to the method mentioned in Ref. 26. We use an oscilloscope with a sampling rate of 25 GSample/s (Teltronix, MSO64) to directly sample the jitter signals. By testing 10-M accumulated periodic signals, the average jitter variance measured is 3.62 ps.

In the above, we have discussed the realization of highest frequency $f_{\max}$ of the periodic signal in the evenly spaced oscillation mode. However, without the knowledge of the relationship between $f_{\max}$ and stages-length L of STR, it is difficult to estimate the throughput. In addition, the number of tokens in the STR structure will change the value of $f_{\max}$. By adjusting the number of tokens in the STR, we measured the value of $f_{\max}$ of different stages-length with the oscilloscope.[26] According to the measured $f_{\max}$ and jitter variance of the STR output signal, the min-entropy of the output signal is determined. Table I lists the oscillation period, phase resolution, and min-entropy for different STR configurations. As can be seen, when the stages-length of STR is $L = 31$, the highest frequency $f_{\max}$ of the output signal can reach 780 MHz. As the stages-length L increases, the highest output frequency $f_{\max}$ degrades. Although the increase in L can improve the min-entropy, it will also decrease $f_{\max}$ at some extent. However, larger L of the STR means that more FPGA logic resources and power will be consumed.

We employ the Toeplitz-hashing extractor to distill random numbers from the raw random sequences. In Sec. III, we know that the size of the Toeplitz matrix affects the extraction efficiency.

TABLE I. Oscillation period, phase resolution, and min-entropy for different self-timed ring configurations.

L	f_{max} (MHz)	T (ns)	$\Delta\varphi$ (ns)	H_{min}
31	779.78	1.283	0.207	0.006 12
63	600.63	1.665	0.132	0.097 65
127	570.99	1.751	0.069	0.478 95
255	508.30	1.967	0.036	0.779 69

A larger Toeplitz matrix can improve the extraction efficiency η of the extractor, but it will also cause more resources consumption. We get the LUT resources consumed by the Toeplitz matrix for input lengths m and plot the curve with experimental data. The resources consumption coefficient τ is determined to be 0.528 from the slope of the curve.

B. Optimization of hardware resources

The goal of optimization is to achieve a high throughput with the least FPGA resources. As mentioned previously, the entropy source and the post-processing are two key processes that determine the throughput of the random number generator, and they should be well balanced in the optimization of the TRNG based on multiple parallel STRs. If the min-entropy of the entropy source is weak, a larger size Toeplitz matrix and more entropy source channels are required to realize high throughput. On the contrary, for an entropy source that can provide a large min-entropy, the size of the Toeplitz matrix and the degree of parallelism can be reduced accordingly.

Substituting $\upsilon = 1$ and $\tau = 0.528$ into (10), the total resources consumption of the random number generator can be given by

$$S_T = NL + 0.528mn, \tag{11}$$

where

$$N = T_h/f_s\left[H_{min} - \left(\log\frac{1}{\varepsilon^2}\right)\Big/m\right]. \tag{12}$$

For a 10-Gbps throughput, the consumption FPGA resources under different configurations of the entropy source and post-processing are simulated using (9) and (11), as shown in Table II. Six Toeplitz-hashing extractors with different extraction efficiency are used to extract random numbers from the entropy sources with different stages-length L and degree of parallelism N.

According to the analysis above (Table II), the min-entropy of the STR increases with the stages-length L. Given the throughput

TABLE II. Comparison of resources consumption under different configurations of entropy source and post-processing for a 10-Gbps TRNG.

Entropy source		Post-processing							
L	N	M	n	γ	η (%)	S_S	S_P	S_T	ω[a,b] (%)
31	2 206	326 797	1900	0.0058	95.0	68 386	327 843 138	327 911 524	108 007.75
	2 794	65 360	300	0.0046	75.0	86 614	10 352 942	10 439 556	3 438.59
	3 810	36 311	123	0.0034	55.0	118 110	2 343 259	2 461 369	810.72
	5 987	25 139	54	0.0021	35.0	185 597	714 701	900 298	296.54
	11 116	**19 935**	**23**	**0.0012**	**18.9**	**344 596**	**242 091**	**586 687**	**194.87**[c]
	13 970	19 224	18	0.0009	15.0	433 070	179 117	612 187	201.64
63	180	20 482	1900	0.0928	95.0	11 340	20 546 851	20 558 191	6 771.47
	228	4 096	300	0.0732	75.0	14 364	648 848	663 212	218.44
	310	2 276	123	0.0537	55.0	19 530	146 859	166 389	54.81
	488	1 576	54	0.0342	35.0	30 744	44 793	75 537	24.88
	668	**1 363**	**34**	**0.0249**	**25.5**	**42 084**	**24 469**	**66 553**	**21.92**[c]
	1 137	1 205	18	0.0146	15.0	71 631	11 226	82 857	27.29
127	39	4 175	1900	0.4550	95.0	4 953	4 189 164	4 194 117	1381.46
	49	835	300	0.3592	75.0	6 223	132 290	138 513	45.62
	67	464	123	0.2634	55.0	8 509	29 943	38 452	12.67
	105	322	54	0.1676	35.0	13 335	9 133	22 468	7.40
	111	**310**	**49**	**0.1581**	**33.0**	**14 097**	**8 021**	**22 118**	**7.29**[c]
	244	246	18	0.0718	15.0	30 988	2 289	33 277	10.96
255	27	2 565	1900	0.7407	95.0	6 885	2 573 331	2 580 216	849.87
	34	513	300	0.5848	75.0	8 670	81 264	89 934	29.62
	46	286	123	0.4288	55.0	11 730	18 393	30 123	9.92
	64	**211**	**65**	**0.3081**	**39.5**	**16 320**	**7242**	**23 562**	**7.76**[c]
	73	198	54	0.2729	35.0	18 615	5610	24 225	7.98
	169	151	18	0.1170	15.0	43 095	1406	44 501	14.66

[a] $\omega = S_T/S_{FPGA}$, where S_{FPGA} is the total LUT resources of the FPGA chip we utilized.
[b] For $\omega > 75\%$, the simulation results of LUT resource consumption of the FPGA chip are shown.
[c] The lowest occupation of total resources for constant L.

T_h = 10 Gpbs, security parameter $\varepsilon = 20^{-50}$, and the constant stages-length L, the degree of parallelism N decreases with the increase in the extraction efficiency η. Small N means lower resources consumption of the entropy source S_S and higher η means more resources consumption S_P for the extractor. Hence, there is an optimal degree of parallelism N or, equivalently, an extraction efficiency η, which corresponds to the lowest occupation of total resources S_T for constant L, as shown in Table II. Notice that for different stages-length L of STR, the corresponding optimal extraction efficiency is different, which results in distinct S_T. From these S_T, the lowest S_T can be found out.

Obviously, the STRs with $L = 31$ is not suitable as entropy sources. Because the entropy provided by this structure is very low, even if the degree of parallelism of the channels and the extraction efficiency of the extractor are optimized, the required throughput (10 Gbps) cannot be reached when all the FPGA resources are occupied. For $L = 255$, since the min-entropy is large enough, even if the Toeplitz matrix with a moderate extraction efficiency $\eta = 39.5\%$ is employed in the post-processing, a 10-Gbps throughput is feasible. In this case, a high throughput can be achieved with the degree of parallelism $N = 64$.

In Table II, by comparing the total consumed LUT resources in different implementation schemes, we find the optimal parameters settings for the entropy source and the extractor, which are $L = 127$, $N = 111$, $m = 310$, and $n = 49$. Using these optimal parameters, we implement the multiple parallel STR-based TRNG on a Xilinx Virtex-7 FPGA device XC7VX485T, and a throughput of 10 Gbps is achieved. The total consumption of LUT resources is 22 118, which occupies ~7.29% of the total LUT resources (S_{FPGA} = 303 600) of the FPGA chip.

C. Optimization of throughput

In some scenarios, the hardware resources available for the TRNG are fixed. In these cases, the goal of optimization is to achieve a throughput as high as possible with the given FPGA hardware resources. Similar to the analysis process above, we detail the optimization procedures in the following and show how to balance the design parameters of the entropy sources with typical STR stages-lengths (31, 63, 127, and 255) and the post-processing to maximize the throughput of our parallel random number generator. To this end, the throughput of the TRNG can be derived by combining (9) and (11) and be expressed by

$$T_h = \frac{(\omega S_{FPGA} - 0.528mn)nf_s}{mL}. \tag{13}$$

For the given FPGA resources ωS_{FPGA} and security parameter ε of 20^{-50}, we can find the maximum of the throughput by varying the input length m of the Toeplitz matrix according to (13) for different stages-length L. By comparing the resulting throughputs of different stages-length L, the highest throughputs and the corresponding optimized m are determined. Then, using (7) and (12), the degree of parallelism N of the channels and the output length n of the extractor can be calculated, respectively. Finally, the parallel random number generator can be built based on the optimal parameters L, N, m, and n.

As shown in Fig. 6, for different stages-length L of STR, the corresponding throughput T_h is plotted as a function of the optimal input length m of the Toeplitz matrix. When $\omega = 0.05$, it can be seen that the maximal throughput of the STR with stages-length $L = 31$ is only a few Mbps, and thus, it is not suitable as the entropy source. Because the min-entropy that the entropy source of $L = 31$ can provide is very low, in this case, the required input length of the extractor is large (m = 16 340) and more LUT resources (56.71%) are required to construct the Toeplitz matrix to guarantee a high extraction efficiency for entropy sources. Although the resources employed in post-processing and entropy source can be carefully balanced through the optimization strategy, the attainable throughput is still limited. On the contrary, for an entropy source with a relatively large min-entropy, since it can provide sufficient entropy, it is not necessary to adopt a long input length m to improve the extraction efficiency. In this case, more resources can be used to construct the entropy source and a higher throughput can be obtained. In addition, considering that the relatively short input length m of the Toeplitz matrix can cause an insufficient processing capability of the post-processing, a higher system clock is necessary to deal with the post-processing process. By comparing the throughput obtained after optimization of stages-length L and input length m in Fig. 6, we find that the STR with $L = 127$ is the best candidate to serve as the basic unit of the entropy source for our design.

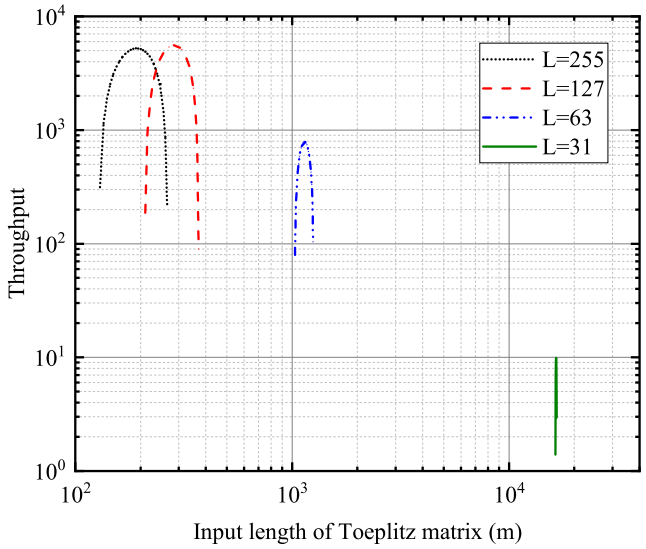

FIG. 6. The throughput vs the input length m of the Toeplitz matrix at different stages-length L. The total occupied coefficient of LUT resources of the FPGA is $\omega = 0.5$.

To evaluate the performance of random number generators with different configurations, we define resource utilization efficiency κ as the ratio of the throughput of the random number generator to the resources consumed,

$$\kappa = T_h/S_T. \tag{14}$$

In Table III, we summarize the maximal throughput and resource utilization efficiency κ under different resource consumptions for $L = 127$. For the case that the available LUT resources are low ($\omega = 0.05$), the maximal throughput of 5641.7 Mbps can be achieved with optimized configuration parameters ($L = 127$, $N = 76$,

TABLE III. The maximal throughput and resource utilization efficiency κ under different resource consumptions for $L = 127$.

L	N	ω	T_h (Mbps)	m	n	S_S/S_T (%)	κ
127	931	0.5	145 157	483	132	77.89	0.9562
127	443	0.25	58 610	401	93	74.12	0.7722
127	254	0.15	28 982	355	71	70.83	0.6364
127	76	0.05	5 641	284	37	63.58	0.3716

$m = 284$, and $n = 37$). The resources consumption ratio S_S/S_T of the entropy source to the available hardware resources is 63.58%, and the resource utilization efficiency $\kappa = 0.3716$. When the total occupied coefficient ω is increased to 0.5, a considerable throughput of 145 157 Mbps can be achieved and the resource utilization efficiency κ is drastically improved to 0.9562. We also note that S_S/S_T does not vary significantly while ω changes.

V. EVALUATION

A. Randomness evaluation

To test the randomness of the binary sequence generated by our random number generator, we employ both the NIST SP 800-22 and DIEHARD statistical test suites for data statistical testing. To this end, the generated random numbers are transferred to a computer through an Ethernet interface. The total amount of test data is 1 Gbits, which is divided into 100 sequences, i.e., each sequence contains 10 Mbits data. The NIST test results for our data are shown in Table IV. For all 15 test terms of the NIST test suite, the P-value of each test sequence is larger than the lowest significant level $P = 0.01$. For items that include multiple sub-tests, the arithmetic average is selected. The confidence level proportion of sequences that satisfying $P > 0.01$ is larger than 0.98, which indicates that the generated random bitstream has good statistical characteristics. Our extracted bit sequence from the STR-based TRNG also successfully passes the DIEHARD test suite, as shown in Table V. For the test terms that provide a set of P-values, a Kolmogorov–Smirnov (KS) test is applied. The test is successful when $0.025 < P < 0.975$ at the 0.05 level. Notice that the raw bit sequence directly generated from the TRNG without post-processing failure to pass the statistical tests due to its low entropy.

B. Comparison of our device with other FPGA-based TRNG

Table VI shows the consumed hardware resources of entropy resource, the method of post-processing, and the throughput for our TRNG and other typical FPGA-based TRNG. Using the ring oscillator (RO) as a random source, Wang *et al.* achieved a throughput of 7.69 Gbps with less hardware consumption for entropy resource, but a large number of resources are required to carry out the high resolution time-to-digital converter (TDC) based sampling.[3] Cherkaoui *et al.* presented a 200 Mbps TRNG using a 255-stages STR.[20] Notice that it consumes more hardware resources for the entropy source than our device. Recently, another interesting work based on coherent sampling was reported by Martin *et al.*,[29] the corresponding performance is remarkable because of portability. However, a throughput of only 4 Mbps was obtained.

Some designs using hard macros or specific components such as phase-locked loop (PLL) or digital-clock-manager (DCM) as entropy source or clock are attractive in terms of resource consumption and design complexity.[30,31] Johnson *et al.* exploited the feature of a dynamic partial reconfiguration of the DCM to improve the randomness as well as the performance in terms of area and throughput.[31] Note that DCM or PLL resources of the FPGA are very limited; therefore, it is impossible to configure a large number of DCMs to achieve a high-speed TRNG.

TABLE IV. NIST test result.

Statistical test	P-value	Proportion	Result
Frequency	0.275 709	0.991	Success
BlockFrequency	0.104 993	0.989	Success
CumulativeSums	0.214 439	0.991	Success
Runs	0.620 465	0.993	Success
LongestRun	0.029 796	0.998	Success
Rank	0.896 345	0.996	Success
FFT	0.651 693	0.985	Success
NonOverlappingTemplate	0.534 581	0.993	Success
OverlappingTemplate	0.206 629	0.986	Success
Universal	0.707 513	0.992	Success
ApproximateEntropy	0.128 132	0.984	Success
RandomExcursions	0.565 637	0.992	Success
RandomExcursionsVariant	0.571 209	0.987	Success
Serial	0.216 713	0.992	Success
LinearComplexity	0.104 371	0.989	Success

TABLE V. Diehard test result.

Statistical test	P-value	Result
Birthday spacings (KS)	0.150 782	Success
Overlapping permutations	0.831 665	Success
Ranks of 31 × 31 matrices	0.332 820	Success
Ranks of 32 × 32 matrices	0.671 777	Success
Ranks of 6 × 8 matrices (KS)	0.688 579	Success
Bitstream test	0.205 86	Success
Monkey test OPSO	0.152 6	Success
Monkey test OQSO	0.080 6	Success
Monkey test DNA	0.144 6	Success
Count 1's in stream of bytes	0.384 155	Success
Count 1's in specific bytes	0.106 967	Success
Parking lot test (KS)	0.465 268	Success
Minimum distance test (KS)	0.742 220	Success
3DSPHERES test (KS)	0.320 204	Success
Squeeze test	0.928 020	Success
Overlapping sums test (KS)	0.628 944	Success
Runs test (up) (KS)	0.347 394	Success
Runs test (down) (KS)	0.553 801	Success
Craps test no. of wins	0.380 752	Success
Craps test throws per game	0.301 202	Success

TABLE VI. Comparison of the proposed TRNG with other FPGA-based TRNG.

Reference	This work	Cherkaoui *et al.*[20]	Martin *et al.*[29]	Wang *et al.*[3]
Entropy source	STR	STR	STR	RO
Post-processing	Toeplitz-hashing	Parity filter	Parity filter	LFSR
LUTs of entropy	14 097	320	32	448
Throughput	10 Gbps	200 Mbps	4 Mbps	7.69 Gbps

VI. DISCUSSION

The pseudorandom number generator (PRNG) has been widely exploited in many applications. In the following, the performance of TRNG and PRNG in numerical simulation and cryptography applications is discussed, and a brief comparison of some typical FPGA-based PRNGs with our proposed TRNG is given.

PRNG, such as the Mersenne twister,[32] chaotic generator,[33] linear congruential generator,[34,35] and shuffled nested Weyl sequence (SNWS) generator,[36–38] can generate faster random bitstream than other types of TRNGs. Their results are reproducible, which makes them very popular in numerical simulations. For example, in the Monte Carlo method, the pseudorandom number that simulates the desired distribution statistics can be considered random enough even if a predictable sequence is generated. i we know the seeds in PRNG, we can repeat the simulation results, while TRNG makes it challenging to reproduce the results during debugging and operation. For TRNG, an alternative way to repeat the simulation result is storing the sequence generated. However, it is a challenge for the simulation that requires a large amount of data. Notice that the length of the SNWS generator can be increased to any requested length by applying extra loops and non-linear functions.[38]

For many cryptology applications, it is not enough for random numbers to obey the statistical properties of random numbers such as uniformity and independence. The unpredictability (forward security) and backward security of the random are necessary, i.e., knowledge of a portion of the random sequence cannot allow a malicious attacker to deduce the previous and subsequent values of the generator.[39]

TRNG employs some fundamental random physical phenomena to generate random sequences, and these physical events are intrinsically unpredictable. In contrast, most PRNG does not possess the ability to generate random numbers that can be used in secure cryptology applications. In principle, the internal state of the PRNG can be inferred from a large number of its output sequences and the whole output sequence can be predicted even if the parameters in the generator are unknown.[39]

Finally, we make a brief comparison of the hardware resources and throughput of some typical FPGA-based PRNGs with our proposed TRNG. In terms of the throughput, the parallel linear feedback shift registers (LFSRs)-based PRNG[40] can reach a throughput of 343 Gbps, while it is 10 Gbps for our proposed STR-based TRNG. The Mersenne Twister-based PRNG[41] with parallel block random access memories (BRAMs) is competitive when considering the ratio of throughput to the resource occupation of the FPGA. The above-mentioned PRNGs can be employed as fast and economical generators in the field of numerical simulation. In contrast, our STR-based TRNG can generate an unpredictable and secure random bit sequence for cryptology applications.

VII. CONCLUSION

In this article, we have designed and demonstrated a hardware-efficient high-speed true random number generator with a 10-Gbps data rate on an FPGA. To improve the randomness of the generated random numbers, the STR is used as the basic unit of the entropy source. By modeling the entropy source and the acquisition process, a lower bound of the min-entropy of the sampled output sequence is derived to guarantee the randomness of the output sequence. Furthermore, a Toeplitz-hashing extractor with a security parameter is exploited in the post-processing to safely extract the random numbers from the raw bit sequence. Note that the consumed hardware resources at a high throughput of TRNG based on FPGA is a critical issue. By analyzing the effects of the key parameters of the entropy source and post-processing on the throughput and consumed hardware resources at a predetermined data rate or limited resources, we propose an approach to improve the utilization efficiency of the hardware resources by making a joint optimization for the degree of parallelism of the STRs and the extraction efficiency of the extractor.

Our work provides several effective design guidelines for the STR-based high-speed random number generators. The results of resource optimization present an effective method to determine the stages-length of the STR in the entropy source, the degree of parallelism of the channels, and the extraction efficiency in the post-processing unit. Although the min-entropy can be maximized by increasing the stages-length of the STR, the designers should not choose the maximum of min-entropy as the goal. The optimization approaches for a given throughput or hardware resources can improve the resource utilization efficiency and provide guidance for TRNG built on different hardwares. In fact, our approach for performance optimization can be applied to other design platforms, such as ASIC, where limited hardware resources are a concern. Furthermore, it would be very interesting work to investigate the performance optimization of a heterogeneous framework TRNG consisting of different STRs based on the results of our work.

ACKNOWLEDGMENTS

This work was supported by the National Key R&D Program of China (Grant No. 2016YFA0301403), the National Natural Science Foundation of China (NSFC) (Grant Nos. 11774209

and 61378010), the Key R & D Project of Shanxi Province (Grant No. 201803D121065), and Shanxi 1331KSC.

DATA AVAILABILITY

The data that support the findings of this study are available from the corresponding author upon reasonable request.

REFERENCES

[1]X. Huang, Y. Xiang, E. Bertino, J. Zhou, and L. Xu, "Robust multi-factor authentication for fragile communications," IEEE Trans. Dependable Secure Comput. **11**, 568–581 (2014).
[2]M. Bucci and R. Luzzi, "Fully digital random bit generators for cryptographic applications," IEEE Trans. Circuits Syst. I **55**, 861–875 (2008).
[3]Y. Wang, C. Hui, C. Liu, and C. Xu, "Theory and implementation of a very high throughput true random number generator in field programmable gate array," Rev. Sci. Instrum. **87**, 044704 (2016).
[4]C. Li, B. Feng, S. Li, J. Kurths, and G. Chen, "Dynamic analysis of digital chaotic maps via state-mapping networks," IEEE Trans. Circuits Syst. I **66**, 2322–2335 (2019).
[5]A. Beirami and H. Nejati, "A framework for investigating the performance of chaotic-map truly random number generators," IEEE Trans. Circuits Syst. II **60**, 446–450 (2013).
[6]D. Lubicz and N. Bochard, "Towards an oscillator based TRNG with a certified entropy rate," IEEE Trans. Comput. **64**, 1191–1200 (2015).
[7]K. Demir and S. Ergun, "Random number generators based on irregular sampling and Fibonacci-Galois ring oscillators," IEEE Trans. Circuits Syst. II **66**, 1718–1722 (2019).
[8]J. D. J. Golic, "New methods for digital generation and postprocessing of random data," IEEE Trans. Comput. **55**, 1217–1229 (2006).
[9]C. Tokunaga, D. Blaauw, and T. Mudge, "True random number generator with a metastability-based quality control," IEEE J. Solid-State Circuits **43**, 78–85 (2008).
[10]P. Z. Wieczorek, "An FPGA implementation of the resolve time-based true random number generator with quality control," IEEE Trans. Circuits Syst. I **61**, 3450–3459 (2014).
[11]M. Majzoobi, F. Koushanfar, and S. Devadas, "FPGA-based true random number generation using circuit metastability with adaptive feedback control," in *Proceedings of International Workshop on Cryptographic Hardware and Embedded Systems* (Springer Berlin Heidelberg, 2011), pp. 17–32.
[12]N. Nalla Anandakumar, S. K. Sanadhya, and M. S. Hashmi, "FPGA-based true random number generation using programmable delays in oscillator-rings," IEEE Trans. Circuits Syst. II **67**, 570–574 (2020).
[13]B. Ray and A. Milenkovic, "True random number generation using read noise of flash memory cells," IEEE Trans. Electron Devices **65**, 963–969 (2018).
[14]Q. Wang, S. Yu, C. Li, J. Lü, X. Fang, C. Guyeux, and J. M. Bahi, "Theoretical design and FPGA-based implementation of higher-dimensional digital chaotic systems," IEEE Trans. Circuits Syst. I **63**, 401–412 (2016).
[15]M. François, D. Defour, and C. Negre, "A fast chaos-based pseudorandom bit generator using binary64 floating-point arithmetic," Informatica **38**, 115–124 (2014); available at http://www.informatica.si/index.php/informatica/article/view/691.
[16]V. B. Suresh and W. P. Burleson, "Entropy and energy bounds for metastability based TRNG with lightweight post-processing," IEEE Trans. Circuits Syst. I **62**, 1785–1793 (2015).
[17]B. Sunar, W. Martin, and D. Stinson, "A provably secure true random number generator with built-in tolerance to active attacks," IEEE Trans. Comput. **56**, 109–119 (2007).
[18]K. Wold and C. H. Tan, "Analysis and enhancement of random number generator in FPGA based on oscillator rings," in *2008 International Conference on Reconfigurable Computing and FPGAs* (IEEE, Cancun, Mexico, 2008), pp. 385–390.
[19]A. Cherkaoui, V. Fischer, A. Aubert, and L. Fesquet, "Comparison of self-timed ring and inverter ring oscillators as entropy sources in FPGAs," in *Proceedings of Design, Automation and Test in Europe Conference and Exhibition (DATE)* (IEEE, Dresden, Germany, 2012), pp. 1325–1330.
[20]A. Cherkaoui, V. Fischer, L. Fesquet, and A. Aubert, "A very high speed true random number generator with entropy assessment," in *Proceedings of International Workshop on Cryptographic Hardware and Embedded Systems (CHES'13)* (Springer Berlin Heidelberg, Santa Barbara, CA, USA, 2013), Vol. 8086, pp. 179–196.
[21]Y. Liu, R. C. C. Cheung, and H. Wong, "A bias-bounded digital true random number generator architecture," IEEE Trans. Circuits Syst. I **64**, 133–144 (2017).
[22]V. Rožić and I. Verbauwhede, "Hardware-efficient post-processing architectures for true random number generators," IEEE Trans. Circuits Syst. II **66**, 1242–1246 (2019).
[23]X. Ma *et al.*, "Postprocessing for quantum random-number generators: Entropy evaluation and randomness extraction," Phys. Rev. A **87**, 062327 (2013).
[24]J. Hamon, L. Fesquet, B. Miscopein, and M. Renaudin, "High-level time-accurate model for the design of self-timed ring oscillators," in *2008 14th IEEE International Symposium on Asynchronous Circuits and Systems* (IEEE, Newcastle upon Tyne, UK, 2008), pp. 29–38.
[25]V. Fischer, F. Bernard, N. Bochard, and M. Varchola, "Enhancing security of ring oscillator-based trng implemented in FPGA," in *Proceedings of IEEE International Conference on Field Programmable Logic Application* (IEEE, Heidelberg, Germany, 2008), pp. 245–250.
[26]A. Cherkaoui, V. Fischer, A. Aubert, and L. Fesquet, "A self-timed ring based true random number generator," in *Proceedings of 2013 IEEE 19th International Symposium on Asynchronous Circuits Systems (ASYNC)* (IEEE, Santa Monica, CA, USA, 2013), pp. 99–106.
[27]G. Gimenez, A. Cherkaoui, R. Frisch, and L. Fesquet, "Self-timed ring based true random number generator: Threat model and countermeasures," in *Proceedings of 2nd International IEEE Verification Security Workshop (IVSW)* (IEEE, Thessaloniki, Greece, 2017), pp. 31–38.
[28]M. Tomamichel, C. Schaffner, A. Smith, and R. Renner, "Leftover hashing against quantum side information," IEEE Trans. Inf. Theory **57**, 5524–5535 (2011).
[29]H. Martin, P. Peris-Lopez, J. E. Tapiador, and E. San Millan, "A new TRNG based on coherent sampling with self-timed rings," IEEE Trans. Ind. Inform. **12**, 91–100 (2016).
[30]B. Valtchanov, V. Fischer, and A. Aubert, "Enhanced TRNG based on the coherent sampling," in *2009 3rd International Conference on Signals, Circuits and Systems (SCS)* (IEEE, Medenine, Tunisia, 2009), pp. 1–6.
[31]A. P. Johnson, R. S. Chakraborty, and D. Mukhopadyay, "An improved DCM-based tunable true random number generator for Xilinx FPGA," IEEE Trans. Circuits Syst. II **64**, 452–456 (2017).
[32]M. Matsumoto and T. Nishimura, "Mersenne twister: A 623-dimensionally equidistributed uniform pseudo-random number generator," ACM Trans. Model. Comput. Simul. **8**, 3–30 (1998).
[33]M. Bakiri, J.-F. Couchot, and C. Guyeux, "Ciprng: A VLSI family of chaotic iterations post-processings for linear pseudorandom number generation based on Zynq MPSoC," IEEE Trans. Circuits Syst. I **65**, 1628–1641 (2018).
[34]D. Knuth, "Deciphering a linear congruential encryption," IEEE Trans. Inf. Theory **31**, 49–52 (1985).
[35]G. Marsaglia and A. Zaman, "A new class of random number generators," Ann. App. Probab. **1**, 462–480 (1991).
[36]B. L. Holian, O. E. Percus, T. T. Warnock, and P. A. Whitlock, "Pseudorandom number generator for massively parallel molecular-dynamics simulations," Phys. Rev. E **50**, 1607–1615 (1994).
[37]K. V. Tretiakov and K. W. Wojciechowski, "Efficient Monte Carlo simulations using a shuffled nested Weyl sequence random number generator," Phys. Rev. E **60**, 7626–7628 (1999).
[38]K. W. Wojciechowski, "Pseudorandom number generators based on the Weyl sequence," Comput. Methods Sci. Technol. **5**, 81–85 (1999).
[39]M. Herrero-Collantes and J. C. Garcia-Escartin, "Quantum random number generators," Rev. Mod. Phys. **89**, 015004 (2017).
[40]D. B. Thomas and W. Luk, "The LUT-SR family of uniform random number generators for FPGA architectures," IEEE Trans. VLSI Syst. **21**, 761–770 (2013).
[41]S. Wu, J. Jiang, and Y. Fu, "Hardware architecture for the parallel generation of long-period random numbers using MT method," in *Computer Engineering and Technology* (Springer Berlin Heidelberg, Berlin, Heidelberg, 2013), pp. 8–15.