TROT: A Three-Edge Ring Oscillator Based True Random Number Generator With Time-to-Digital Conversion

Miloš Grujić and Ingrid Verbauwhede, *Fellow, IEEE*

***Abstract*— This paper introduces a new true random number generator (TRNG) based on a three-edge ring oscillator. Our design uses a new technique with a time-to-digital converter to effectively acquire jitter accumulated independently by each edge. As a part of the security evaluation, we present the stochastic model of the TRNG's digital noise source and estimate a lower bound of the min-entropy per random bit. Starting from the obtained entropy bound, we propose a procedure for selecting and implementing an area-efficient and throughput-optimal post-processing function based on the best known linear codes that will increase the output min-entropy rate to more than 0.999. The proposed TRNG exquisitely balances low design effort and resource consumption with high throughput and a high min-entropy rate, making it more suitable for randomness-demanding and resource-constrained platforms than the state-of-the-art. The complete implementation of the TRNG digital noise source and the post-processing occupies 33 slices and achieves a throughput of 12.5 Mbps on Xilinx Zynq-7000 FPGAs. The min-entropy of the generated random bits is assessed by NIST SP 800-90B entropy estimators, and the tested sequences pass the AIS-31 test suit.**

***Index Terms*— Entropy, hardware security, multimode ring oscillator, post-processing, stochastic model, true random number generator (TRNG).**

I. Introduction

RANDOM number generators are indispensable components of any modern security system. With physically unclonable functions (PUFs), true random number generators (TRNGs) are the only cryptographic primitives producing truly unpredictable bits for generating secrets in symmetric- and public-key cryptography. Random number generators are also extensively used in various randomization-based countermeasures for protecting cryptographic implementations against side-channel attacks (SCA). Among others, the low-latency masking schemes for countering the SCA incur a high area penalty, leaving only limited resources for random number generation [1]. These schemes also require many random bits per execution cycle that a TRNG often cannot provide. Therefore, they resort to faster pseudo-random number generators (PRNGs) to generate random masks [1]. Although producing statistically perfect random bits, the output of a cryptographically secure PRNG becomes entirely predictable once its inner state is leaked or guessed due to its deterministic nature. As it has recently been shown by De Meyer [2], to prevent side-channel leakage of a PRNG and avoid masking PRNG itself, its inner state should be refreshed with truly random bits from a TRNG much more frequently than previously required.

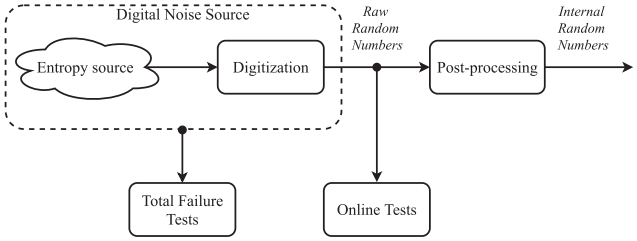

Fig. 1. Architecture of a modern true random number generator.

Due to their ubiquity in hardware security, the TRNGs have become subject to rigorous evaluations by the US [3], European [4] and Chinese [5] certification bodies. In addition to passing the statistical tests, modern TRNG designs should have an estimation of the amount of entropy they can provide. Moreover, the TRNG designers should provide a theoretical justification of the unpredictability of its output, as requested by the American NIST SP 800-90B standard [3]. On the other hand, TRNGs for cryptographic applications compliant with the AIS-31 standard [4] should also have a stochastic model from which a lower bound of the Shannon entropy can be estimated.

Fig. 1 illustrates a general TRNG architecture compatible with both the AIS-31 and the NIST SP 800-90B approaches. The entropy source is a component with nondeterministic behavior that exploits an inherently random physical process. The digitization module converts the output of the entropy source, which is often in analog form, into *raw random*

Manuscript received October 12, 2021; revised December 23, 2021 and February 7, 2022; accepted February 22, 2022. This work was supported in part by the CyberSecurity Research Flanders under Grant VR20192203, in part by the Research Council KU Leuven C1 under Contract C16/15/058, and in part by the European Commission through the Horizon 2020 Research and Innovation Program through QRANGE under Grant H2020-FETFLAG-2018-03-820405 and Cathedral European Research Council (ERC) Advanced under Grant 695305. This article was recommended by Associate Editor W. Liu. *(Corresponding author: Miloš Grujić.)*

The authors are with imec-COSIC, KU Leuven, 3001 Leuven, Belgium (e-mail: milos.grujic@esat.kuleuven.be; ingrid.verbauwhede@esat.kuleuven.be).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TCSI.2022.3158022.

Digital Object Identifier 10.1109/TCSI.2022.3158022

numbers (bits). The entropy source and the digitization module together constitute a digital noise source. The raw random numbers are usually not perfectly unpredictable and need a form of post-processing to increase the unpredictability, measured in entropy per bit, i.e., entropy rate. The random numbers that are output to the user application after the post-processing are called the *internal random numbers (bits)*. The raw random numbers undergo online tests during the whole operation to detect drops in the entropy and notify the user application. The operation of the digital noise source is closely monitored by the total failure tests, which can immediately detect malfunctioning of the digital noise source and prevent future TRNG outputs. According to the AIS-31, the average Shannon entropy rate of the internal random bits should be higher than 0.997 for the two highest TRNG security classes. On the other hand, NIST SP 800-90B allows claiming a min-entropy rate of the internal random bits of up to 0.999 when the arithmetic post-processing is used. Compared to the post-processing based on cryptographic hash functions or block ciphers, the arithmetic post-processing has the advantage of much lower resource consumption and provides information-theoretical security when used appropriately.

Although a plethora of TRNG designs have been put forward in the last 20 years, merely a handful of them possess a stochastic model and entropy estimation as a part of the security evaluation [6]–[14]. The throughput and min-entropy estimation of some of the TRNG designs without the stochastic model would likely significantly degrade once their stochastic model has been built. Most TRNG designs with the standard-compliant stochastic models are designed either for high throughput [7], [14] or low area [8], [9], [11]. Since modern TRNGs should not only be used for classical and post-quantum cryptographic applications, but also for providing randomness to the protection mechanisms against the SCA, there is a need for the designs with simultaneously high throughput and low area which are compliant with both AIS-31 and NIST SP 800-90B standards.

To address the lack of low area – high entropy TRNGs with NIST SP 800-90B and AIS-31 compliant stochastic models, this work introduces a new TRNG based on a three-edge ring oscillator with time-to-digital conversion – TROT. Thanks to both efficient entropy generation and optimized post-processing, the TROT offers one of the best area versus throughput trade-offs among previously reported FPGA-compatible TRNGs with a stochastic model. Its digital noise source couples a three-edge mode ring oscillator with a time-to-digital converter (TDC) in a way that enables efficient extraction of the independent white Gaussian jitter present in all three edges. We introduce the stochastic model of the TROT that accounts for the effects of the inherent hardware process variations and employ it to estimate the lower bound on the min-entropy contained in each raw random bit. To increase the min-entropy rate of the internal random bits, we apply the post-processing based on binary linear codes [15]. We propose a throughput optimization method for selecting adequate code and present a compact implementation of the post-processing architecture based on the generator matrix of the chosen code. The TROT produces internal random bits with a min-entropy rate of at least 0.999 to fulfill both the AIS-31 and the NIST SP 800-90B entropy requirements. We opted for the min-entropy instead of the Shannon entropy rate because the min-entropy is the most conservative unpredictability measure. We demonstrate the feasibility of our TRNG on a Xilinx Zynq FPGA device, where the TROT requires relative placement constraints. Our design passes all AIS-31 [4] statistical tests without any cryptographic post-processing.

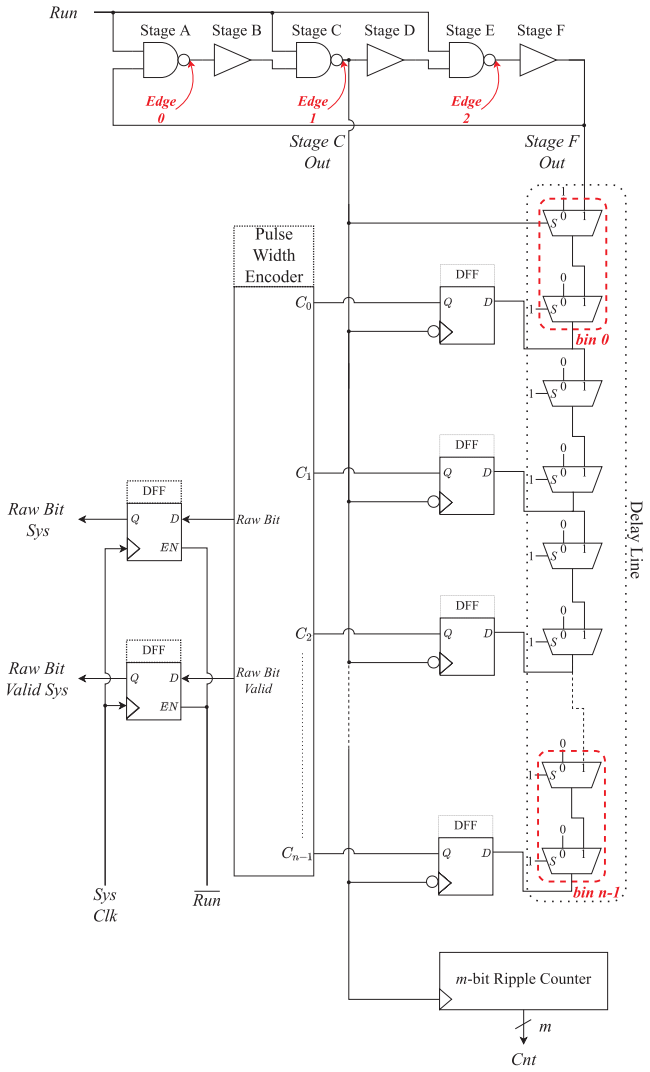

Fig. 2. TROT digital noise source and oscillation counter.

II. TROT Digital Noise Source

The entropy source of the proposed TRNG is a three-edge mode ring oscillator which consists of three inverting and three non-inverting stages, as depicted in the upper part of Fig. 2. Three edges are simultaneously injected by each inverting NAND stage with enable signal *Run*. These edges will have an identical mean period since they propagate through the same stages. The propagation through identical stages significantly reduces the influence of the global and possibly adversarial noise sources, while the local Gaussian noise accumulates independently by each edge. The frequency of the resulting

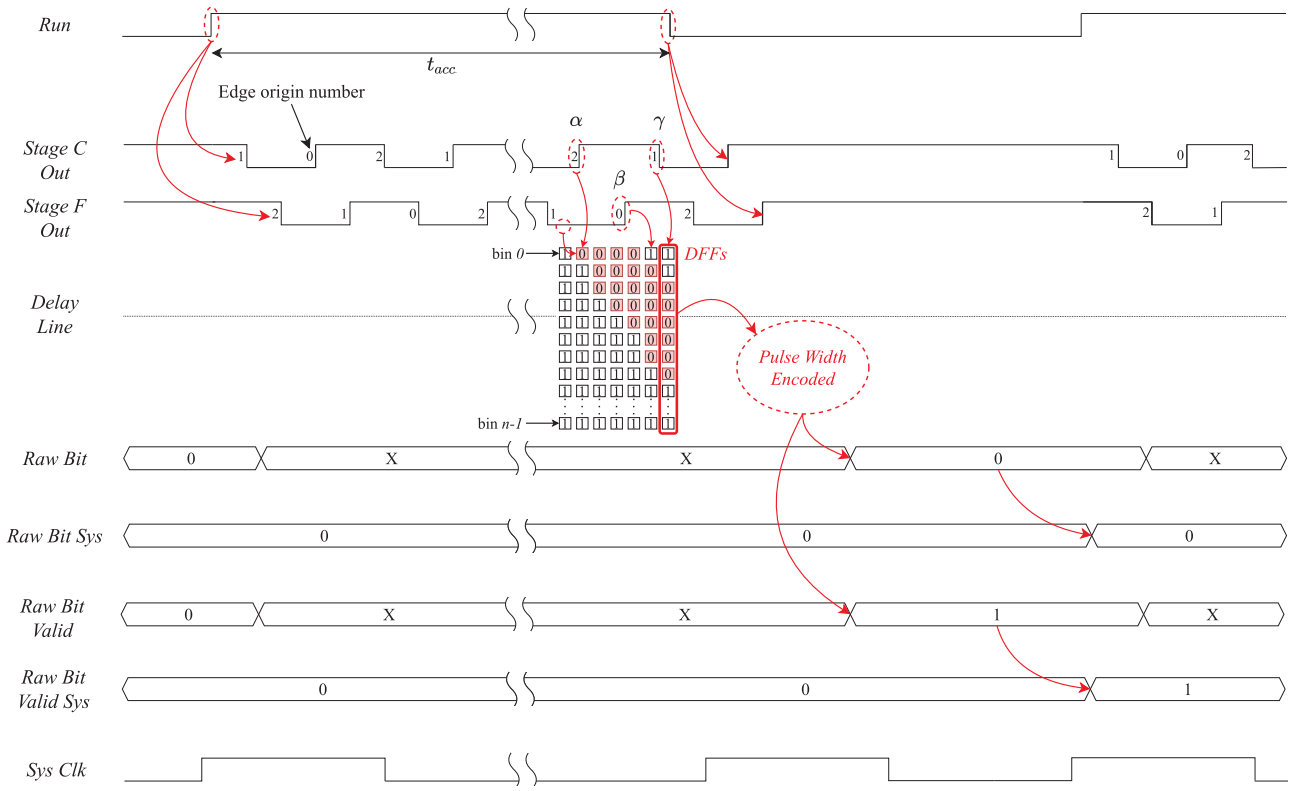

Fig. 3. Timing diagrams of the TROT digital noise source. In this example, the ordered edge triple (α, β, γ) corresponds to $(2, 0, 1)$.

ring oscillator signal at the output of any of the six stages will be three times higher than the frequency of a single-edge ring oscillator with the same stages. Due to noise influence, the three edges will inevitably collide, eventually compelling the ring oscillator to a single-edge mode.

A previously proposed TRNG based on the three-edge ring oscillator by Yang *et al.* [17] extracts randomness by counting the number of oscillations until the collapse of the three-edge mode. Unlike [17], we use a technique based on a single time-to-digital converter (TDC) line as a digitization element to more efficiently exploit the independent timing jitter in each edge. The use of TDC enables higher throughput since the randomness extraction from the three jittery edges can happen much earlier than waiting for the edge collapse. The TDC consists of a delay line formed by high-speed multiplexers and n falling-edge D flip-flops (DFFs) whose D inputs are connected to the outputs of the multiplexers – bins. As seen in Fig. 2, the signal from the third non-inverting stage F is connected to input 1 of the first multiplexer in the delay line. All multiplexers in the delay line have a select signal set to constant 1, except the first, whose select signal comes from the second inverting stage C. The signal from stage C is also used as a clock signal for the DFFs. This oscillator configuration thus falls into a category of multimode multi-phase oscillators, different from the previously proposed single-mode multi-phase oscillators [18], [19].

The pulse width encoder is a combinatorial circuit that outputs a raw bit value and a valid signal based on the DFFs' values $C_0, \ldots, C_{n-1}$. The raw bit depends on the parity of the number of zeros values relative to the total number of DFFs and can be logically represented with:

$$Raw\ Bit = \bigoplus_{i=0}^{n-1} C_i, \tag{1}$$

where n is always even. The raw bit valid signal is set when the first and the last DFF in the TDC have value one, and there is at least one DFF with value zero:

$$Raw\ Bit\ Valid = C_0 \wedge \left(\bigvee_{i=1}^{n-2} \neg C_i \right) \wedge C_{n-1}. \tag{2}$$

We use timing diagrams of the digital noise source in Fig. 3 to demonstrate how this configuration of the digital noise source enables capturing the timing jitter effects of all three edges. We denote with α the last rising edge and with γ the last falling edge at the output of stage C before the oscillations in the ring oscillator are disabled. Similarly, we denote with β the last rising edge at the output of stage F before disabling the ring oscillator. Edges α, β and γ thus correspond each to an edge of a different origin – 0, 1 or 2, as shown in Fig. 2. The ring oscillator is enabled for time period t_{acc} by setting the signal *Run* to 1. The value of t_{acc} is selected so that all three edges accumulate enough timing jitter during that time, as will be explained in Section III. The propagation of the signal at the output of stage F through the delay line is enabled by the edge α. Since the signals at the output of stages C and F have approximately 90° phase difference, the signal at the output of stage F will be low and bins in the delay line will start transitioning from 1 to 0, as indicated in Fig. 3. When the edge β appears at the output of stage F, the bins in the delay line that were previously set to 0 by

the edge α will start transitioning to 1 so that the delay line reflects the state of the signal at the output of stage F. Finally, the edge γ will trigger the corresponding DFFs to sample the state of the delay line. In this manner, the effects of all three edges α, β and γ will be captured by the DFFs. The sampled value of the DFFs is then encoded to the raw bit value and the raw bit validity value by the pulse width encoder according to (1) and (2), respectively. After t_{acc}, the ring oscillator is disabled for one clock cycle. This reset clock cycle ensures independence between consecutive bits and leaves enough time for the pulse width encoder to correctly evaluate the raw bit value and its validity. The ring oscillator is re-enabled on the following rising edge of the clock signal to produce the next random bit and transfer the previous bit and its validity to the system clock domain.

The unavoidable variations of the bins' propagation delays along the delay line will influence the amount of obtainable entropy [20]. While sampling the delay line, the setup or hold time of some DFFs in the TDC might be violated. This event leads to bubbles in the captured TDC code that can be dealt with by bin reordering [7], [21]. To improve the uniformity and minimize bubble manifestations in the TDC, we sacrifice the sampling precision by combining the two adjacent physical bins into one bin of the delay line.

To estimate the jitter accumulated by the three edges and their oscillation period in the design phase, we use an m-bit rising edge ripple counter connected to the output of stage C. This counter can also serve as a total failure test to monitor the number of oscillations during the accumulation time t_{acc}. If the edges collide before the raw random bit is generated due to environmental influences or an attack, the counter will have much lower values than during the regular operation. This observation can be used to raise the alarm to the user application.

III. Security Assessment

The security assessment provides analytical assurance for the quality of the random bits produced by the digital noise source. Starting from the baseline assumptions about the entropy source and the digitization, we build a stochastic model of the TROT that we use to estimate the min-entropy of the raw random bits.

A. Notations and Baseline Assumptions

We denote the probability of an event E as $\mathbb{P}(E)$. The cumulative distribution function and the probability density function of a normally distributed random variable X with mean μ_X and standard deviation σ_X are denoted as $F_{\mu_X,\sigma_X}(x)$ and $f_{\mu_X,\sigma_X}(x)$, respectively.

The ring oscillator is always sampled in the three-edge mode and the timing jitter affects all three edges. The edges' white Gaussian noise jitter components are mutually independent and their individual observations are independent of each other. Therefore, we only exploit this form of jitter to extract the randomness. The white Gaussian jitter is independent of other noise sources present in the circuit and its variance for each edge grows linearly with time. This increase is characterized by the parameter called jitter strength. Jitter strength is a platform-dependent (chip-dependent) variable and it is obtained empirically as in [21]. We do not extract randomness from timing noise components whose outcomes are correlated at the outputs of stages C and F or any other, possibly correlated and adversarial noise sources, such as power supply or flicker noise, that are present in the circuit. We refer to these components as timing components from the not-exploited noise sources. The not-exploited noise sources might marginally contribute to the total entropy, but they are not used as randomness sources in our model. Thus, the effects of all timing components except the white Gaussian jitter are not removed but instead acknowledged by considering them to be completely deterministic so that a conservative entropy estimation can be provided. However, if the timing influence of the not-exploited noise sources on each edge individually becomes higher than the propagation delay of a single stage of the ring oscillator, the sampled *Raw Bit* value will not be valid and the *Raw Bit Valid* signal will stay at 0, thereby detecting invalid mode of operation. In the model, we further account for the realistic non-identical propagation delays of the delay line bins. These propagation delays are also platform-dependent and empirically obtainable [21]. Together with the jitter strength and the edge oscillation period, they are used as input to the stochastic model. We adopt the standard assumption that the raw random bits are independent due to the reset between consecutively generated bits [22], but not necessarily identically distributed.

B. Stochastic Model

Since the entropy is extracted from the timing jitter of edges α, β and γ, we need to model the time of their occurrences relative to the rising edge of the ring oscillator enable signal *Run*. We denote these times with t_α, t_β and t_γ, respectively. All three edges will have the same nominal period T_{1-RO} because all three edges propagate through identical stages. Each t_α, t_β and t_γ can correspond to an edge of different origin $i, j, k \in \{0, 1, 2\}$, depending on t_{acc}. Thus, depending on the edge origin, we have:

$$t_\alpha^i = t_{r\to C}^i + \lfloor \frac{t_{acc} - t_{r\to C}^i}{T_{1-RO}} \rfloor \cdot T_{1-RO} + t_{\alpha,noise}^i, \quad (3)$$

$$t_\beta^j = t_{r\to F}^j + \lfloor \frac{t_{acc} - t_{r\to F}^j}{T_{1-RO}} \rfloor \cdot T_{1-RO} + t_{\beta,noise}^j, \quad (4)$$

$$t_\gamma^k = t_{f\to C}^k + \lfloor \frac{t_{acc} - t_{f\to C}^k}{T_{1-RO}} \rfloor \cdot T_{1-RO} + t_{\gamma,noise}^k, \quad (5)$$

where $t_{\{f,r\}\to\{C,F\}}^{\{i,j,k\}}$ is the average time needed for the edge $\{i, j, k\}$ to propagate as the falling (f) or the rising edge (r) to the output of stage $\{C, F\}$ for the first time after enabling the ring oscillator, and $t_{\{\alpha,\beta,\gamma\}noise}^{\{i,j,k\}}$ is the timing component originating from various noise sources in the circuit. The edge origin combination which corresponds to the ordered triple $(t_\alpha, t_\beta, t_\gamma)$ depends on t_{acc} and this triple can take

the combinations:

$$(t_\alpha, t_\beta, t_\gamma) = \begin{cases} \text{I) } (t^0_\alpha, t^1_\beta, t^2_\gamma), & \text{if } t_{acc} \in \\ (t^2_{f\to C} + j\cdot T_{1-RO} + t^2_{\gamma,noise}, & \\ \ t^0_{f\to C} + j\cdot T_{1-RO} + t^0_{\gamma,noise}); & \\ \text{II) } (t^1_\alpha, t^2_\beta, t^0_\gamma), & \text{if } t_{acc} \in \\ (t^0_{f\to C} + j\cdot T_{1-RO} + t^0_{\gamma,noise}, & \\ \ t^1_{f\to C} + (j+1)\cdot T_{1-RO} + t^1_{\gamma,noise}); & \\ \text{III) } (t^2_\alpha, t^0_\beta, t^1_\gamma), & \text{if } t_{acc} \in \\ (t^1_{f\to C} + (j+1)\cdot T_{1-RO} + t^1_{\gamma,noise}, & \\ \ t^2_{f\to C} + (j+1)\cdot T_{1-RO} + t^2_{\gamma,noise}); & \end{cases}$$

where $j \in \mathbb{Z}^+$. The first two terms in (3) – (5) are not noise dependent and we replace them with a single term $\mu^{\{i,j,k\}}_{nom,\{\alpha,\beta,\gamma\}}$. The timing components originating from the noise sources can be decomposed into a component originating from the not-exploited noise sources and a component originating from the exploited zero-mean white Gaussian noise sources – $t^{\{i,j,k\}}_{\{\alpha,\beta,\gamma\},G}$. The not-exploited noise sources will further be treated as deterministic – $t^{\{i,j,k\}}_{\{\alpha,\beta,\gamma\},det}$, representing the worst-case scenario. Hence (3) – (5) can be rewritten in a consolidated manner as:

$$t^{\{i,j,k\}}_{\{\alpha,\beta,\gamma\}} = \mu^{\{i,j,k\}}_{nom,\{\alpha,\beta,\gamma\}} + t^{\{i,j,k\}}_{\{\alpha,\beta,\gamma\},det} + t^{\{i,j,k\}}_{\{\alpha,\beta,\gamma\},G}. \tag{6}$$

The expected values of the times of the occurrences of the edges α, β and γ for a given t_{acc} are:

$$\mu_{\{\alpha,\beta,\gamma\}}(t_{acc}) = \mathbb{E}[t^{\{i,j,k\}}_{\{\alpha,\beta,\gamma\}}] = \mu^{\{i,j,k\}}_{nom,\{\alpha,\beta,\gamma\}} + \mathbb{E}[t^{i,j,k}_{\{\alpha,\beta,\gamma\},det}]. \tag{7}$$

Since the white Gaussian noise is independent of the not-exploited noise sources and the not-exploited sources are considered to be deterministic, for the variances of the times of the edge occurrences we have:

$$\sigma^2_{\{\alpha,\beta,\gamma\}}(t_{acc}) = \sigma^2[t^{\{i,j,k\}}_{\{\alpha,\beta,\gamma\}}] = \sigma^2[t^{\{i,j,k\}}_{\{\alpha,\beta,\gamma\},G}]. \tag{8}$$

Given that the variances of the white Gaussian noise increase linearly with time and $\mu^k_{nom,\gamma} > \mu^j_{nom,\beta} > \mu^i_{nom,\alpha}$, it holds:

$$\sigma^2[t^k_{\gamma,G}] > \sigma^2[t^j_{\beta,G}] > \sigma^2[t^i_{\alpha,G}]. \tag{9}$$

Thus, to simplify the stochastic model, without loss in conservatism, we set $\sigma^2_\gamma(t_{acc}) = \sigma^2_\beta(t_{acc}) = \sigma^2_\alpha(t_{acc}) = \sigma^2_{min}(t_{acc})$, where σ^2_{min} is the lower bound on the white Gaussian noise variance of the edge α for the accumulation time t_{acc}. This simplification helps us avoid using the exact values of each stage's rising and falling edge propagation delays, which are usually not equal and cannot be precisely determined. The value of σ^2_{min} can be calculated from the knowledge of the platform jitter strength J_S, the oscillation period of a single edge T_{1-RO} and the smallest number of periods of the edge i that corresponds to the edge α. This number can be obtained from $Cnt(t_{acc})$ – the value of the ripple counter after t_{acc}. The σ^2_{min} value can be determined as:

$$\sigma^2_{min}(t_{acc}) = J_S \cdot \left(\lceil * \rceil \frac{Cnt(t_{acc}) - 1}{3} - 1 \right) \cdot T_{1-RO}. \tag{10}$$

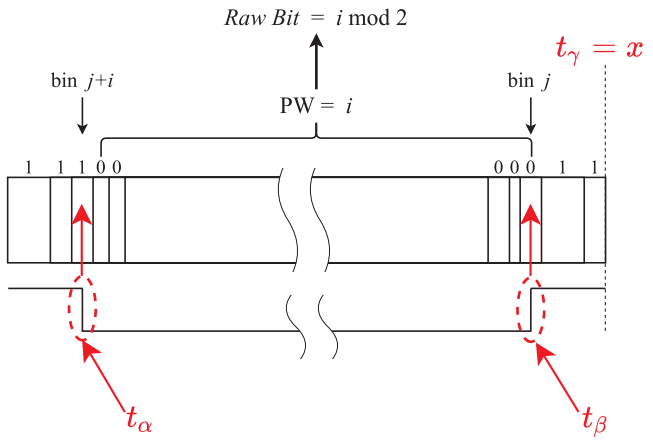

Fig. 4. Modeling of the digitization – determining $PW = i$.

The TDC and the pulse width encoder map relative positions of the edges α and β to the edge γ into a random bit. At the sampling moment, the bins in the delay line will contain a string of zeros encircled by bins with strings of ones – a pulse, as illustrated in Fig. 3. If there are no 0 bins or if the first and the last bin do not have value 1, the raw bit validity signal will have value 0 and the raw bit value will be discarded. We refer to the number of 0s in the delay line as the pulse width (PW). The PW can have any value between 1 and $n-2$, where n is the total number of bins. For the odd PW values, the pulse width encoder outputs 1, and 0 for the even values. Therefore, we calculate the probabilities of all possible PW values to determine the probability of each encoder output. For a given position of the edge γ, we first calculate the probability that the pulse with starting position in bin j will have a width of i bins, as depicted in Fig. 4:

$$\begin{aligned} &\mathbb{P}(PW_j = i | t_\gamma = x) \\ &\quad = \mathbb{P}(t_\beta \in \text{bin } (j) | t_\gamma = x) \\ &\qquad \cdot \mathbb{P}(t_\alpha \in \text{bin } (j+i) | t_\gamma = x). \end{aligned} \tag{11}$$

To remove the condition on the starting position bin, we sum (11) over all possible valid positions j:

$$\begin{aligned} &\mathbb{P}(PW = i | t_\gamma = x) \\ &\quad = \sum_{j=1}^{n-i-1} \mathbb{P}(t_\beta \in \text{bin } (j) | t_\gamma = x) \\ &\qquad \cdot \mathbb{P}(t_\alpha \in \text{bin } (j+i) | t_\gamma = x). \end{aligned} \tag{12}$$

The condition on the position of the edge γ can be removed by integrating out normally distributed t_γ:

$$\begin{aligned} &\mathbb{P}(PW = i) \\ &= \int_{-\infty}^{+\infty} \sum_{j=1}^{n-i-1} \mathbb{P}(t_\beta \in \text{bin } (j) | t_\gamma = x) \cdot \mathbb{P}(t_\alpha \in \text{bin } (j+i) | t_\gamma = x) \\ &\quad \cdot f_{\mu_\gamma, \sigma_{min}}(x)\, dx. \end{aligned} \tag{13}$$

We use $d_{f,i}$ and $d_{r,i}$ to denote $1 \to 0$ and $0 \to 1$ propagation delays of the bin i, respectively. To simplify the notation, we also introduce $d^\Sigma_{r,k} = \sum_{i=0}^k d_{r,i}$ and $d^\Sigma_{f,k} = \sum_{i=0}^k d_{f,i}$.

Then we can rewrite the bin probabilities for t_α and t_β as functions of their distances to $t_\gamma = x$:

$$\begin{aligned}&\mathbb{P}(t_\alpha \in \text{bin}(j+i) | t_\gamma = x)\\ &\quad = \mathbb{P}(x - d^{\Sigma}_{f,j+i} < t_\alpha < x - d^{\Sigma}_{f,j+i-1}),\end{aligned} \tag{14}$$

$$\begin{aligned}&\mathbb{P}(t_\beta \in \text{bin}(j) | t_\gamma = x)\\ &\quad = \mathbb{P}(x - d^{\Sigma}_{r,j} < t_\beta < x - d^{\Sigma}_{r,j-1}).\end{aligned} \tag{15}$$

By substituting (14) and (15) in (13) and using the fact that exploited jitter components are normally distributed, we obtain:

$$\begin{aligned}&\mathbb{P}(PW = i)\\ &= \int_{-\infty}^{+\infty} \sum_{j=1}^{n-i-1} \left[F_{\mu_\beta,\sigma_{min}}(x - d^{\Sigma}_{r,j-1}) - F_{\mu_\beta,\sigma_{min}}(x - d^{\Sigma}_{r,j}) \right]\\ &\quad \cdot \left[F_{\mu_\alpha,\sigma_{min}}(x - d^{\Sigma}_{f,j+i-1}) - F_{\mu_\alpha,\sigma_{min}}(x - d^{\Sigma}_{f,j+i}) \right]\\ &\quad \cdot f_{\mu_\gamma,\sigma_{min}}(x)\, dx.\end{aligned} \tag{16}$$

Since the PW encoding to a raw random bit is used only when the pulse is captured in the TDC, we need to calculate the probability of a specific PW value given that a pulse of any width has been registered in the TDC. We use the capital letter Q to denote the event when the pulse formed by edges α and β is captured in the TDC and its width is at least one TDC bin. The probability of this event is:

$$\mathbb{P}(Q) = \sum_{i=1}^{n-2} \mathbb{P}(PW = i). \tag{17}$$

Finally, the probability of PW being i bins given that the TDC correctly captured a pulse is:

$$\mathbb{P}(PW = i \mid Q) = \frac{\mathbb{P}(PW = i)}{\sum_{i=1}^{n-2} \mathbb{P}(PW = i)}. \tag{18}$$

C. Entropy Estimation of the Raw Random Numbers

The raw random bit is obtained by XOR-ing together outputs of all DFFs in the TDC. Therefore, the binary probabilities are computed by summing the probabilities of all odd pulse widths for bit 1 and all even pulse widths for bit 0:

$$\mathbb{P}(b = l) = \sum_{i=1}^{\frac{n}{2}-1} \mathbb{P}(PW = 2i - l \mid Q), \quad l \in \{0, 1\}. \tag{19}$$

Since each bit is produced after restarting the TRNG and thus assumed to be independent of previous and future outputs, the min-entropy rate of raw random bits can be directly calculated by substituting (16), (18) and (19) in:

$$H^{raw}_{\infty} = -\log_2(max(\mathbb{P}(b = 1), \mathbb{P}(b = 0))). \tag{20}$$

Alternatively, the Shannon entropy rate can be calculated as:

$$H^{raw}_{1} = -\sum_{i=0}^{1} \mathbb{P}(b = i) \cdot \log_2(\mathbb{P}(b = i)). \tag{21}$$

To make a conservative min-entropy estimation claim, besides experimentally obtained J_S, T_{1-RO} and bin propagation delays, we must make additional assumptions on the relations of the values μ_α, μ_β and μ_γ in (16). These values cannot be precisely determined at the design time due to the influence of the operating conditions, the power supply noise and not-exploited correlated local noise sources. From (3) – (5) and (7), we have $\mu^k_{nom,\gamma} - \mu^j_{nom,\beta} \approx d_{RO\,stage}$ and $\mu^j_{nom,\beta} - \mu^i_{nom,\alpha} \approx d_{RO\,stage}$, where $d_{RO\,stage}$ is the average stage delay of the ring oscillator. Therefore, we restrict the difference between μ_α, μ_β and μ_γ to be at least $d_{RO\,stage}/2$ in the model. This restriction is justified because the correlated noise sources will have a relatively small influence compared to the white Gaussian noise for small t_{acc} and because the impact of the power supply noise will be reduced due to its equal influence on all three edges, as previously also observed by [8], [17], [23]. We analyze the effects of μ_α, μ_β and μ_γ on the min-entropy to determine its conservative lower bound $H^{raw}_{\infty,lb}$ in Section IV.

IV. Implementation of the Digital Noise Source and Application of the Stochastic Model

To demonstrate the feasibility of the design and confirm the conservativeness of the min-entropy estimation, we implement the TROT digital noise source in a Xilinx Zynq-7000 FPGA device. We first measure our FPGA chip platform-specific parameters and then apply the stochastic model to determine the lower bound on the min-entropy of the raw random bits $H^{raw}_{\infty,lb}$. Platform-specific parameters are the jitter strength, the propagation delays of the bins in the delay line and the ring oscillator's period in the three-edge mode. Since these parameters are crucial for the correct entropy assessment, they have to be correctly measured before the TRNG is implemented. We chose the system clock period to be $T_{CLK} = 8$ ns, and we measured the white Gaussian noise jitter strength of $J_S = 9.7$ fs on our platform using adapted on-chip differential TDC methodology from [21] with identical routing. The dependence of $H^{raw}_{\infty,lb}$ on the amount of accumulated jitter is used to determine the optimal post-processing construction in Section V.

A. Entropy Source

Each ring oscillator stage is implemented using one look-up table (LUT), while the complete ring oscillator consists of six symmetrically placed LUTs. For the design phase, we used a 9-bit ripple counter at the output of stage C to measure the average period of the ring oscillator. We further monitor its evolution with the increase of the measurement time, as shown in Fig. 5. Measurements are performed in steps of T_{CLK}, and for each step, we repeat the experiment 10^5 times. We observe that for periods higher than 232 ns, the edges in the ring oscillator start colliding, leading the ring oscillator out of the three-edge oscillation mode. This effect can be noted by the sharp rise of the ring oscillator period curve and the increase of its standard deviation. Consequently, the accumulation time t_{acc} has to be lower than 232 ns for the correct operation of the TROT. For the average period of the ring oscillator in the three-edge mode we obtained

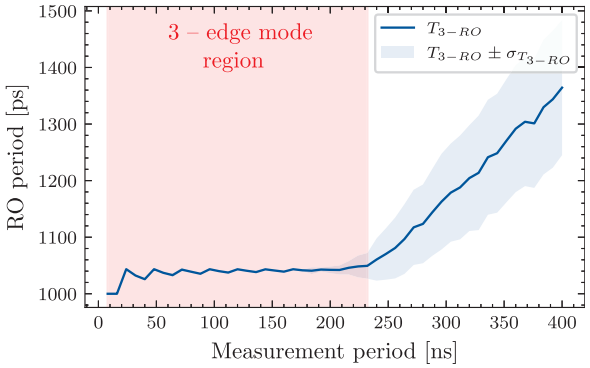

Fig. 5. Evolution of the oscillation period of the ring oscillator – T_{3-RO}.

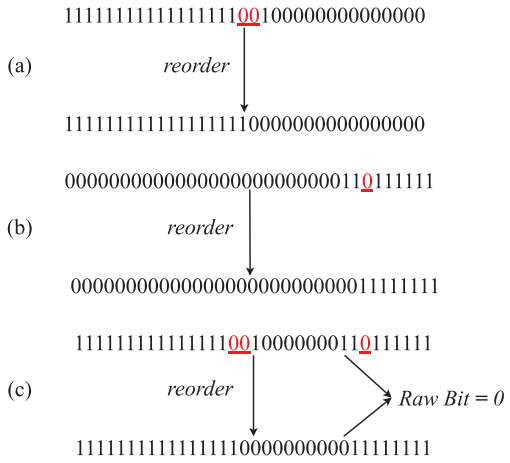

Fig. 6. Examples of bubbles and bin reordering in TDC during: a) rising edge propagation, and b) falling edge propagation during TDC characterization; c) equivalence of ordered and unordered TDC during pulse propagation.

$T_{3-RO} = 1042.57$ ps and the oscillation period of a single edge is $T_{1-RO} = 3 \cdot T_{3-RO} = 12 \cdot d_{RO\,stage} = 3127.7$ ps.

B. Time-to-Digital Converter and Pulse Width Encoder

The delay line is implemented using multiplexer stages of cascaded CARRY4 primitives. The two consecutive stages of each CARRY4 are combined in one bin to decrease the non-uniformity of the bins' propagation time, which has a negative impact on the min-entropy of the TDC-based TRNGs [20]. To measure the propagation delays of the bins, we used the Monte Carlo methodology proposed in [21]. As mentioned in Section II, the captured TDC code will contain bubbles – the occurrences of one or more 0-bins in array of consecutive 1s or the occurrence of one or more 1-bins in array of consecutive 0s. During the propagation delay measurements, the bubbles are handled on the PC by reordering the bins to obtain strings of consecutive 0s and 1s, as depicted in Fig. 6a and Fig. 6b. This procedure is done for every TDC code sample which contains bubbles, but only during the measurements of the bins' delays. On the other hand, as shown by the example in Fig. 6c, the pulse width encoder transforms the width of the captured pulse into one bit, depending only on the parity of the numbers of 0-bins. Thus, the same value is output for both the unordered and ordered TDC bins. This observation implies that the bin reordering is unnecessary

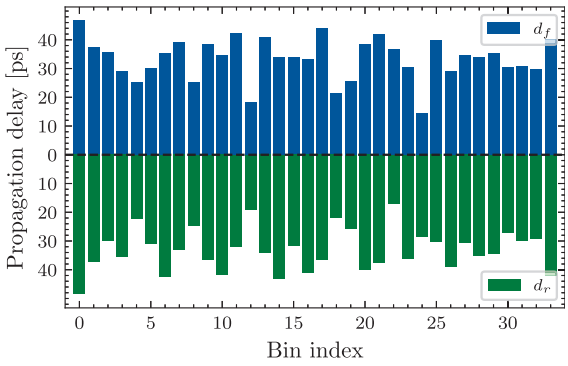

Fig. 7. Characterization of bins in the delay line.

during the TDC operation in TROT since the pulse width encoding into a raw bit is bubble-proof. The obtained rising (d_r) and falling (d_f) delays of the bins are depicted in Fig. 7. Many different calibration techniques have been reported to improve the linearity of the delay line based TDCs [24]–[27]. However, these techniques require hundreds or even thousands of additional LUTs and DFFs. Due to these prohibitively high implementation costs, we chose to instead include bin-to-bin non-linearities in the stochastic model and account for their effect on the lower bound of entropy produced by the TROT digital noise source. Namely, we use individual bin delays in equation (16) in Section III and do not assume that the mean edge position can be precisely determined. The mean edge position is further impacted by various noise sources that are always present but not used in our design. The influence of TDC non-linearity and imprecise mean edge assessment are examined in Subsection IV-C. The number of bins in the delay line needs to be selected such that the total propagation delay for both edges is at least equal to the average period of the ring oscillator in the three-edge mode T_{3-RO}. This condition is satisfied when the delay line has at least 33 bins for our implementation. Since two bins are implemented by one CARRY4, we use 34 bins in 17 CARRY4 primitives and 34 DFFs in the corresponding slices.

The pulse width encoder is realized as a combinatorial circuit for equations (1) and (2) in Section II, and it can be implemented using 15 LUTs. In addition, two DFFs are used to transfer the raw bit value and its validity signal to the system clock domain. Our implementation of the digital noise source required relative placement constraints for symmetrical placement of the ring oscillator stages and for CARRY4 primitives and their corresponding DFFs in FPGA slices above the ring oscillator. The TROT digital noise source fits in 29 slices and its FPGA placement and routing are illustrated in Fig. 8.

C. Stochastic Model Application

By applying the stochastic model and using measured platform-specific parameters, we can numerically calculate the min-entropy for any time difference between μ_α, μ_β and μ_γ, and for different values of the accumulated timing jitter. In this manner, we are also taking into consideration the influence of the not-exploited noise sources on the estimated

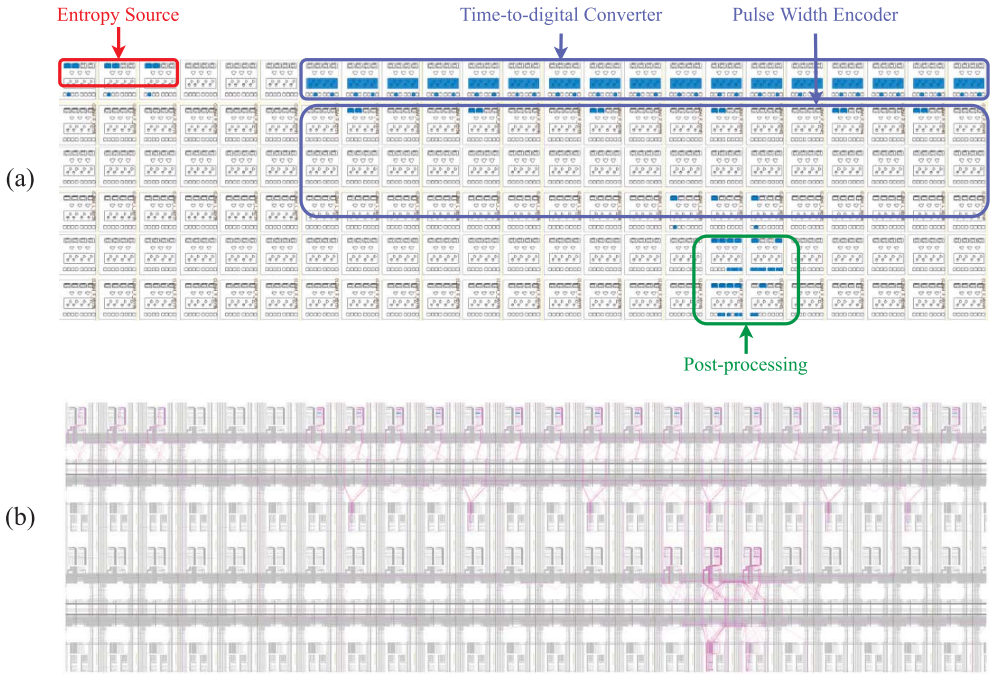

Fig. 8. FPGA placement (a) and routing (b) of the TROT: digital noise source and post-processing.

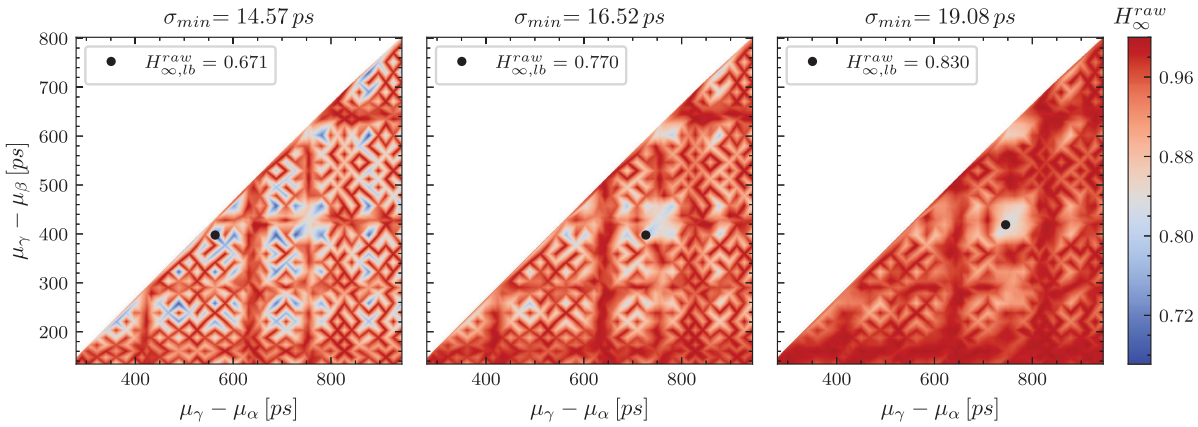

Fig. 9. Dependency of H_{∞}^{raw} on the position of the edges in the TDC line for three different values of the accumulated jitter corresponding to accumulation times 24 ns, 32 ns and 40 ns.

min-entropy. The lower bound on the min-entropy $H_{\infty,lb}^{raw}$ is then obtained by finding the global minimum for each value of the given jitter. Fig. 9 presents the min-entropy estimation as a function of the time distances of μ_α and μ_β to μ_γ for three different amounts of the jitter, where global minima are marked with black dots. We observe that the TDC's non-uniformity indeed significantly impacts the min-entropy and that the position of the global minimum is dependent on the amount of accumulated jitter, i.e., accumulation time. This observation confirms the necessity of including individual bin propagation delays in the stochastic model.

Fig. 10 shows the dependence of the min-entropy's lower bound estimation on the accumulated jitter on the lower x-axis and corresponding accumulation times on the upper x-axis. It can be observed that the min-entropy exhibits a rapid increase for the accumulation times up to 40 ns. For longer accumulation times, the growth tapers off because the non-uniformity of the TDC cannot be overcome even with a considerably higher variance of timing jitter. To confirm the validity of the min-entropy estimations, we applied the entropy estimators from NIST SP 800-90B [3] on the raw random bit sequences obtained for 20 different values of t_{acc} between 8 ns and 160 ns in steps of $T_{CLK} = 8$ ns. The reported results are also presented in Fig. 10. As expected, the min-entropy estimation is always lower than NIST SP 800-90B estimations due to our conservative stochastic model.

D. Comparison With the DC-TRNG

The DC-TRNG [7] is another TRNG design with a stochastic model that also uses time-to-digital conversion for extracting randomness from the ring oscillator jittery edge. Unlike TROT, the DC-TRNG uses a regular one-edge ring oscillator as the entropy source and the time-to-digital converter

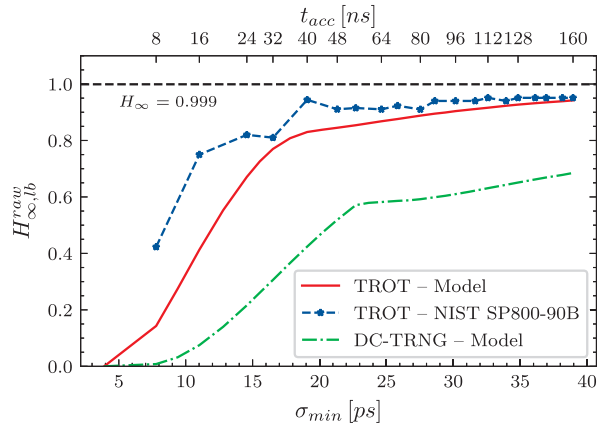

Fig. 10. Dependency of the lower bound of the H^{raw}_{∞} on the amount of the accumulated jitter / accumulation time for the TROT and the DC-TRNG.

to sample the ring oscillator signal by using the system clock after enough white noise jitter has been accumulated. To show the advantage of the TROT design over the DC-TRNG, we estimate its lower bound on the min-entropy of the raw random bits when the DC-TRNG is implemented using identical FPGA resources as the TROT. The dependency of the entropy bound estimation on the accumulation time is also shown in Fig. 10. It can be observed that the min-entropy bound of the TROT grows significantly faster for shorter accumulation times and it remains constantly higher than the min-entropy bound of the DC-TRNG for longer accumulation times.

V. Post-Processing Construction

A. Entropy Estimation of the Internal Random Numbers

As shown in Section IV, the min-entropy of the digital noise source very slowly approaches the desired bound of $H^{int}_{\infty} = 0.999$. Thus, to increase the min-entropy while maintaining high throughput and low implementation cost, we opt to use post-processing based on the binary linear codes [15]. This post-processing represents a generalization of the commonly used XOR post-processing – parity filter. To explain it, we start with a theorem that gives the relation between the min-entropy of the random bits post-processed by any vectorial Boolean function and maximal one-dimensional bias of its output.

Theorem 1: ***(follows from [15])*** *Let $F : \mathbb{F}_2^n \to \mathbb{F}_2^k$ be a vectorial Boolean function with coordinate functions $(f_0, \ldots, f_{k-1}) : \mathbb{F}_2^n \to \mathbb{F}_2$ and δ_{max} maximal one-dimensional bias of its output, defined as $\delta_{max} = \max_{u \in \mathbb{F}_2^k \setminus \{0\}} \left| \mathbb{P}\left(\sum_{i=0}^{k-1} u_i f_i(x) = 1\right) - 0.5 \right|$. Then, the min-entropy rate of the output of F is greater or equal to*:

$$1 - \log_{2^k}(1 + 2^{k+1} \cdot \delta_{max}). \tag{22}$$

It was also proved in [15] that the min-entropy rate of the output bits of F is at least $1 - \log_{2^k}(1 + 2^{k+d} \cdot e^d)$ when F is defined by $F(x) = \boldsymbol{G} \cdot \boldsymbol{x}$. Here, $\boldsymbol{G}$ is $k \times n$ generator matrix of a $[n, k, d]$ linear code and all n bits of $\boldsymbol{x} = (x_0, \ldots, x_{n-1})$ are independent and identically distributed (IID) with constant bias $e = |\mathbb{P}(x_i = 1) - 0.5|$. In contrast to commonly used Von Neumann's debiasing that outputs perfectly unbiased bits but requires IID inputs, this post-processing construction does not require identically distributed bits. With the knowledge of the lower bound on the min-entropy of the raw random bits produced by the TROT, the min-entropy rate of the internal random bits obtained by post-processing with a generator matrix of a linear code can be bounded from below. To this end, we give and prove the following proposition based on Theorem 1.

Proposition 1: *Let $\boldsymbol{x} = (x_0, \ldots, x_{n-1})$ be a sequence of independent but not necessarily identically distributed bits produced by a digital noise source with min-entropy per bit of at least $H^{raw}_{\infty,lb}$. Then, for the min-entropy rate of the output bits H^{int}_{∞} of the post-processing function $F(x) = \boldsymbol{G} \cdot \boldsymbol{x}$, where $\boldsymbol{G}$ is generator matrix of a $[n, k, d]$ linear code, it holds*:

$$H^{int}_{\infty} \geq 1 - \log_{2^k}(1 + (2^{1-H^{raw}_{\infty,lb}} - 1)^d \cdot 2^k). \tag{23}$$

Proof: Since the i^{th} coordinate function of $F(x)$ corresponds to the inner product of the i^{th} row of $\boldsymbol{G}$ and n-bit vector $\boldsymbol{x}$, every $f_i(x)$ is a modulo-2 sum of at least d bits of $\boldsymbol{x}$, by definition of the generator matrix $\boldsymbol{G}$ of the linear code with minimum distance d [15]. The inner product $\sum_{i=0}^{k-1} u_i f_i(x)$ is also a sum of at least d bits of $\boldsymbol{x}$ for all $u \in \mathbb{F}_2^k \setminus \{0\}$, since any non-zero sum of the codewords of a linear code is also a codeword with minimum Hamming weight d. Let $\mathcal{I}_u$ be a set of all indices of $\boldsymbol{x}$ such that $\sum_{i=0}^{k-1} u_i f_i(x) = \bigoplus_{i \in \mathcal{I}_u} x_i$ for $u \in \mathbb{F}_2^k \setminus \{0\}$. Then, the one-dimensional bias of $\sum_{i=0}^{k-1} u_i f_i(x)$ can be computed by applying the piling-up lemma [28]:

$$\delta_u = \left| \mathbb{P}\left(\sum_{i=0}^{k-1} u_i f_i(x) = 1\right) - \frac{1}{2} \right| = 2^{|\mathcal{I}_u|-1} \cdot \prod_{i \in \mathcal{I}_u} e_i,$$

where e_i is bias of the bit x_i. Given that the lower bound on the min-entropy per input bit is known and all bits of $\boldsymbol{x}$ are independent, e_i can be bounded from above $e_i \leq 2^{-H^{raw}_{\infty,lb}} - \frac{1}{2}$. By using this inequality and the fact that by the definition of $\mathcal{I}_u$, the cardinality of $\mathcal{I}_u$ is at least d for all $u \in \mathbb{F}_2^k \setminus \{0\}$, we obtain:

$$\delta_u \leq \delta_{max} \leq 2^{d-1} \cdot \left(2^{-H^{raw}_{\infty,lb}} - \frac{1}{2}\right)^d. \tag{24}$$

Finally, substituting the bound on δ_{max} from (24) in (22) of Theorem 1 leads to (23). □

B. Throughput Optimization

The final throughput of the TROT after post-processing with the generator matrix of a linear code with code rate k/n is computed as:

$$TP = \frac{k}{n} \cdot \frac{1}{t_{acc} + T_{CLK}} \cdot 1\,bit, \tag{25}$$

To optimize the throughput, we need to properly select the code such that $H^{int}_{\infty} \geq 0.999$ holds for given $H^{raw}_{\infty,lb}$.

High values of (25) are obtained with codes that have both high k/n and need low $H^{raw}_{\infty,lb}$, i.e., low t_{acc}. We start our optimization procedure with a list of 32 896 best known binary linear codes from [29]. For each code, we determine the minimum necessary $H^{raw}_{\infty,lb}$ needed to achieve $H^{int}_{\infty} \geq 0.999$ by (23) and the code rate k/n. We then classify a code

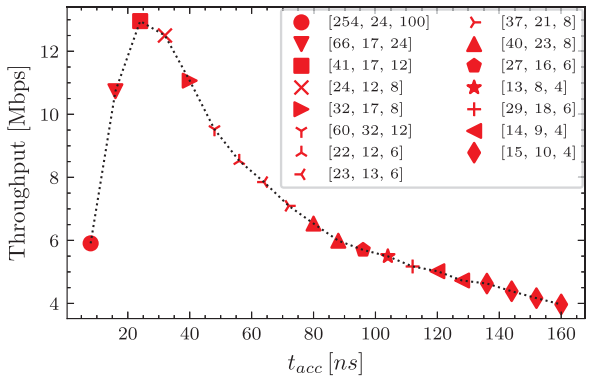

Fig. 11. Expected throughput after post-processing with the generator matrix of a suitable linear code for different accumulation times and $H_{\infty}^{int} \geq 0.999$.

as *suitable* if: (i) among all codes with the same k/n, it has the lowest minimum necessary $H_{\infty,lb}^{raw}$; (ii) among all codes with same minimum necessary $H_{\infty,lb}^{raw}$, it has the highest k/n; (iii) all other suitable codes with higher k/n have higher minimum necessary $H_{\infty,lb}^{raw}$. The set of suitable codes will contain codes that best meet both the high code rate and the low minimum necessary $H_{\infty,lb}^{raw}$ criteria. The obtained 468 suitable codes are sorted in descending order of their minimum necessary $H_{\infty,lb}^{raw}$. Then, as in Fig. 10, for 20 different t_{acc} values between 8 ns and 160 ns in steps of T_{CLK}, we calculate $H_{\infty,lb}^{raw}$, and for each value we select the first suitable code from the sorted list with a lower minimum necessary $H_{\infty,lb}^{raw}$. Fig. 11 presents maximum achievable throughput by (25) for different accumulation times and obtained suitable codes. The highest throughput of 12.957 *Mbps* is achieved for $t_{acc} = 24$ ns by post-processing with the generator matrix of *[41, 17, 12]* code, while the second-highest throughput of 12.5 Mbps is achieved with the generator matrix of the *[24, 12, 8]* code for $t_{acc} = 32$ ns. For $t_{acc} > 160$ ns, the throughput continues to fall under 4 Mbps.

C. Custom Post-Processing Architecture

The *[41, 17, 12]* code is not cyclic, and thus it does not have a generator polynomial that can be used for efficient hardware implementation. However, if we somewhat sacrifice the throughput, an efficient hardware implementation can be achieved with *[24, 12, 8]* Golay code. We first apply elementary transformations on the standard form of its generator matrix $\boldsymbol{G_S}$:

$$\boldsymbol{G_S} = \left[\begin{array}{c|cccccccccccc} \mathbb{I}_{\mathbf{12}} & 1&1&0&1&1&1&1&0&1&0&0&0 \\ & 0&1&1&0&1&1&1&1&0&1&0&0 \\ & 0&0&1&1&0&1&1&1&1&0&1&0 \\ & 0&0&0&1&1&0&1&1&1&1&0&1 \\ & 1&0&0&0&1&1&0&1&1&1&1&0 \\ & 0&1&0&0&0&1&1&0&1&1&1&1 \\ & 1&0&1&0&0&0&1&1&0&1&1&1 \\ & 1&1&0&1&0&0&0&1&1&0&1&1 \\ & 1&1&1&0&1&0&0&0&1&1&0&1 \\ & 1&1&1&1&0&1&0&0&0&1&1&0 \\ & 0&1&1&1&1&0&1&0&0&0&1&1 \\ & 1&0&1&1&1&1&0&1&0&0&0&1 \end{array}\right],$$

where $\mathbb{I}_{\mathbf{12}}$ is an identity matrix of size 12. This matrix has the form $[\mathbb{I}_{\mathbf{12}} \,|\, \boldsymbol{A}]$, where $\boldsymbol{A}$ is a circulant 12×12 matrix. Since the column permutations of the generator matrix do not change the minimum distance of the linear code, matrix $\boldsymbol{G_S}$ can be directly transformed to obtain a hardware implementation-friendly form: $\boldsymbol{G} = [\boldsymbol{A} \,|\, \mathbb{I}_{\mathbf{12}}]$. This representation and circulant property of $\boldsymbol{A}$ enable a very compact implementation that occupies 19 DFFs and 11 LUTs, as illustrated in Fig. 12. The complete post-processing fits in 4 slices, and its placement and routing are depicted in Fig. 8. During the first twelve clock cycles, the post-processed bit valid signal has value 0 and the raw random bits are shifted into registers $Q0 - Q11$. In the subsequent twelve cycles, these registers are connected in a circular shift register through a multiplexer at the input of $Q0$, while the valid signal has value 1. At the same time, the post-processed bits are obtained by XOR-ing the incoming raw bits and the outputs of the circular shift register at the positions determined by the ones in the first row of matrix $\boldsymbol{A}$.

VI. Implementation Comparisons and Statistical Evaluations

A fair comparison of the TRNG designs on FPGAs is not a straightforward task since most designs do not have a stochastic model or an explicit statement of the achievable entropy rate, as required by AIS-31 and NIST SP 800-90B security standards. Further, the TRNGs with entropy estimation do not always use the same entropy metric or are designed to provide different entropy rates. We compare TROT with several recent FPGA compatible TRNGs in Table I. In addition to hardware utilization, throughput and power consumption estimation, we report the estimated entropy rate of the output bits, the maximum achievable throughput per slice and the availability of the AIS-31 compliant stochastic model. Note that for both TROT implementations - with and without the post-processing, we use the same accumulation time of 32 ns, which results in lower raw min-entropy than most previously published designs. However, since TROT uses information-theoretic post-processing for the cost of only four additional slices, the min-entropy rate can be brought to above 0.999 with a 50 % throughput reduction. As can be seen in Table I, the TROT has the highest entropy rate among previously reported TRNG designs. The entropy rate of TROT is estimated from the conservative stochastic model and thus provides higher security level guarantees [4]. The two most recent designs - JL-TRNG [30] and MFRO-TRNG [31] achieve higher throughput than TROT, but they are not supported by the AIS-31 compliant stochastic model. Moreover, the power consumption of MFRO-TRNG [31] is almost 400 times higher than the power consumption of TROT. Similarly, the smallest design in Table I – LRO-TRNG [32] is also not supported by the stochastic model. The stochastic model and the formal entropy estimation based on the model are needed for the use of TRNGs in high-end security applications [4]. Compared with a TRNG with a stochastic model and throughput of the same order – the DC-TRNG [7], our design is more lightweight. The design presented in [40] is the second version of a TRNG with a stochastic model – ES-TRNG [8]. According to Table I,

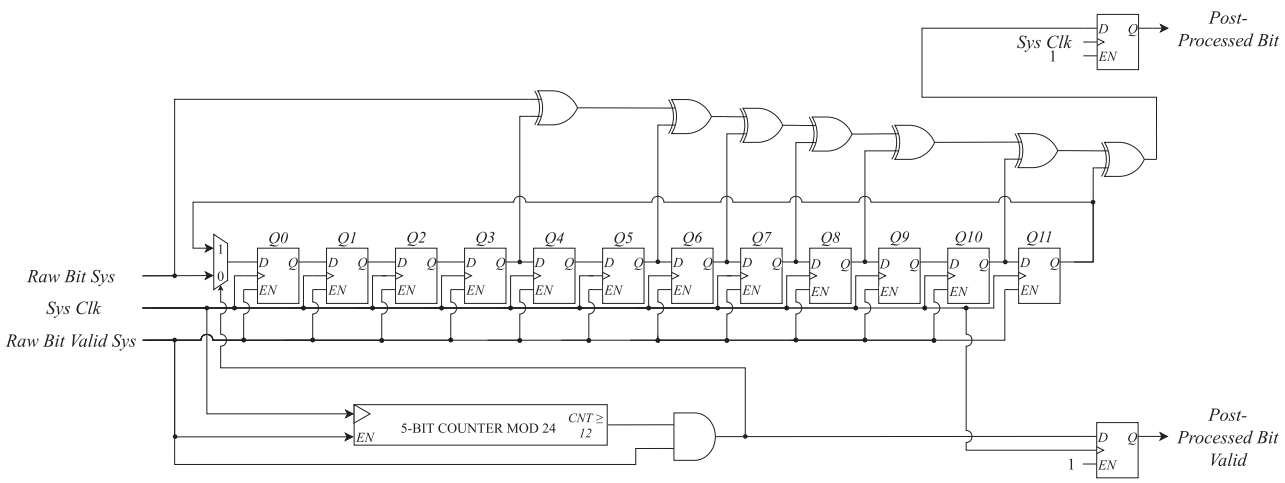

Fig. 12. Hardware architecture of the TROT post-processing with the generator matrix of the [24, 12, 8] code.

TABLE I
IMPLEMENTATION RESULTS AND COMPARISON WITH EXISTING FPGA TRNG DESIGNS

TRNG design	FPGA family	Entropy estimation	Area	Throughput [Mbps]	Throughput / Slice [Mbps / Slice]	Power [mW]	Stoch. mod. AIS-31
JL-TRNG [30]	Spartan 6 / Virtex 6	$H_\infty = 0.891$[a]	56 LUTs / 19 FFs	100	7.142	1.15	✗
MFRO-TRNG [31]	Virtex 6	$H_\infty = 0.985$[a]	24 LUTs / 2 FFs	290	48.33	3687	✗
LRO-TRNG [32]	Spartan 6	$H_1 = 0.999$[b]	4 LUTs / 3 FFs	0.76	0.76	*	✗
LLRED-TRNG [33]	Cyclone IV	$H_1 \geq 0.999$	343 LUTs / 229 FFs	0.018	*not applicable*[c]	10.22	✗
REC-TERO [34]	Zynq-7000 (Artix-7)	*	194 LUTs / 61 FFs	3.33	0.06795	*	✗
TC-TERO [35]	Artix-7	$H_1 = 0.9993$[b]	40 LUTs / 29 FFs	1.91	0.191	*	✓
DEP-TRNG [36]	Artix-7	$H_\infty = 0.9942$	252 LUTs / 150 FFs[d]	3	0.0476	79	✓
MTB-TRNG [37]	Spartan 6	*	271 slices	1	0.00369	90	✗
ERO[e] [9], [6]	Spartan 6	$H_1 = 0.999$[b]	46 LUTs / 19 FFs	0.0042	0.00035	2.16	✓
MURO[e] [10], [6]	Spartan 6	$H_1 = 0.999$[b]	521 LUTs / 131 FFs	2.57	0.0196	54.72	✓
COSO[e] [11], [6]	Spartan 6	$H_1 = 0.999$[b]	18 LUTs / 3 FFs	0.54	0.108	1.22	✓
PLL[e] [12], [6]	Spartan 6	$H_1 = 0.981$[b]	34 LUTs / 14 FFs / 2 PLLs	0.44	0.0488	10.6	✓
TERO[e] [13], [6]	Spartan 6	$H_1 = 0.999$[b]	39 LUTs / 12 FFs	0.625	0.0625	3.312	✓
STR [14]	Virtex 5	$H_1 = 0.99$	$>$ 616 LUTs / 616 FFs[f,g]	100	0.6493	*	✓
COSO-PA [38]	Virtex 5	*	109 slices	4.08	0.0374	*	✗
JBA-TRNG [39]	Virtex UltraScale+	$H_\infty^{raw} \geq 0.429$	184 slices (791 LUTs / 559 FFs / 33 CARRY8s)[h]	2.43	0.0132	*	✓
DC-TRNG [7]	Spartan 6	$H_1 \geq 0.999$	67 slices[g]	14.3	0.2134	*	✓
ES-TRNG [8]	Spartan 6	$H_1 \geq 0.997$	10 LUTs / 5 FFs / 1 CARRY4[g]	1.15	0.2875	*	✓
ES-TRNG v2 [40]	Spartan 6	$H_\infty = 0.815$[a]	7 LUTs / 6 FFs / 1 CARRY4	10	2.5	*	✓[i]
FC-TRNG [41]	Artix-7	$H_\infty^{int} = 0.893$[a]	62 LUTs / 21 FFs	5.102 (mean)	0.3189	*	✗
TROT w/o post-proc.	Zynq-7000 (Artix-7)	$H_\infty^{raw} \geq 0.770$ $H_1^{raw} \geq 0.978$	29 slices (21 LUTs / 36 FFs / 17 CARRY4)	25	0.8621	8.843	✓
TROT w/ post-proc.		$H_\infty^{int} \geq 0.999$ $H_1^{int} \geq 0.99999$	33 slices (32 LUTs / 55 FFs / 17 CARRY4)	12.5	0.3787	9.514	

* Information not provided in the original publication [a] Min-entropy (H_∞) estimated by NIST SP 800-90B [b] Shannon entropy (H_1) estimated by AIS-31 T8 test [c] Intel's Cyclone IV does not use slices as logic structures [d] Resources without redundant blocks and entropy tests
[e] Reported implementation results from [6]; these designs are also implemented in Intel Cyclone V FPGAs
[f] Estimated minimal resources for configuration that passes NIST SP 800-22 [16] [g] Resources for the digital noise source, while the design also contains post-processing [h] Resources for both digital noise source and post-processing [i] Stochastic model from [8], but entropy not derived from the model

this design achieves similar throughput as TROT. However, its entropy estimation is not supported by the stochastic model, i.e., it is obtained by running NIST SP 800-90B black-box estimators. The most similar design to ours is the one of [41], which implements the three-edge ring oscillator TRNG on the Artix-7 FPGA and is based on the design of [17]. Our design has similar resource utilization, but it achieves 2.5 times higher throughput. On the other hand, the TROT has at least

TABLE II
RESULTS OF THE NIST SP 800-90B ENTROPY ESTIMATION

Estimator	H_∞
Most Common Value	0.999715
Collision	0.986374
Markov	0.999866
Compression	0.960924
t-Tuple	**0.945605**
LRS	0.99869
Multi Most Common in Window Prediction	0.999821
Lag Prediction	0.999803
MultiMMC Prediction	0.999809
LZ78Y Prediction	0.999742

TABLE III
RESULTS OF THE AIS-31 STATISTICAL TEST SUITE

Test	Pass rate	Result
T0 – Disjointness test	1/1	Pass
T1 – Monobit test	257/257	Pass
T2 – Poker test	257/257	Pass
T3 – Runs test	257/257	Pass
T4 – Long run test	257/257	Pass
T5 – Autocorrelation test	257/257	Pass
Test	**Test statistic / Pass condition**	**Result**
T6 – Uniform distribution test	$\lvert\mathbb{P}(1) - 0.5\rvert = 0.00113 / < 0.025$ $\lvert\mathbb{P}(01) - \mathbb{P}(11)\rvert = 9.899 \cdot 10^{-4} / < 0.02$	Pass
T7 – Test for homogeneity	$T[0] = 2.91; T[1] = 0.04; T[00] = 0.06; T[01] = 0.003; T[10] = 0.03; T[11] = 0.002; / < 15.13$	Pass
T8 – Entropy estimation	$H_1 = 7.996 / \geq 7.976$	Pass

TABLE IV
MIN-ENTROPY ESTIMATES IN THE ON-CHIP ATTACK OPERATING CONDITIONS

Estimator	H_∞
Most Common Value	0.91184
Collision	0.90506
Markov	0.91642
Compression	**0.77925**
t-Tuple	0.86614
LRS	0.98458
Multi Most Common in Window Prediction	0.82979
Lag Prediction	0.82980
MultiMMC Prediction	0.91518
LZ78Y Prediction	0.91518

an order of magnitude higher throughput when compared with designs with the stochastic model and lower resource consumption – ERO [6], COSO [6], PLL [6], TERO [6] and ES-TRNG [8]. Among the TRNGs with the stochastic model, the TROT achieves the second-best result in throughput per slice metric, quantifying the area versus throughput trade-off. According to this criterion, the STR [14] is the only TRNG that performs better. However, this TRNG has significant area requirements and substantially higher design effort. Moreover, its entropy rate is lower than the TROT's and compression with 4-stage XOR post-processing is required to obtain the random bits of the same quality, thereby reducing the throughput per slice to 0.1623 – more than twice lower than that of our design.

To evaluate the statistical quality of the random bits produced by the TROT, we applied AIS-31 statistical test suite [4] and non-IID track of NIST SP 800-90B [3] min-entropy estimators.

The non-IID track of NIST SP 800-90B consists of ten min-entropy estimators and the final estimation is the minimum of all ten estimators. The obtained min-entropy value is confirmed by running the restart tests [3]. We applied the NIST SP 800-90B estimators on $5 \cdot 10^8$ internal random bits and summarized the results in Table II, where the bold value indicates the lowest estimate. Note that these estimators are black-box tests and thus often underestimate the min-entropy [3]. To demonstrate this, we applied the same estimators on $5 \cdot 10^8$ of the random bits produced by NIST SP 800-90B compliant Intel's TRNG on Intel Core i7-107050H CPU. This TRNG has a formal entropy justification [42] and it uses cryptographic post-processing based on AES CBC-MAC to produce random bits with full entropy [43]. However, NIST SP 800-90B estimators report only 0.92905 bits of min-entropy, thus severely underestimating the actual min-entropy of Intel's TRNG. This result highlights the importance of not relying only on the black-box tests for the entropy estimation and having a viable stochastic model.

For statistical verification with the AIS-31 test suite [4], we applied the evaluation method B for the PTG.2 class of TRNGs, the highest security class that does not require cryptographic post-processing. This method comprises nine tests $T0 - T8$ that are applied to 10 MB of the internal random numbers. Table III lists the evaluation results for all nine tests and additional test statistics for $T6 - T8$ tests.

VII. RESILIENCE AGAINST THE ATTACKS, VOLTAGE AND TEMPERATURE VARIATIONS

To examine the resilience of the TROT against the on-chip attacks [44], we implemented 1600 one-stage ring oscillators as power-wasting circuits. We place two ring oscillators in a single slice and a total of 800 slices are placed around the TROT, as close as possible to the digital noise source. All ring oscillators are simultaneously periodically enabled at the frequency of 25 kHz. We collect the raw random bits produced by the TROT and estimate the min-entropy with NIST SP 800-90B tests. The entropy estimation results are shown in Table IV. Since the minimal estimate is above the model-predicted 0.770, we can conclude that this type of attack on the TROT seems insufficiently effective.

We performed the frequency injection attacks equivalent to those proposed in [45] and [46], which were proven successful when applied on the single-edge RO-based TRNGs. For our attacks, we inject oscillating signals in the delay line next to the three-edge RO to induce strong interaction between the RO signal and the injected signal. In the first scenario, the injected signal comes from a separate single-edge RO with two stages, placed and routed so that its frequency matches as closely as possible to the frequency of the TROT's RO in the three-edge mode. This injected signal causes locking of the edges, preventing the collapse of the three-edge mode in TROT, as shown in Fig. 13. The estimated min-entropy of the raw random bits produced during this attack is 0.79. Therefore, the jitter reduction in this scenario seems to be limited due

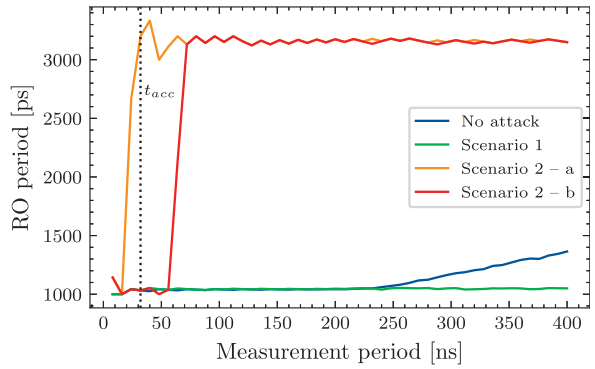

Fig. 13. Frequency injection attack: ring oscillator period evolution for different attack scenarios. In scenario 2, the locking can occur before (Scenario 2 – a) or after (Scenario 2 – b) sampling the raw bit.

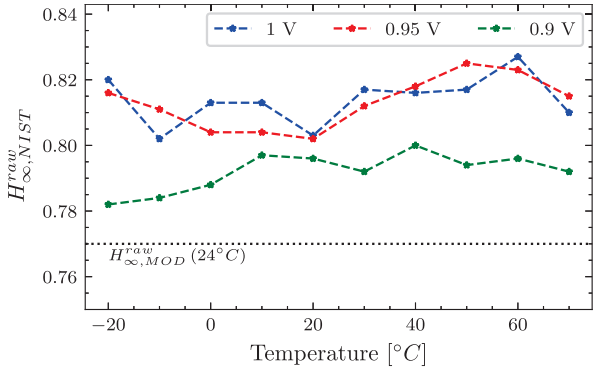

Fig. 14. Min-entropy of the raw random bits with varying temperature and supply voltage.

to the very short accumulation time and TROT reset between each generated bit. In the second scenario, another single-edge RO with six stages is used to create the injection signal with a frequency equal to TROT's RO frequency in the single-edge mode. This injection signal affects the TROT by significantly reducing the duration of the three-edge mode, as also depicted by two examples in Fig. 13. We observed that the duration of the three-edge mode is not always consistent between the experiments. In some experiments, the attack is unsuccessful since the transition from the three-edge to the single-edge mode occurs after the raw bit has already been sampled. In other experiments, the attack is deemed successful as the transition to the single-edge mode occurs before sampling the raw bit. However, since the pulse width encoder would not correctly capture the edges in a single-edge mode when the transition occurs before the bit sampling, the raw bit validity signal will remain low, thereby preventing output of the compromised bits.

We examined the influence of the lower voltage supplies and different temperatures on the entropy produced by the TROT digital noise source. The temperature experiments are performed in Espec SH-662 climate chamber with swiping temperatures in $[-20°C,\ 70°C]$ range with $10°C$ step for three different logic core supply voltages: 1 V (nominal), 0.95 V and 0.9 V. For lower logic core voltages, the communication logic stops properly functioning and reliable data collection was not feasible. We run NIST SP 800-90B estimators with the FPGA restarted between each experiment for each temperature-voltage operating point. The experimental results are depicted in Fig. 14. We observe that the estimated entropies are consistently above the bound given by the model in Section III. There is no clear trend for 1 V and 0.95 V, and the estimated min-entropies are always higher than 0.8. On the other hand, the estimated entropies are systematically lower when the core supply voltage is set to 0.9 V. When combined with temperatures below $0°C$, the entropies show a downward trend, with the lowest entropy of above 0.78 for $-20°C$. Thanks to the low worst-case min-entropy bound derived from the stochastic model, such entropy drop does not represent a security threat since the post-processing is designed to compensate for the raw bit min-entropies higher than 0.770.

VIII. Conclusion

In this paper, we presented a new true random number generator – TROT and its design method. The TROT incorporates a three-edge mode ring oscillator with a novel TDC-based digitization technique and optimized information-theoretically secure post-processing to obtain a relatively compact design with high throughput and low design effort. Additionally, TROT achieves a min-entropy rate of more than 0.999 and it is supported by a conservative security assessment as prescribed by both AIS-31 and NIST SP 800-90B. These properties make our TRNG more suitable than previously reported designs to implement with the area- and randomness-consuming cryptographic systems and SCA countermeasures. Future work will also include the design of dedicated on-the-fly tests, achieving full compliance with AIS-31 and NIST SP800-90B standards.

References

[1] L. De Meyer, "Cryptography in the presence of physical attacks: Design, implementation and analysis," Ph.D. dissertation, Dept. Eng. Sci., Dept. Elect. Eng., KU Leuven, Leuven, Belgium, 2020.

[2] L. De Meyer, "Recovering the CTR DRBG state in 256 traces," *IACR Trans. Cryptogr. Hardw. Embedded Syst.*, vol. 2020, no. 1, pp. 37–65, Nov. 2019.

[3] M. S. Turan, E. Barker, J. Kelsey, K. McKay, M. Baish, and M. Boyle, "NIST special publication 800-90B: Recommendation for the entropy sources used for random bit generation," NIST, Gaithersburg, MD, USA, Tech. Rep. NIST Special Publication 800-90B, Jan. 2018.

[4] W. Killmann and W. Schindler, "A proposal for: Functionality classes for random number generators," BSI, Bonn, Germany, Tech. Rep., Sep. 2011.

[5] D. Li *et al.*, "GB/T 32915-2016 information security technology—Randomness test methods for binary sequence," Standard, Standardization Admin. PRC, Beijing, China, Tech. Rep. GM/T 0005-2012, Aug. 2016.

[6] O. Petura, U. Mureddu, N. Bochard, V. Fischer, and L. Bossuet, "A survey of AIS-20/31 compliant TRNG cores suitable for FPGA devices," in *Proc. 26th Int. Conf. Field Program. Log. Appl. (FPL)*, Aug. 2016, pp. 1–10.

[7] V. Rožić, B. Yang, W. Dehaene, and I. Verbauwhede, "Highly efficient entropy extraction for true random number generators on FPGAs," in *Proc. 52nd Annu. Des. Autom. Conf.*, Jun. 2015, pp. 1–6.

[8] B. Yang, V. Rožić, M. Grujić, N. Mentens, and I. Verbauwhede, "ES-TRNG: A high-throughput, low-area true random number generator based on edge sampling," *IACR Trans. Cryptogr. Hardw. Embedded Syst.*, vol. 2018, no. 3, pp. 267–292, 2018.

[9] M. Baudet, D. Lubicz, J. Micolod, and A. Tassiaux, "On the security of oscillator-based random number generators," *J. Cryptol.*, vol. 24, no. 2, pp. 398–425, 2011.

[10] 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.

[11] P. Kohlbrenner and K. Gaj, "An embedded true random number generator for FPGAs," in *Proc. ACM/SIGDA 12th Int. Symp. Field Program. Gate Arrays (FPGA)*, Feb. 2004, pp. 71–78.

[12] V. Fischer and M. Drutarovský, "True random number generator embedded in reconfigurable hardware," in *Proc. 4th Int. Workshop Cryptogr. Hardw. Embedded Syst. (CHES)*, Aug. 2002, pp. 415–430.

[13] M. Varchola and M. Drutarovský, "New high entropy element for FPGA based true random number generators," in *Proc. 12th Int. Workshop Cryptograph. Hardw. Embedded Syst. (CHES)*, Aug. 2010, pp. 351–365.

[14] A. Cherkaoui, V. Fischer, L. Fesquet, and A. Aubert, "A very high speed true random number generator with entropy assessment," in *Proc. 15th Int. Workshop Cryptogr. Hardw. Embedded Syst. (CHES)*, Aug. 2013, pp. 179–196.

[15] P. Lacharme, "Post-processing functions for a biased physical random number generator," in *Proc. Int. Workshop Fast Softw. Encryption*, Feb. 2008, pp. 334–342.

[16] L. E. Bassham *et al.*, "A statistical test suite for random and pseudorandom number generators for cryptographic applications," NIST, Gaithersburg, MD, USA, Tech. Rep. Special Publication 800-22, Rev. 1a, Apr. 2010.

[17] K. Yang, D. Fick, M. B. Henry, Y. Lee, D. Blaauw, and D. Sylvester, "16.3 A 23Mb/s 23pJ/b fully synthesized true-random-number generator in 28 nm and 65 nm CMOS," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2014, pp. 280–281.

[18] H. Zhuang *et al.*, "A second-order noise-shaping SAR ADC with passive integrator and tri-level voting," *IEEE J. Solid-State Circuits*, vol. 54, no. 6, pp. 1636–1647, Jun. 2019.

[19] H. Zhuang, J. Liu, H. Tang, X. Peng, and N. Sun, "A fully dynamic low-power wideband time-interleaved noise-shaping SAR ADC," *IEEE J. Solid-State Circuits*, vol. 56, no. 9, pp. 2680–2690, Sep. 2021.

[20] M. Grujić, V. Rožić, B. Yang, and I. Verbauwhede, "A closer look at the delay-chain based TRNG," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2018, pp. 1–5.

[21] B. Yang, V. Rožić, M. Grujic, N. Mentens, and I. Verbauwhede, "On-chip jitter measurement for true random number generators," in *Proc. Asian Hardw. Oriented Secur. Trust Symp. (AsianHOST)*, Oct. 2017, pp. 91–96.

[22] M. Bucci and R. Luzzi, "Design of testable random bit generators," in *Proc. 7th Int. Workshop Cryptogr. Hardw. Embedded Syst. (CHES)*, Aug. 2005, pp. 147–156.

[23] P. Haddad, V. Fischer, F. Bernard, and J. Nicolai, "A physical approach for stochastic modeling of TERO-based TRNG," in *Proc. 17th Int. Workshop Cryptogr. Hardw. Embedded Syst. (CHES)*, Sep. 2015, pp. 357–372.

[24] R. Machado, J. Cabral, and F. S. Alves, "Recent developments and challenges in FPGA-based time-to-digital converters," *IEEE Trans. Instrum. Meas.*, vol. 68, no. 11, pp. 4205–4221, Nov. 2019.

[25] Y. Wang *et al.*, "Performance analysis and IP core implementation of two high performance time-to-digital converters on Xilinx 7-series FPGA," *Nucl. Instrum. Methods Phys. Res. A, Accel. Spectrom. Detect. Assoc. Equip.*, vol. 1020, Dec. 2021, Art. no. 165866.

[26] M. Parsakordasiabi, I. Vornicu, A. Rodriguez-Vazquez, and R. Carmona-Galan, "An efficient TDC using a dual-mode resource-saving method evaluated in a 28-nm FPGA," *IEEE Trans. Instrum. Meas.*, vol. 71, pp. 1–13, 2022.

[27] J. Zheng, P. Cao, D. Jiang, and Q. An, "Low-cost FPGA TDC with high resolution and density," *IEEE Trans. Nucl. Sci.*, vol. 64, no. 6, pp. 1401–1408, Jun. 2017.

[28] M. Matsui, "Linear cryptanalysis method for DES cipher," in *Proc. Workshop Theory Appl. Cryptogr. Techn. Adv. Cryptol. (EUROCRYPT)*, May 1993, pp. 386–397.

[29] M. Grassl. (2007). *Bounds on the Minimum Distance of Linear Codes and Quantum Codes*. Accessed: Jan. 15, 2021. [Online]. Available: http://www.codetables.de

[30] X. Wang *et al.*, "High-throughput portable true random number generator based on jitter-latch structure," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 68, no. 2, pp. 741–750, Feb. 2021.

[31] J. Cui *et al.*, "Design of true random number generator based on multi-stage feedback ring oscillator," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, early access, Sep. 8, 2021, doi: 10.1109/TCSII.2021.3111049.

[32] R. Della Sala, D. Bellizia, and G. Scotti, "A novel ultra-compact FPGA-compatible TRNG architecture exploiting latched ring oscillators," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, early access, Oct. 20, 2022, doi: 10.1109/TCSII.2021.3121537.

[33] L. B. Carreira, P. Danielson, A. A. Rahimi, M. Luppe, and S. Gupta, "Low-latency reconfigurable entropy digital true random number generator with bias detection and correction," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 67, no. 5, pp. 1562–1575, May 2020.

[34] B. Acar and S. Ergun, "A reconfigurable random number generator based on the transient effects of ring oscillators," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 67, no. 9, pp. 1609–1613, Sep. 2020.

[35] N. Fujieda, "On the feasibility of TERO-based true random number generator on Xilinx FPGAs," in *Proc. 30th Int. Conf. Field-Program. Log. Appl. (FPL)*, Aug. 2020, pp. 103–108.

[36] H. Martin, G. Di Natale, and L. Entrena, "Towards a dependable true random number generator with self-repair capabilities," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 65, no. 1, pp. 247–256, Jan. 2018.

[37] P. Z. Wieczorek, "Lightweight TRNG based on multiphase timing of bistables," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 63, no. 7, pp. 1043–1054, Jul. 2016.

[38] J. Yang, Y. Ma, T. Chen, J. Lin, and J. Jing, "Extracting more entropy for TRNGs based on coherent sampling," in *Proc. 12th Int. Conf. Security Privacy Commun. Syst.*, Oct. 2016, pp. 694–709.

[39] X. Li, P. Stanwicks, G. Provelengios, R. Tessier, and D. Holcomb, "Jitter-based adaptive true random number generation for FPGAs in the cloud," in *Proc. Int. Conf. Field-Program. Technol. (ICFPT)*, Dec. 2020, pp. 112–119.

[40] N. Klein, E. Harel, and I. Levi, "The cost of a true random bit—On the electronic cost gain of ASIC time-domain-based TRNGs," *Cryptography*, vol. 5, no. 3, p. 25, Sep. 2021.

[41] R. Serrano *et al.*, "A fully digital true random number generator with entropy source based in frequency collapse," *IEEE Access*, vol. 9, pp. 105748–105755, 2021.

[42] R. J. Parker, "Entropy justification for metastability based nondeterministic random bit generator," in *Proc. IEEE 2nd Int. Verification Secur. Workshop (IVSW)*, Jul. 2017, pp. 25–30.

[43] D. Johnston, *Random Number Generators—Principles and Practices, A Guide for Engineers and Programmers*. Berlin, Germany: De Gruyter, Sep. 2018.

[44] D. Mahmoud and M. Stojilović, "Timing violation induced faults in multi-tenant FPGAs," in *Proc. Des., Autom. Test Eur. Conf. Exhib. (DATE)*, Mar. 2019, pp. 1745–1750.

[45] U. Mureddu, N. Bochard, L. Bossuet, and V. Fischer, "Experimental study of locking phenomena on oscillating rings implemented in logic devices," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 66, no. 7, pp. 2560–2571, Jul. 2019.

[46] Y. Koyen, A. Peetermans, V. Rožić, and I. Verbauwhede, "Attacking hardware random number generators in a multi-tenant scenario," in *Proc. Workshop Fault Detection Tolerance Cryptogr. (FDTC)*, Sep. 2020, pp. 18–25.

Miloš Grujić received the B.Sc. and M.Sc. degrees in electrical engineering from the University of Belgrade School of Electrical Engineering in 2014 and 2015, respectively. He is currently pursuing the Ph.D. degree with the COSIC Research Group, Department of Electrical Engineering (ESAT), KU Leuven, Belgium. His main research interests include random number generators (RNGs), design of lightweight tests for the RNGs, and hardware implementations of cryptographic primitives.

Ingrid Verbauwhede (Fellow, IEEE) is currently a professor with the research group COSIC, Electrical Engineering Department, KU Leuven, Belgium. At COSIC, she leads the embedded systems and hardware group. She is also an adjunct professor with the EE Department, University of California, Los Angeles, California. She is a member of the Royal Academy of Belgium for Science and the Arts. She was a recipient of an ERC Advanced Grant in 2016. She received the IEEE 2017 Computer Society Technical Achievement Award.