Received September 24, 2021, accepted October 6, 2021, date of publication October 13, 2021, date of current version October 20, 2021.

*Digital Object Identifier 10.1109/ACCESS.2021.3119593*

# A Dual Entropy-Based Digital Random Number Generator

**HİKMET SEHA ÖZTÜRK AND SALİH ERGÜN, (Member, IEEE)**
TUBITAK Informatics and Information Security Research Center, 41470 Gebze, Kocaeli, Turkey
Corresponding author: Salih Ergün (salih.ergun@tubitak.gov.tr)

**ABSTRACT** This paper introduces the dual-entropy method for oscillator-based digital random number generators (RNG). The standard model for elaborating jitter-based RNG is expanded to account for the sampling time uncertainty. It is explained that combining the jitter of the sampling clock with the uncertainty of the sampled signal proves to be the most efficient method for harvesting entropy from jitter. Dual-entropy configuration also improves the robustness of the RNG against correlation, interference, and injection-based attacks in hardware. Numerical analysis and experimental results show that the negative effect of external interference on the entropy of the output bitstream is significantly reduced in dual-entropy-based RNG. The proposed method is demonstrated on a ring-oscillator-based design and implemented on FPGA. It is tested using NIST 800-22, Dieharder, and TestU01 batteries and observed to produce successful random bitstreams at a maximum center frequency of 45 MHz on Zedboard Zynq and 114 MHz on ZCU102 Zynq Ultrascale+ evaluation kits without post-processing.

**INDEX TERMS** Dual entropy, interference, irregular sampling, ring oscillators, random number generators.

## I. INTRODUCTION

Information security has become more important than ever with the digitalization of the world. Modern cryptographic systems used in computation, commerce, and communications are constructed in compliance with Kerchoff's principle: The security of a cryptosystem should solely rely on the secret key [1]. Unpredictability of the secret key is therefore of utmost importance, which makes the design of a reliable and high-quality Random Number Generators (RNG) an interesting research topic.

RNGs are the hardware modules that exploit natural entropy sources to generate random bits. These entropy sources include thermal and shot noise in circuitry [2], [3], metastability [4], radioactive decay [5] and even quantum mechanical uncertainty [6]. These entropy sources can be processed using analog or digital circuitry. Despite the wide range of electrical components and customization available in analog design, digital RNGs have been gaining popularity for the last decade due to their simplicity in design and prototyping, thanks to programmable logic solutions. Digital RNGs are also easier to integrate into digital systems, both during the operation and during the design cycle.

The associate editor coordinating the review of this manuscript and approving it for publication was Cesar Vargas-Rosales.

Most digital RNGs use metastability, thermal noise, or jitter in the circuit as the entropy source. The simple and most common implementation utilizes multiple ring oscillators (RO), where a ring oscillator is an odd number of inverters cascaded to form a loop [7]. Each RO has a jitter due to uncertainty in signal transition, where sampling the signal within this interval yields a random bit. RO outputs are then XORed to combine uncertainties of various rings, returning a final single random bit. Another type of circuit uses chaotic feedback systems instead of simple ring oscillators, where even if the sampled signal is not captured around a transition, the value is still unknown due to chaotic behavior. [8] and [9] used Fibonacci and Galois ring oscillator structures, where [10] uses a chaotic matrix to generate the oscillating signal. These RNGs fundamentally employ the same idea of sampling an uncertain signal with a clock signal, hence referred to as "regular sampling of irregular waveform".

Another approach is to use the entropy source to have the sampling time uncertain instead of the data itself, which results in a similar outcome. [8] and [11] demonstrated this concept by achieving random bitstreams by sampling a clock signal at uncertain occasions. It is also explained in [11] that having the sampling signal irregular helps reduce the negative effect caused by the coupling of close inverters on entropy. The method used in these studies is referred to as "irregular sampling of regular waveform".

In this paper, we combine these approaches and propose a dual-entropy-based RNG, which can be referred to as "irregular sampling of irregular waveform". We start by expanding the currently accepted model of RO-based RNGs to account for the jitter of the sampling clock and explain how this can be exploited to increase entropy with the same resources. We then carry a comparative numerical analysis on interference vulnerability between regular sampling and the proposed dual-entropy methods. The interference resistance of the proposed method is further verified with the experimental test results. Finally, we compare the maximum sampling frequencies and show that the proposed method can achieve higher throughput than regular sampling and irregular sampling of regular waveform methods.

The organization of the paper is as follows. Section 2 briefly explains the theory behind RO RNG with and without the sampling time uncertainty, then discusses the dual-entropy method within the same model. Implementation details are presented in section 3. Section 4 explains the numerical analysis of interference and the corresponding experimental test setup. Statistical test results are given in section 5. Security aspects and limitations of the proposed RNG are discussed in section 6. Finally, the conclusions are summarized in section 7.

## II. THEORETICAL BACKGROUND

The randomness of the ring-oscillator-based RNG relies on the variance of the oscillation period of the rings due to electrical noise. This is referred to as period jitter, and it is the physical entropy source harvested by the circuit. XOR gate is used to combine multiple RO outputs, which combines the uncertainty intervals of ROs. Sampling the output of XOR yields a random bit if the sampling occurs during an uncertainty interval. If the sampling is done regularly with a period of $T_{clk}$, the sampling time of each bit with respect to the RO period is deterministic and can be calculated as

$$t_{clk} = T_{clk}(m+1) + \phi_{clk} \bmod T_{RO}$$

where $m$ is the number of previous samples. It is easy to recognize that if the sampling period $T_{clk}$ and RO period $T_{RO}$ are relatively prime, i.e., RO and sampling clock are not fully or partially synchronous, the sampling time eventually iterates through all of the RO period. The ratio of total uncertainty interval to the RO period is therefore equal to the ratio of random bits in the final bitstream. The design question, therefore, becomes "how many ROs we need to put before we can expect output bitstream to be completely random?".

### *A. THE URN MODEL*

Sunar *et al.* [12] tackles this question by converting it into a combinatorial problem using the urn model from probability theory. The period of the ring oscillator is divided into N equal length "urns", each urn $J_m$ representing the interval $[(m-1)\frac{T_{RO}}{N}, m\frac{T_{RO}}{N}]$ for $m = 1, \ldots, N$. The transition region of each ring oscillator is assumed to randomly fill one of these urns depending on its phase, as experimental data showed that phase of independent ROs are uniformly distributed [13].

As [12] explained, the calculation of the number of ROs expected to fill all the urns is analogous to the Coupon Collector's problem if the rings are imagined to be added into the circuit one by one. At an arbitrary point of ring addition process when $r$ out of $N$ urns are filled, the probability of the next RO jitter hitting an empty urn is $\frac{N-r}{N}$. This makes the expected number of ring additions to fill the next urn $\frac{N}{N-r}$. Using this relation, the expected number of rings required to fill $N$ urns from the beginning is found as

$$n = \sum_{r=0}^{N-1} \frac{N}{N-r} = \sum_{r=1}^{N} N\frac{1}{r} \approx NlogN. \tag{1}$$

This results in an impractical amount of rings, even for a moderate number of urns such as $N = 50$, because it gets significantly more challenging to fill remaining urns as there are fewer empty urns left.

A solution is to aim for a fill rate lower than 100% and compensate the remaining bits with post-processing. The same study [12] indirectly determines the required ring number $n$ as a function of total urn number $N$, urn fill rate $f$, and confidence level $p$.

Without going into details of the numbers, we observe that even in the case of a reduced fill rate, the number of rings needed increases approximately linearly with $N$ in [12]. Our approach is to effectively reduce the number $N$ by combining the circuit with a second entropy source to reduce $n$. Before explaining our method, we need to explore how the width, and hence the number, of the urns are calculated.

### *B. CALCULATION OF URN WIDTH*

For a given ring oscillator $RO_m$, the probability density function (PDF) of its periods can be expressed with the normal distribution

$$P_{RO}(t-\phi_{RO_m}) = \mathcal{N}(\phi_{RO_m}, \sigma_{RO}^2) = \frac{1}{\sigma_{RO}\sqrt{2\pi}} e^{\frac{-(t-\phi_{RO_m})^2}{2\sigma_{RO}^2}} \tag{2}$$

This assumption model the phase jitter as the physical entropy source in the RNG as shown in Fig. 1b. For the rising RO transition given in Fig. 1a, the sampled bit will be 1 if the transition happens before the rising edge of the sampling clock. If we denote the sampling time as $\phi_{RO} + k$, the probability of sampling a one can be expressed as $P(\phi_{RO}+k > T_{RO})$, which is found by the cumulative distribution function (CDF)

$$p = \frac{1}{2}[1 + \text{erf}(\frac{k}{\sigma_{RO}\sqrt{2}})] \tag{3}$$

where $\text{erf}(t)$ denotes the Gauss error function. The amount of entropy harvested from this sampled bit can be calculated with the binary entropy function

$$H(p) = p\log_2 p - (1-p)\log_2(1-p). \tag{4}$$

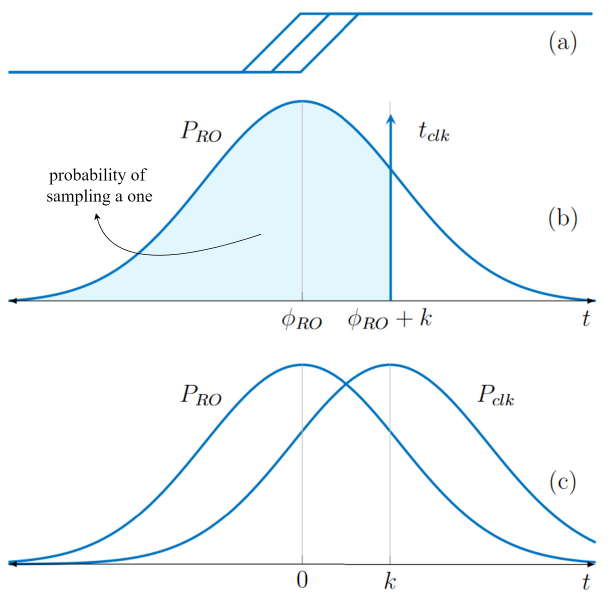

**FIGURE 1.** **(a) RO transition uncertainty. (b) Probability distribution with deterministic sample time. (c) Inclusion of sampling uncertainty.**

For the calculation of the urn width, a target entropy $E$ per sampled bit should be selected by the designer. The corresponding probability interval $[p, 1-p]$ is calculated using the inverse of (4). The interval of $k$ yielding the target probability for sampling a zero is found as $[-Q(p), Q(p)]$ using the Gaussian quantile function

$$Q(p) = \sigma_{RO}\sqrt{2}\,\text{erf}^{-1}(2p-1). \tag{5}$$

Equation (5) shows that urn width $2w = 2\,Q(p)$ depends on the target entropy and the standard deviation of the RO period, which depends on the number of inverters and the physical attributes of an inverter.

Notice that the calculation of (5) has been made possible by the following assumptions: The arrival time of the RO posedge shows a Gaussian distribution, whereas the sampling time is a calculable deterministic value. While the former is well-known to be supported by experimental data [3], [14], the latter is not entirely accurate. We argue that the sampling clock should also be represented as a probabilistic distribution since it is also an output of an oscillator somewhere and inevitably have some jitter, which should be accounted for.

## C. THE EFFECT OF SAMPLING JITTER

Fig. 1c shows the modified sampling process. The phase $\phi_{RO}$ is ignored for simplicity of calculation. The sample time and the arrival time of the RO posedge are defined as

$$t_{RO} \sim P_{RO}(t) = \mathcal{N}(0,\, \sigma_{RO}^2), \text{ and}$$
$$t_{clk} \sim P_{clk}(t) = \mathcal{N}(k,\, \sigma_{clk}^2)$$

respectively. The probability of sampling a zero can now be expressed as:

$$P(t_{clk} < t_{RO}) = P(t_{RO} = t') \times P(t_{clk} < t') \quad \text{for } \forall t' \in \mathbb{R}$$

We know that $P(t_{clk} < t')$ for any given $t'$ is the CDF expressed in (3). Integrating this through all possible $t_{RO} = t'$ values we get:

$$p = \int_{-\infty}^{\infty} P_{clk}(t')[\int_{-\infty}^{t'} P_{RO}(t)\,dt]\,dt' \tag{6}$$
$$= \int_{-\infty}^{\infty} \frac{1}{\sigma_{RO}\sqrt{2\pi}} e^{\frac{-(t'-k)^2}{2\sigma_{RO}^2}} \frac{1}{2}[1+\text{erf}(\frac{-t'}{\sigma_{clk}\sqrt{2}})]\,dt' \tag{7}$$

Using the relation given in [15] for integral of a Gaussian multiplied by an error function, (6) can be re-written as

$$p = \frac{1}{2}[1+\text{erf}(\frac{k}{\sqrt{2(\sigma_{clk}^2+\sigma_{RO}^2)}})]. \tag{8}$$

Similarly, the half urn-width is found

$$Q(p) = \sqrt{2(\sigma_{RO}^2+\sigma_{clk}^2)}\,\text{erf}^{-1}(2p-1). \tag{9}$$

Equations (8) and (9) prove that the sampling time uncertainty is equally effective as the uncertainty of the sampled signal. The value $p$ approaches to $\frac{1}{2}$ as the variance of the distributions rise, resulting in increased entropy per bit $E = H(p)$. The number of urns $N = \frac{T_{RO}}{2w}$ effectively reduces as $\sigma_{clk}$ increases, which increases $w = Q(p)$. Notice that (9) is equivalent to (5) for $\sigma_{clk} = 0$, in which case the PDF $P_{clk}$ approaches to the dirac delta function $\delta(t-k)$.

To quantify the impact of the irregular clock, we carry a small analysis for the case of $\sigma_{RO} = \frac{T}{50}$. Fig. 2a and 2b show the change in urn width and the total number of urns as the jitter of the sampling clock increases. Fig. 2c shows the entropy harvested per bit if the total number of urns and ring oscillators are kept constant as the sampling clock uncertainty increases.

## D. INCREASING THE SAMPLING JITTER

From a practical perspective, the sampling clock, e.g., the system clock in a SoC, rarely has as much uncertainty as a RO. These clocks usually have correcting mechanisms, such as negative feedback in phase-locked loops (PLL) [16], which prevents jitter from accumulating [17]. For example, a longer than usual period is more likely to be followed by a shorter period due to phase correction, introducing a correlation between sampling time of adjacent bits.

Our idea is to effectively increase the urn width by using ring oscillators with increased jitter to generate the sampling signal. Without the correction feedback, the jitter of each period in a free-running ring oscillator permanently accumulates into all of the succeeding periods [18]. The absolute jitter accumulated over $l$ periods can be expressed as [19]:

$$\sigma_{abs}^2 = \sum_{j=1}^{l} \sigma_{RO}^2 = l\sigma_{RO}^2 \tag{10}$$

To allow ring oscillators to accumulate jitter after each sampling without reducing the sampling frequency, we employ multiple ROs and use their rising edges in an

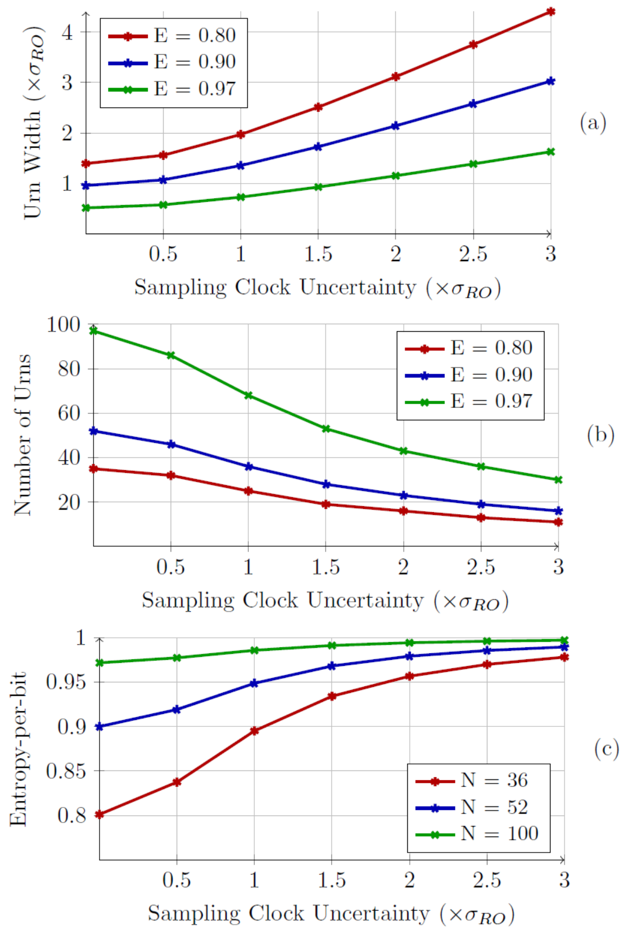

**FIGURE 2.** Change of (a) Urn width, (b) number of urns, (c) entropy-per-bit, with the sampling clock uncertainty.

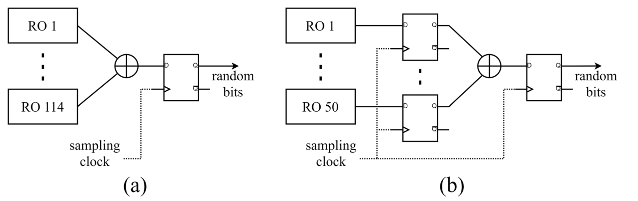

**FIGURE 3.** (a) RO-based RNG core introduced by Sunar *et al.* [12], (b) improved version proposed by Wold and Tan [7].

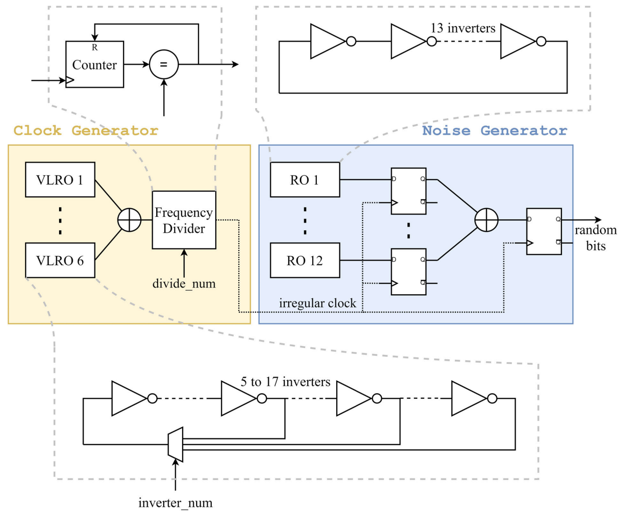

**FIGURE 4.** Proposed RNG architecture consisting of noise generator and clock generator parts.

interleaving fashion. This is accomplished by XORing $p$ number of ring oscillators, followed by a frequency division by $d$, shown as the clock generator part in Fig. 4. The XOR combines all of the transitions in the underlying rings, whereas the divider module allows only one out of $d$ posedges to propagate. The RO number $p$ being relatively prime with division amount $d$ ensures each ring is used once every $d$ cycles. As a result, the improvement shown in Fig. 2 can be accomplished with only the addition of a couple of ring oscillators and a divider.

## III. IMPLEMENTATION DETAILS

Conventional RO-based RNG core is depicted in Fig. 3. Fig. 3a shows the original circuit introduced by Sunar *et al.* [12], and Fig. 3b shows the improved version proposed by Wold and Tan [7]. The proposed RNG using dual-entropy configuration is shown in Fig. 4. For the noise generator part, flip flops have been used before combining the RO outputs as proposed by [7]. This has been proved to be mathematically equivalent to directly XORing in [20] while performing better in physical implementations. Implementation is carried on two different boards, Zedboard Zynq and Zynq UltraScale+ ZCU102 evaluation kits, using VHDL language on Xilinx Vivado Design Suite. The designed circuit mainly consists of two distinct sets of ring oscillators: One to generate the random data identical to conventional RO-based RNGs, and the other to generate the irregular clock for sampling the generated irregular waveform. For simplicity, we refer to these parts as noise and clock generators, respectively.

The complete setup used during data acquisition consists of a data transfer circuit along with the RNG, as shown in Fig. 5. The noise generator consists of 12 ring oscillators with 13 inverters each. The outputs of the rings are sampled separately with the same irregular clock before being merged with an XOR tree to achieve a single random bit. The clock generator part consists of 6 rings whose outputs are XORed directly. Variable-length ring oscillators (VLRO) are used in this part instead of standard ROs due to experimental reasons, namely because they allow the modification of the oscillation frequency. The number of active inverters is controlled by the signal *inverter_num*. The frequency divider increases the control over the sampling frequency by allowing only one in every *divide_num* rising edges to propagate. This also causes RO transitions to be selected in an interleaving fashion, allowing them to accumulate jitter between each selection. Both *inverter_num* and *divide_num* signals are controlled by the

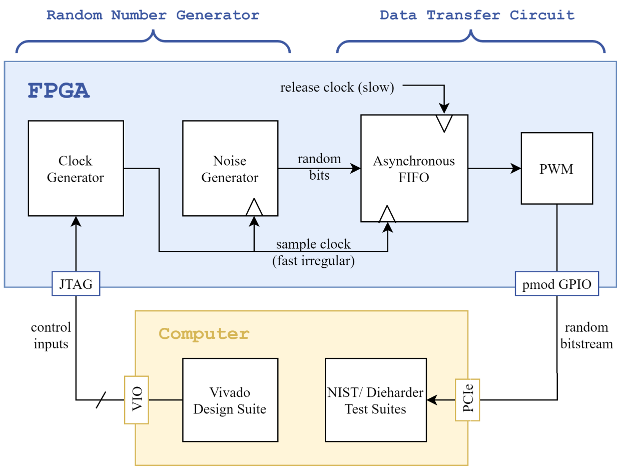

**FIGURE 5.** **Complete test setup used during data acquisition.**

Vivado Virtual I/O (VIO) interface, and each combination of the two results in a different average sampling frequency.

Inverters and other logic gates used in this RNG are instantiated as physical look-up table (LUT) primitives to prevent logic optimizations that may change the circuit. Each intended instance is also constrained to avoid getting affected by any redundancy removal and allow combinatorial loops necessary for the RNG with attribute specification [21]. The RNG implementation uses 288 LUTs and 18 FFs on Zedboard Zynq and a single LUT/FF pair more on the ZCU102 due to the increased size of the divider unit.

Data is outputted from the pmod GPIO on the FPGA board and captured from the Peripheral Component Interconnect Express (PCIe) on the computer. The PCIe link used in our setup allows data transmission in the form of pulse-width modulation with a throughput of 6.25Mbps. We use an asynchronous FIFO to manage data collection at high speed. FIFO uses FPGA BRAM resources to store 1 million consecutive bits with the fast irregular clock, which is the amount of data NIST 800-22 tests use for each block. The stored bits are then released with a 6.25 MHz clock and transferred to the computer, and the process is repeated for more data. PWM module handles the encoding necessary for data transfer only and does not modify the bitstream.

## IV. INTERFERENCE ANALYSIS

Acar and Ergün [11] explained that correlation could be found between the data collected from adjacent ring oscillators, making it possible to apply cryptanalysis by attaching an attack circuit. Similarly, signals passing through the same or adjacent logic blocks on FPGA affecting each other can reduce the quality of random bits. For example, a periodic signal such as the system clock being coupled to the RNG circuit can increase certain periodic components in the power spectrum of RNG output, consequently reducing the irregularity.

We first investigate how the interference from an external oscillator affects the randomness of our bitstreams via numerical analysis and compare the dual-entropy and regular sampling methods. Then we proceed to implement a separate test setup to add interference, allowing this comparison to be made experimentally on FPGA.

The concept of approximate entropy (ApEn) is used for the quantification of entropy since the computation of Shannon Entropy is not practically possible as it requires a bitstream of infinite length [22]. A greater ApEn value indicates a higher level of irregularity, which approaches $ln(2) \approx 0.69$ for a perfectly random sequence.

### A. NUMERICAL ANALYSIS

We start by obtaining the waveforms of the ring oscillators in both noise generator and clock generator by routing each to an FPGA output and capturing from an oscilloscope with a sampling rate of 10 GS/s.

For the regular sampling method, each noise generator ring is converted to binary at 40 MHz and XORed with each other to generate random bits. To generate bitstream with the dual-entropy method, the irregular clock should be generated from the clock generator rings, which is done by XORing the unsampled RO outputs. This is done numerically instead of sampling the combined signal from an FPGA output because it allows the addition of interference into ring oscillators individually. Also, the combined signal is irregular and includes many high-frequency components, which would have likely been filtered out at the FPGA pins due to the limited driving capabilities of IO buffers.

In [23], the analog transfer function for a LUT6 instance used as a two-input xor gate is modeled as:

$$xor2(v_i, v_j) = x_i\bar{x}_j + x_j\bar{x}_i \tag{11}$$

where $x_i$ and $\bar{x}_i$ are defined as

$$x_i = \frac{1}{1+e^{-a(v_i-b)}}, \tag{12}$$

$$\bar{x}_i = \frac{1}{1+e^{a(v_i-b)}}. \tag{13}$$

Ring oscillators are combined by repeatedly applying $xor2$ operation with parameters $a = 30$ and $b = 0.5$ to get the irregular clock waveform. Rising edges of this irregular clock are detected and downsampled to simulate the divide unit in the circuit. The division amount is selected as the integer value that yields the average sampling frequency closest to 40 MHz in order to allow fair comparison. The remaining posedges are used to sample the noise generator rings and convert their outputs to binaries, which are merged using the $xor2$ function.

Bitstreams of length 20k are obtained for both methods, and their approximate entropy is calculated using MATLAB. Both methods yield ApEn values around $ln(2)$. To simulate the impact of interference on our RNG, we add a sinusoidal to each of the ring oscillator waveforms and recompute approximate entropy for the newly generated bitstreams. Fig. 6 shows

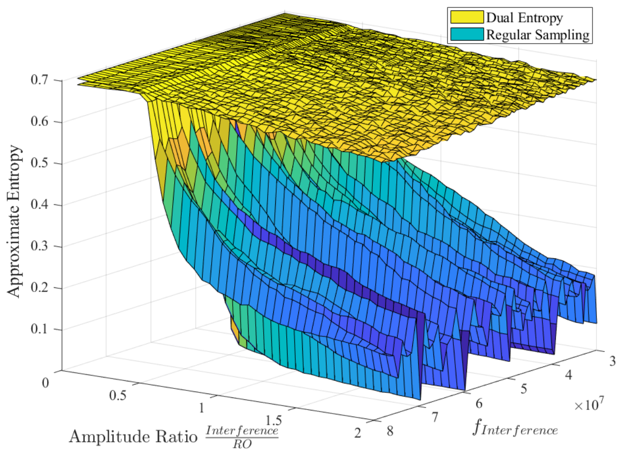

**FIGURE 6.** Effect of interference on ApEn for regularly and irregularly sampled bitstreams.

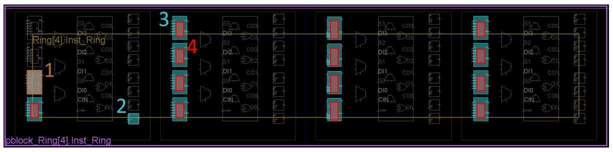

**FIGURE 7.** FPGA placement of a ring oscillator with external interference.

the change of ApEn values with the amplitude and frequency of the added interference. The entropy of the regularly sampled bitstream diminishes rapidly as the amplitude of the interference increases, whereas the entropy of the irregularly sampled bitstream only shows a slight decline. Furthermore, irregular sampling avoids the sudden entropy drops when the sampling frequency is equal to the harmonics of the interference.

### B. TEST SETUP

For the interference analysis, we aim to cause coupling between an external signal and the ring oscillators by repeatedly passing them through the same logic blocks. Each inverter in ring oscillators is replaced with LUT_6_2 primitives that correspond to two inverters in parallel. One inverter of each instance is used to form the ring oscillator loop, whereas the other inverter provides a path for the clock signal to pass close enough to the rings so that it can cause coupling. The interfering clock is run at 100 MHz, which is reasonably fast to expect interference and also close to the sampling frequencies so that its effect can be observed. The buffer instances in which the RO outputs propagate within the FPGA are also modified to accommodate two parallel buffers, allowing the clock signal to propagate along. Fig. 7 shows the FPGA placement of a modified ring, where unit 1 is a bypass MUX for frequency analysis, 2 is the DFF at the output of the ring, 3 is one of 13 inverters in the ring, and 4 is one of the extra inverters in which the interfering clock signal passes through. Interference analysis is carried on Zedboard Zynq evaluation kit.

## V. MEASUREMENT AND TEST RESULTS

The frequency spectrum of the noise generator output is analyzed before data collection, and it is observed to resemble a noise signal band-limited at 67 MHz on Zedboard. The autocorrelation value for a flatband noise with bandwidth $B$ is known to be at its lowest value when the sampling frequency is a divisor of $2B$ [24]. By iterating with the *inverter_num* and *divide_num* parameters, we adjust the irregular clock and obtain a 45 MHz average sampling frequency. 40 MB of data collected with these parameters in blocks of 1 million bits and subjected to NIST 800-22 tests [25]. The same amount is used for the TestU01 batteries [26]. For the Dieharder tests [27], a bitstream of 8GB is achieved by combining multiple takes from the same RNG. Data collection is carried similarly for the ZCU102 implementation with an average sampling frequency of 114 MHz. Results are presented in Tables 1 and 2 respectively and show that both implementations passed both NIST and Dieharder tests. The same bitstreams also passed all tests under Alphabit, Rabbit, Block Alphabit, and SmallCrush batteries of TestU01.

Interference resistance is also measured using the test setup explained in the previous section. As explained in Section 4, ApEn is used for the evaluation of entropy. NIST test suite generates an ApEn p-value for each bitstream using the underlying ApEn test, and a final singular p-value is presented in test results via the application of a chi-square test. Higher p-values indicate a more uniform distribution of underlying p-values and a stronger irregularity [25].

Four different sets of data are collected in each frequency range, where the variables are the type of the sampling clock and the existence of interference. Table 3 shows the resulting ApEn p-values. It is observed that the existence of interference introduces enough regularity to the noise signal so that the ApEn p-value reduces by one order of magnitude on average, even dropping below the 0.01 threshold. On the other hand, the ApEn values in the bit sequences obtained by the irregular sampling method show only slight variation, showing that the interference is not nearly effective.

Note that it is generally not easy to observe interference on FPGAs due to the relatively isolated placement of logic blocks and instances. In fact, interference analysis had to be repeated numerous times with different implementations of the same circuit before a reasonable amount of interference is observed at the output using the method shown in Fig. 7.

Next, we compare the proposed method with the regular sampling of irregular waveform by changing the sampling method on the same implementation and comparing throughputs. The sampling frequency is gradually increased in both methods, 40 MB of data collected at each frequency level and subjected to the NIST 800-22 and TestU01 batteries. The results presented in Table 4 show that the dual-entropy method has provided more than 100% increase in throughput for less than 70% area overhead in both implementations.

Finally, the proposed method is compared to the irregular sampling of regular waveform. While it offers similar benefits

**TABLE 1.** Statistical test results of the dual entropy based RNG implemented on Zedboard Zynq-7000 development board.

| NIST | | | | DIEHARDER | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **Statistical Test** | **P Values** | **Rate** | **P/F** | **Statistical Test** | **P Values** | **P/F** | **Statistical Test** | **P Values** | **P/F** |
| Frequency | 0.502986 | 319/320 | Pass | diehard_birthdays | 0.735452 | Pass | diehard_craps | 0.088-0.550 | Pass |
| Block Frequency | 0.425817 | 320/320 | Pass | diehard_operm5 | 0.762818 | Pass | mars_tsang_gcd | 0.025-0.378 | Pass |
| Cumulative Sums | 0.131500 | 319/320 | Pass | diehard_rank_32x32 | 0.612414 | Pass | sts_monobit | 0.915793 | Pass |
| Runs | 0.267238 | 325/320 | Pass | diehard_rank_6x8 | 0.833529 | Pass | sts_runs | 0.849293 | Pass |
| Longest Run | 0.915478 | 316/320 | Pass | diehard_bitstream | 0.058505 | Pass | sts_serial | 0.005-0.990 | Pass |
| Rank | 0.026835 | 315/320 | Pass | diehard_opso | 0.402855 | Pass | rgb_bitdist | 0.012-0.970 | Pass |
| FFT | 0.944403 | 316/320 | Pass | diehard_oqso | 0.525649 | Pass | rgb_min_distance | 0.202-0.915 | Pass |
| Nonoverlapping Temp. | 0.013509 | 317/320 | Pass | diehard_dna | 0.252556 | Pass | rgb_permutations | 0.293-0.949 | Pass |
| Overlapping Template | 0.115810 | 314/320 | Pass | diehard_count_1s_str | 0.305822 | Pass | rgb_lagged_sum | 0.029-0.999 | Pass[1] |
| Universal | 0.425817 | 319/320 | Pass | diehard_count_1s_byt | 0.560317 | Pass | rgb_kstest_test | 0.113467 | Pass |
| Approximate Entropy | 0.403403 | 318/320 | Pass | diehard_parking_lot | 0.569232 | Pass | dab_bytedistrib | 0.326449 | Pass |
| Random Excursions | 0.009535 | 201/204 | Pass | diehard_2dsphere | 0.193265 | Pass | dab_dct | 0.057497 | Pass |
| Random Exc. Variant | 0.191687 | 201/204 | Pass | diehard_3dsphere | 0.655598 | Pass | dab_filltree | 0.405-0.887 | Pass |
| Serial | 0.096217 | 318/320 | Pass | diehard_squeeze | 0.871061 | Pass | dab_filltree2 | 0.555-0.991 | Pass |
| Linear Complexity | 0.403403 | 317/320 | Pass | diehard_runs | 0.633-0.746 | Pass | dab_monobit2 | 0.961038 | Pass |

[1] 1 out of 33 tests returned a Weak p-value. Default pass threshold is $0.005 < p < 0.995$. For a perfectly random sequence, 1 out of 100 tests should return a Weak result. A single run of Dieharder consists of more than 100 tests.

**TABLE 2.** Statistical test results of the dual entropy-based RNG implemented on Zynq Ultrascale+ ZCU102 evaluation kit.

| NIST | | | | DIEHARDER | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **Statistical Test** | **P Values** | **Rate** | **P/F** | **Statistical Test** | **P Values** | **P/F** | **Statistical Test** | **P Values** | **P/F** |
| Frequency | 0.850337 | 319/320 | Pass | diehard_birthdays | 0.383128 | Pass | diehard_craps | 0.161-0.443 | Pass |
| Block Frequency | 0.345449 | 319/320 | Pass | diehard_operm5 | 0.709290 | Pass | mars_tsang_gcd | 0.366-0.881 | Pass |
| Cumulative Sums | 0.708280 | 319/320 | Pass | diehard_rank_32x32 | 0.734958 | Pass | sts_monobit | 0.529885 | Pass |
| Runs | 0.340461 | 317/320 | Pass | diehard_rank_6x8 | 0.732680 | Pass | sts_runs | 0.919888 | Pass |
| Longest Run | 0.099885 | 320/320 | Pass | diehard_bitstream | 0.781298 | Pass | sts_serial | 0.026-0.990 | Pass |
| Rank | 0.038187 | 317/320 | Pass | diehard_opso | 0.185784 | Pass | rgb_bitdist | 0.185-0.978 | Pass |
| FFT | 0.484646 | 318/320 | Pass | diehard_oqso | 0.990817 | Pass | rgb_min_distance | 0.011-0.641 | Pass |
| Nonoverlapping Temp. | 0.776784 | 312/320 | Pass | diehard_dna | 0.750906 | Pass | rgb_permutations | 0.392-0.641 | Pass |
| Overlapping Template | 0.959132 | 316/320 | Pass | diehard_count_1s_str | 0.005800 | Pass | rgb_lagged_sum | 0.028-0.990 | Pass |
| Universal | 0.437274 | 319/320 | Pass | diehard_count_1s_byt | 0.219274 | Pass | rgb_kstest_test | 0.120889 | Pass |
| Approximate Entropy | 0.885045 | 316/320 | Pass | diehard_parking_lot | 0.307485 | Pass | dab_bytedistrib | 0.777616 | Pass |
| Random Excursions | 0.334538 | 204/206 | Pass | diehard_2dsphere | 0.678042 | Pass | dab_dct | 0.139831 | Pass |
| Random Exc. Variant | 0.917870 | 202/206 | Pass | diehard_3dsphere | 0.992274 | Pass | dab_filltree | 0.666-0.940 | Pass |
| Serial | 0.758528 | 314/320 | Pass | diehard_squeeze | 0.084443 | Pass | dab_filltree2 | 0.404-0.713 | Pass |
| Linear Complexity | 0.663130 | 318/320 | Pass | diehard_runs | 0.073-0.103 | Pass | dab_monobit2 | 0.994910 | Pass |

**TABLE 3.** Interference vulnerability comparison between regular sampling and dual entropy methods.

| **Method** | **Sampling Frequency** | **p-values** | |
|---|---|---|---|
| | | Without Interference | With Interference |
| Regular Sampling | 33.3 MHz | 0.2548 | 0.0274 |
| | 31.25 MHz | 0.1017 | 0.0042 |
| | 25 MHz | 0.0618 | 0.0067 |
| Dual Entropy | ~30.7 MHz | 0.2630 | 0.1597 |
| | ~29.6 MHz | 0.2315 | 0.2589 |
| | ~25.8 MHz | 0.4606 | 0.3504 |

**TABLE 4.** Maximum operating frequency comparison between regular sampling and dual entropy methods.

| **FPGA** | **Method** | **Sampling Frequency** | **Test Results** | |
|---|---|---|---|---|
| | | | NIST 800-22 | TestU01 |
| Zedboard Zynq | Regular Smp. | 20 MHz | Pass | Pass |
| | Regular Smp. | 25 MHz | Pass | **Fail*** |
| | Dual Entropy | 45 MHz | Pass | Pass |
| ZCU102 Ultrascale+ | Regular Smp. | 50 MHz | Pass | Pass |
| | Regular Smp. | 60 MHz | Pass | **Fail*** |
| | Dual Entropy | 114 MHz | Pass | Pass |

* Multiple tests failed under Alphabit, Rabbit, BlockAlphabit

against interference, it is significantly behind in terms of throughput, mainly because the sampling frequency needs to be substantially lower than the frequency of a regular clock [11]. [11] features a similar RO-based RNG and has a throughput of 1.87 Mbps despite using over five times ROs. [28] uses fewer resources, but its bit generation speed is at only 273 kbps, less than 1/150 of the throughput of the proposed circuit on the same FPGA.

## VI. DISCUSSIONS

The circuit we proposed is a proof-of-concept for the dual-entropy method. We deliberately used a simple RO-based circuit in order to ensure that the performance of the RNG is coming from the intended physical entropy source and not from the pseudo-randomness of the complex harvesting mechanism. Similarly, post-processing is avoided to demonstrate that it is the raw RNG output showing sufficient statistical properties.

With that being said, even without post-processing, randomness tests alone should not be used as proof of a RNG. It has been shown that pseudo-randomness can be enough to pass statistical tests [20]. Although there is no such study targeting Dieharder, we use the passing point of tests solely for comparative purposes. For a security-critical application, increasing the number of rings or adding a known cryptographic post-processing can still be considered for an extra level of security. It is also possible to construct a ''provably secure'' version of this RNG such as [12] by combining the measurements from a specific technology with formulas provided in Section 2, and [12], but this is left as future work.

We do not recommend overly increasing the number of clock generator rings. Although the limits are not clear, having too many transitions in a period may cause them to overlap or start violating the setup/hold times of the registers or physical limitations of the XOR gate. While these do not necessarily reduce the randomness, their effects lie outside of the current theoretical justification.

The proposed dual-entropy method combines the entropy source with sampling time uncertainty, which is substantially different from other ''Dual Entropy Core'' RNGs in the literature such as [29]–[31]. They aim to increase the randomness by combining different entropy sources with a comparator or an XOR block but miss out on crucial security features acquired by employing irregular sampling. For example, on FPGA-based implementations, our method provides a secure backbone against correlation and interference-based attacks. Even if the noise source fails, or its behavior is determined by an attack circuit such as the one in [11], the data is not compromised as the sampling time is still uncertain. For the ASIC case, the proposed RNG can be included in digital design flow with less isolation and constraints than its counterparts due to its robustness, resulting in a more area-efficient design. Regardless of the implementation, it can provide a significant increase in throughput with the same number of noise generator rings. The need for an external clock is also removed, although this can be compensated by the resource used for the generation of the irregular clock.

Finally, the dual-entropy method provides protection against attacks targeting the power supply. Valtchanov *et al.* [19] pointed out the effect of accumulated manipulable global deterministic jitter on ROs and determined this as a potential attack vector. The same authors proposed generating the sampling clock inside the same FPGA as a countermeasure in [32], as in the case of dual-entropy. Both clock generator and the sampled signal are affected by the global jitter in a similar fashion, reducing the effectiveness of the attack [32].

## VII. CONCLUSION

This paper introduced the concept of dual-entropy on digital random number generators. The proposed concept is demonstrated on a ring-oscillator-based design and implemented on 28nm and 20nm Zynq FPGAs. Dieharder, NIST 800-22, and TestU01 batteries are used for the evaluation of the generated bitstreams. The effect of sampling time uncertainty is modeled for RO-jitter-based RNG. It is explained that the dual-entropy method provides a more efficient way of utilizing ring oscillators, achieving more than double throughput with less than 70% resource overhead while also eliminating the need for an external clock.

Robustness against external interference of the proposed method is verified through approximate entropy values in a comparative analysis. Dual-entropy configuration is also explained to provide a layer of security against correlation and injection-based attacks. Future work includes adapting the proposed method to other digital RNGs in the literature and the IC fabrication of the circuit. The benefits of the proposed method are expected to be better observed on-chip measurements due to increased jitter and interference.

## REFERENCES

[1] F. A. P. Petitcolas, *Kerckhoffs' Principle*. Boston, MA, USA: Springer, 2011, p. 675, doi: 10.1007/978-1-4419-5906-5_487.
[2] L. Gong, J. Zhang, H. Liu, L. Sang, and Y. Wang, ''True random number generators using electrical noise,'' *IEEE Access*, vol. 7, pp. 125796–125805, 2019.
[3] Y. Lu, H. Liang, L. Yao, X. Wang, H. Qi, M. Yi, C. Jiang, and Z. Huang, ''Jitter-quantizing-based TRNG robust against PVT variations,'' *IEEE Access*, vol. 8, pp. 108482–108490, 2020.
[4] C.-C. Wang and S.-W. Lu, ''100 MHz random number generator design using interleaved metastable NAND/NOR latches,'' in *Proc. IEEE Asia Pacific Conf. Circuits Syst. (APCCAS)*, Dec. 2020, pp. 98–101.
[5] M. Rohe, *Randy—A True-Random Generator Based on Radioactive Decay*. Saarbrücken, Germany: Saarland Univ., 2003, pp. 1–36.
[6] J. F. Dynes, Z. L. Yuan, A. W. Sharpe, and A. J. Shields, ''A high speed, postprocessing free, quantum random number generator,'' *Appl. Phys. Lett.*, vol. 93, no. 3, Jul. 2008, Art. no. 031109.
[7] K. Wold and C. H. Tan, ''Analysis and enhancement of random number generator in FPGA based on oscillator rings,'' in *Proc. Int. Conf. Reconfigurable Comput. (FPGAs)*, Dec. 2008, pp. 385–390.
[8] K. Demir and S. Ergun, ''Random number generators based on irregular sampling and Fibonacci–Galois ring oscillators,'' *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 66, no. 10, pp. 1718–1722, Oct. 2019.
[9] U. Güler, S. Ergün, and G. Dündar, ''A digital IC random number generator with logic gates only,'' in *Proc. IEEE Int. Conf. Electron., Circuits, Syst.*, Dec. 2010, pp. 239–242.
[10] Y. Yang, S. Jia, Y. Wang, S. Zhang, and C. Liu, ''A reliable true random number generator based on novel chaotic ring oscillator,'' in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2017, pp. 1–4.
[11] B. Acar and S. Ergun, ''Correlation-based cryptanalysis of a ring oscillator based random number generator,'' in *Proc. IEEE 61st Int. Midwest Symp. Circuits Syst. (MWSCAS)*, Aug. 2018, pp. 1050–1053.
[12] B. Sunar, W. J. Martin, and D. R. Stinson, ''A provably secure true random number generator with built-in tolerance to active attacks,'' *IEEE Trans. Comput.*, vol. 56, no. 1, pp. 109–119, Jan. 2007.
[13] W. R. Coppock and C. Philbrook, *A Mathematical and Physical Analysis of Circuit Jitter With Application to Cryptographic Random Bit Generation*. Worcester, MA, USA: Worcester Polytechnic Institute, Major Qualifying Project Report, 2005.
[14] M. A. Prada-Delgado, C. Martinez-Gomez, and I. Baturone, ''Auto-calibrated ring oscillator TRNG based on jitter accumulation,'' in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, Oct. 2020, pp. 1–4.
[15] E. W. Ng and M. Geller, ''A table of integrals of the error functions,'' *J. Res. Nat. Bur. Standards B. Math. Sci.*, vol. 73, no. 1, pp. 7–8, 1969.
[16] B. Razavi, *RF Microelectronics*, vol. 2. New York, NY, USA: Prentice-Hall, 2012, pp. 597–606.
[17] S. Robson, ''A ring oscillator based truly random number generator,'' M.S. thesis, Dept. Elect. Comput. Eng., Master Appl. Sci. Elect. Comput. Eng., Univ. Waterloo, Waterloo, ON, Canada, 2013.

[18] A. Hajimiri, S. Limotyrakis, and T. H. Lee, "Jitter and phase noise in ring oscillators," *IEEE J. Solid-State Circuits*, vol. 34, no. 6, pp. 790–804, Jun. 1999.

[19] B. Valtchanov, A. Aubert, F. Bernard, and V. Fischer, "Modeling and observing the jitter in ring oscillators implemented in FPGAs," in *Proc. 11th IEEE Workshop Design Diag. Electron. Circuits Syst.*, Apr. 2008, pp. 1–6.

[20] N. Bochard, F. Bernard, V. Fischer, and B. Valtchanov, "True-randomness and pseudo-randomness in ring oscillator-based true random number generators," *Int. J. Reconfigurable Comput.*, vol. 2010, pp. 1–13, Dec. 2010.

[21] Xilinx, Inc. (2021). *Vivado Design Suite User Guide: Synthesis UG901*. Accessed: Sep. 1, 2021. [Online]. Available: https://www.xilinx.com/content/dam/xilinx/support/documentation/sw_manu%als/xilinx2021_1/ug901-vivado-synthesis.pdf

[22] S. M. Pincus, "Approximate entropy as a measure of system complexity," *Proc. Nat. Acad. Sci. USA*, vol. 88, pp. 2297–2301, Mar. 1991. [Online]. Available: https://europepmc.org/articles/PMC51218

[23] T. Addabbo, A. Fort, R. Moretti, M. Mugnaini, H. Takaloo, and V. Vignoli, "A new class of digital circuits for the design of entropy sources in programmable logic," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 67, no. 7, pp. 2419–2430, Jul. 2020.

[24] K. Demir and S. Ergün, "Analysis of regular sampling of chaotic waveform and chaotic sampling of regular waveform for random number generation," *IEICE Trans. Fundam. Electron., Commun. Comput. Sci.*, vol. E102.A, no. 6, pp. 767–774, Jun. 2019, doi: 10.1587/transfun.E102.A.767.

[25] L. E. Bassham III, A. L. Rukhin, J. Soto, J. R. Nechvatal, M. E. Smid, E. B. Barker, S. D. Leigh, M. Levenson, M. Vangel, and D. L. Banks, *Sp 800-22 Rev. 1a. A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications*. Gaithersburg, MD, USA: National Institute of Standards & Technology, 2010.

[26] P. L'Ecuyer and R. Simard, "TestU01: AC library for empirical testing of random number generators," *ACM Trans. Math. Softw.*, vol. 33, no. 4, pp. 1–40, 2007.

[27] R. G. Brown. (2021). *Dieharder: A Random Number Test Suite*. Accessed: Jun. 24, 2021. [Online]. Available: https://webhome.phy.duke.edu/~rgb/General/dieharder.php

[28] R. Gunay and S. Ergun, "Irregular sampling of regular waveform based random number generator exploiting two simultaneous metastable events of tetrahedral oscillators," in *Proc. IEEE 63rd Int. Midwest Symp. Circuits Syst. (MWSCAS)*, Aug. 2020, pp. 615–618.

[29] I. Cicek, A. E. Pusane, and G. Dundar, "A new dual entropy core true random number generator," *Analog Integr. Circuits Signal Process.*, vol. 81, no. 1, pp. 61–70, 2014.

[30] I. Cicek, A. E. Pusane, and G. Dundar, "An integrated dual entropy core true random number generator," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 64, no. 3, pp. 329–333, Mar. 2017.

[31] I. Koyuncu, M. Tuna, I. Pehlivan, C. B. Fidan, and M. Alçın, "Design, FPGA implementation and statistical analysis of chaos-ring based dual entropy core true random number generator," *Anal. Integr. Circuits Signal Process.*, vol. 102, no. 2, pp. 445–456, Feb. 2020.

[32] V. Fischer, F. Bernard, N. Bochard, and M. Varchola, "Enhancing security of ring oscillator-based TRNG implemented in FPGA," in *Proc. Int. Conf. Field Program. Log. Appl.*, 2008, pp. 245–250.

**HİKMET SEHA ÖZTÜRK** received the B.Sc. degree in electronics and communication engineering from Istanbul Technical University, İstanbul, Turkey, in 2019. He is currently pursuing the M.Sc. degree in electrical and electronics engineering with Boğaziçi University, İstanbul. He has been a Researcher with TÜBİTAK BİLGEM, since 2019. His research interests include random number generators, hardware security, and cryptographic accelerators.

**SALİH ERGÜN** (Member, IEEE) received the B.Sc. and M.Sc. degrees in electronics and telecommunication engineering from Istanbul Technical University, İstanbul, Turkey, in 1998 and 2000, respectively, the Ph.D. degree in electrical engineering and information systems from The University of Tokyo, Tokyo, Japan, in 2011, and the postdoctoral degree from the Institute of Industrial Science, The University of Tokyo.

He then became an Associate Professor, in 2018. In 2000, he joined the National Research Institute of Electronics and Cryptology, TÜBİTAK, Turkey, where he was with the Cryptographic Hardware Group as the Manager. He has taken crucial roles in establishing research infrastructures of TÜTEL (IC Design and Education Laboratory) and prepared their feasibility and sustainability plans. In 2015, he founded the ERARGE Research and Development Center, İstanbul, and also the ERGTECH Research Center, Switzerland, to commercialize the productized devices for global markets. He has coordinated more than twenty national and international projects in various domains. Among these projects, the Collaborative City Co-Design Platform (C3PO) received the ITEA Award of Excellence 2018 for Business Impact and the Building Information Modelling in City (BIMy) received the ITEA Award of Excellence 2021 for Innovation. He is an inventor of various patents on security applications. He has published more than 100 papers in refereed journals and conference proceedings. His research interests include smart cyber-physical systems, industrial electronics, embedded systems, multimedia systems and applications, nonlinear systems and circuits, and information security systems, with special emphasis on software and hardware for cryptographic applications.

Dr. Ergün received the Best Paper Awards at the European Conference on Circuit Theory and Design (ECCTD) 2005 and the IEEE Asia Pacific Conference on Circuits and Systems (APCCAS) 2014, and the Excellent Paper Award at the Intelligent Information Hiding and Multimedia Signal Processing Conference (IIHMSP) 2017.