SN Computer Science (2022) 3:321
https://doi.org/10.1007/s42979-022-01219-5

ORIGINAL RESEARCH

Tight-ES-TRNG: Improved Construction and Robustness Analysis

Itamar Levi[1] · **Davide Bellizia**[2] · **François-Xavier Standaert**[2]

Received: 27 March 2020 / Accepted: 19 May 2022

Abstract
Recently in CHES-2018 Yang et al. demonstrated a very low cost and high performance true random number generator (TRNG) dubbed ES-TRNG. The main novelty of this class of TRNGs is in the methodology of extracting entropy from the accumulated phase jitter, i.e., by using a mechanism of repeatedly sample high-speed clock-edges with high resolution. In this manuscript, we demonstrate how it is possible to increase the number of edges in a cycle (with a very low cost) such that edges accommodate more and more from the full distribution of the phase jitter (this is where the "tightness" is coming from). By utilizing this mechanism we are able to reduce the number of required "repeated samples" (as compared to the ES-TRNG) and to substantially increase the achievable entropy level. We show how it is possible to fine-grain balance the implemented Ring-Oscillators (ROs) periods on FPGAs by using specialized constraints, such as controlling LUTs inputs and distance between elements. We evaluate the validity of our design with the NIST SP800-90B entropy evaluation suite and support the results with a stochastic model which augments the model of Yang et al. to take into account our new design characteristics. The proposed design is able to achieve 5.6 Mbps with an estimated (worst-case) min-entropy level of 0.88 bits—without post-processing (on the raw samples). On the same platform and under the same conditions (i.e. without post-processing), the ES-TRNG was able to maximally produce 1.6 Mbps with min-entropy of 0.5. The manuscript is concluded with a cautionary note and robustness analysis of this class of TRNGs. We demonstrate how dangerous is the affect of parameters such as the external temperature, slow drifts in the power supply voltage, and transient noise (due to logic activity). In essence, we show how small drifts in these parameters concretely reduce both efficiency and the estimated min-entropy levels of the TRNG.

Keywords Entropy source · FPGA · High-performance · Low-area · Stochastic model · TRNG · True random number generator · Robustness

Introduction

True random noise generators (TRNGs) exploit and extract randomness from physical noise sources in electronic systems. Broadly (and traditionally), they are classified according to the mechanisms from which the noise is extracted. The two main groups are the meta-stable-based [1–7] and phase-jitter based generators [8–14]. The first class aims to amplify very small voltage/current noise in a metastable structure (e.g. back-to-back inverter pair, memory or flip-flop elements), and to extract the final stable value while resetting the system between consecutive calls. Due to design challenges such as mismatch and variations, and physical sensitivities such as signal-integrity and cross-talk, such extractors require considerable post-processing and significant health-test mechanisms like aggressive offset cancellation, serial de-correlation and bias removal (e.g. see [1, 6, 7]) which though effective results in very high cost (e.g. in energy and performance). The second class of generators, which are in the focus of this work, aim to extract randomness from noise in the time domain. Many examples exist for oscillator based TRNGs structures. Already in 2007 Sunar et al. [9] demonstrated a multiple Ring-Oscillator (RO) based structure (later denoted as MURO) accompanied with

✉ Itamar Levi
itamar.levi@biu.ac.il

Davide Bellizia
davide.bellizia@uclouvain.be

François-Xavier Standaert
fstandae@uclouvain.be

[1] Bar-Ilan University, Ramat Gan, Israel

[2] Université catholique de Louvain, Ottignies-Louvain-la-Neuve, Belgium

Published online: 08 June 2022

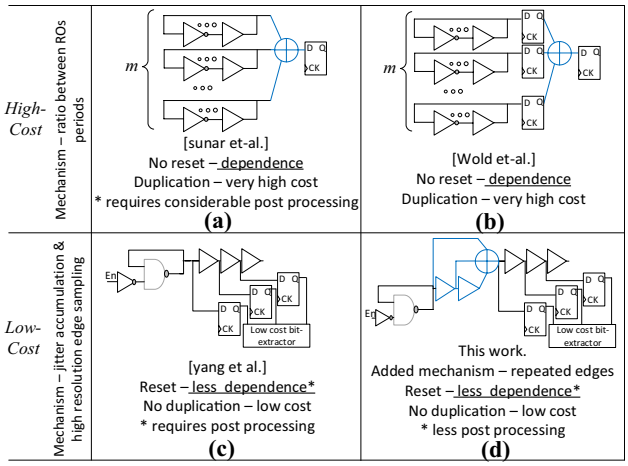

Fig. 1 TRNG schemes: **a** XORing independent oscillators—high cost [9]; **b** XORing independent oscillators with pre-sample—high cost [10]; **c** ES-TRNG—low cost construction [15]; **d** This work—improved Tight-ES-TRNG

a stochastic model, as illustrated in Fig. 1a. If the m oscillators are identical (have same period) and independent, i.e. independently accumulate phase jitter, it was argued that the XORed output of the independent oscillators will be close to uniform. However, practical values showed that quite a large number of oscillators (m) were required to meet these requirements and in practical designs each of the oscillators were instantiated with a large period (many delay elements), both contributing to the high relative cost of this solution. Moreover, in later works Wold et al. [10] showed that due to physical limitations (and the large required values of m), the generator suffers from considerable dependence between subsequent bits and bias. They added to the structure a sampling layer, as illustrated in Fig. 1b, which resolved most of the aforementioned dependence and bias shortcomings. However, the relative high cost of the structure, e.g. in area and energy, remained.

Quite recently, in CHES-2018 Yang et al. proposed the ES-TRNG structure [15] which also extracts randomness from phase jitter in the time domain. They utilize several unique circuitry mechanisms which enabled a design of a very low-area TRNG. It was shown to reach comparable performance (bit-rates) as prior-art. As illustrated in Fig. 1c, a series of delayed versions of the oscillator output is generated by a delay line, these are sampled by another oscillator, repeatedly (repeated sampling). The multiple "snapshots" along with a minimal processing logic enabled capturing and decoding the (noisy) oscillator edge with high-resolution. Another important feature was the Reset strobe between extracted bits (En signal in the scheme) which aids in bits-independence claims.

In this work, we are concerned with ultra low cost structures. i.e. circuits which can generate maximum bit-rates (performance) with minimal area-cost. One typical motivation for such a research is the distribution of randomness (e.g. local spatial security mechanisms or providing masks for masking gadgets and shuffling countermeasures [16–20]), or to provide a very large amount of randomness on a resources constrained application (energy, performance and area or their combination). Therefore, our goal in this research is to enhance the good characteristics of this low-cost class of generators (i.e. the ES-TRNG). To do so, we are combining and utilizing concepts from [9, 10] along with design optimization efforts and FPGA circuit-level enhancements. Simply put, we build a circuit mechanism to increase the number of *edges* in a cycle with a very low cost. Our goal is that edges will populate more from the full distribution of the phase jitter (i.e. "tightness"). The outcome is a (1) reduced generator bias, (2) reduced number of "repeated samples" and (3) smaller required phase-jitter accumulation time. As a consequence of the reduced-bias, substantially increased min-entropy levels and bit-rates are achieved. We show how it is possible to fine/gross-grain balance the implemented Ring-Oscillators (ROs) periods on FPGAs by using specialized constraints, such as controlling LUTs inputs and distance between elements.

Despite the concrete improvements the proposed TRNG provide (while working in nominal conditions), we add another contribution in this manuscript while demonstrating scenarios which can easily jeopardize the quality and efficiency of this class of TRNGs. For this purpose, the manuscript is concluded with an emphasis (or cautionary-note) regarding the large affect of: the external temperature, slow drifts in the power supply voltage, and fast transient noise (due to logic activity). The contribution we have in that direction is mainly to demonstrate how big is the challenge to assure reliable operation when these parameters vary maliciously or passively; whilst this type of analysis was not performed in most FPGA-based prior-art. The robustness-analysis performed demonstrate that this set of parameters must be taken into account in the TRNG design-cycle, by the health-tests performed, and the control-and-correction mechanisms for all adversarial scenarios (e.g. assuming a regulated device or not).

Paper Organization The manuscript starts with introducing the main objectives of the research and the relevant prior art. We follow with briefly surveying the structure, operation and the stochastic model of the ES-TRNG which is of special interest in this manuscript. We then present our contributions starting with an intuitive description and structure of the Tight-ES-TRNG. After we layout the main principles, we discuss the special design-knobs we utilize, circuit-level improvements and fine-grain optimizations. We then follow the necessary step of updating the TRNG stochastic model. This provides us with a basis to discuss the experiments,

where we evaluate the Tight-ES-TRNG architecture implemented on Xilinx Spartan 6 FPGA. This last section includes an in-depth examination of sensitivities to changes in environmental factors, external and internal (which can be malicious or not) and different design factors. These sections are complemented with background information on the designs we evaluate, our measurement and evaluation setups and the tools used in this manuscript. The subsequent section concludes the manuscript.

Background

In this section, we recall the structure, operation, and a brief description of the necessary parts of the ES-TRNG and its corresponding stochastic model which serves as the baseline comparison object in this work.

Notations In this manuscript, variables are denoted with capital letters, sampled values with lowercase letters, functions with sans serif fonts and vectors with bold letters. We use standard notations for mean (μ), standard deviation (σ), and we denote with $f_{\mu,\sigma}(x)$ the probability density of a normal distribution $\mathcal{N}(\mu, \sigma^2)$ with the random variable X with realization x. With $\Pr(x \leq \alpha)$ we denote the (cumulative) probability and with $\Pr(x \leq \alpha | \beta)$ we denote the conditional (cumulative) probability with standard notations for parameters (α and β). Shannon entropy, the min-entropy and an estimation of the min-entropy are denoted by H_1, H_∞ and $\hat{H}_\infty$, respectively. As estimating the min-entropy is controversial and not a simple task, it is typically done by a complex set of tools and tests such as the NIST package SP-800 90B [21] which we will use in this work (and commonly is used).

ES-TRNG Construction and Operation

We recall the main characteristics of the structure and operation of the ES-TRNG [15] while trying to keep only what is required for this manuscript.

Figure 2 shows the basic building blocks of the ES-TRNG. Trying to keep the same notations, the main (sampled) oscillator and the sampling oscillator are denoted by Osc1 and Osc2 along with their respective periods, T01 and T02. From Osc1 we generate a set of delayed versions of the oscillator output by a delay line (denoted by Osc1_d and Osc1_d2 in the figure). These are sampled with the fast sampling clock Osc2,[1] repeatedly. On this set of sampled values, a logical processing takes place (as depicted in the small look-up table embedded in the figure) to generate the

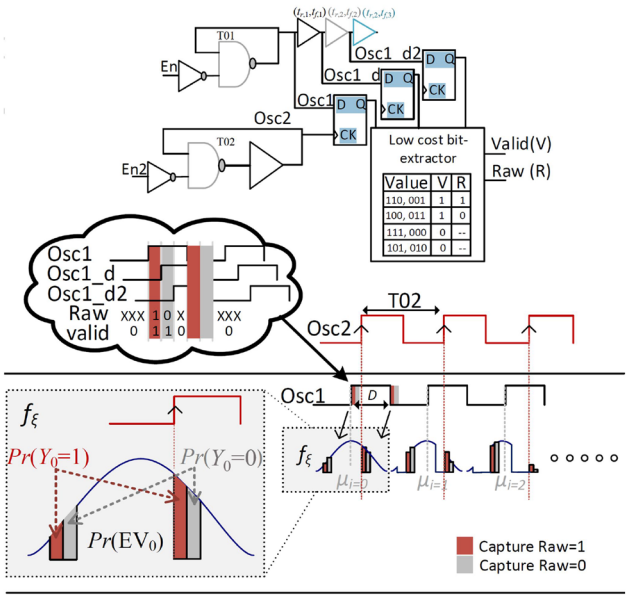

Fig. 2 ES-TRNG schemes and operation (illustration)

raw random bit (R) and an additional valid (V) signal. This mechanism enables the high-resolution of capturing and decoding the (noisy) clock edge. I.e., if a rising/falling-edge of Osc1 occurs very-close to the rising-edge of Osc2, a Valid Raw bit of '1' will be decoded and captured (this behavior is illustrated in the cloud-like zoom-in waveforms on the figure). Likewise, if it will rise a bit further away from the rising-edge of Osc2 a Valid Raw bit of '0' will be decoded and captured. Capturing a '1' and a '0' are symbolized by a light red and gray, respectively on the figure. Otherwise, (i.e. if the change in Osc1 was further away from the rising-edge of Osc2), the sample is not valid and the sampling process will be repeated until reaching the "high-resolution" zone. Between Valid bits, a Reset strobe is signaled to both oscillators (En and En2 signals in the scheme). This mechanism is important to increase consecutive bits independence (see the cautionary note in "Entropy Comparison" on this topic) and to control the Reset time of Osc2 (by En2) to match the required accumulation time of jitter, denoted by t_A (to keep the same vocabulary as in [15]). The phase of Osc1 is denoted by ξ ($\xi \in (0, 1]$), and the phase distribution is denoted by f_ξ. As typically done in this class of accumulated jitter based TRNGs, we assume that the phase distribution is Gaussian. Clearly, as t_A increases, the accumulated phase jitter variance (or σ^2) increases. After the reset strobe of Osc2 has been released (and enough phase jitter was accumulated, owing to a large t_A), the distribution of the phase, f_ξ will be sampled by Osc2 (as shown in the lower part of Fig. 2). The mean value of the distribution in the first sampling cycle ($i = 0$), μ_0, depends on many factors. Some factors are deterministic, such as routing delays and setup-,rise- or fall-times

[1] We interchangeably use the terms clock and oscillator where the intention is clear from the context.

of gates and flip-flops, and some are not. We list several examples for the second case: (1) low-frequency environmental factors such as the external temperature and power supply voltage drifts (2) the profile of the device usage (e.g. logical activity within the device and around the generator). We stress that these parameters have a significant impact on the generator and we will come back to this point when we discuss robustness and challenges ("Temperature Dependence" and "External Voltage Dependence—Low and High Frequency Noise"). On the Gaussian distribution of the first-cycle, i=0, we denote regions on the figure in which, if the edge (ξ) arrives, it will evolve to the event of a valid '1' (denoted by $Y_0 = 1$) in red (and correspondingly, a valid '0', $Y_0 = 0$ in gray). As defined in [15], the probability of these events is denoted by $\Pr(Y_0 = 1)$ and $\Pr(Y_0 = 0)$, correspondingly. These probabilities are associated with the area under the curve in those regions. The event of a non-valid bit in cycle i is denoted by $\Pr(Ev_i)$, as illustrated in the lower part of the figure for $i = 0$. If, in the first attempt, a valid bit was not captured, then in the next cycle (t+T02) we would ask ourselves what will now be the probability distribution of the phase $f_{\xi}(i = 1)$, that is conditioning the fact that an event Ev_0 took place. The resulting distribution, which excludes the regions of a valid bit from the previous cycle, is illustrated at the bottom of the figure (with a mean value of $\mu_{i=1}$). If again a valid bit was not captured (event Ev_1), the process will iteratively continue.

ES-TRNG Stochastic Model

With the goal of being concise, we recall only necessary parts of the ES-TRNG stochastic model (and relevant variables which we will need to modify and adapt later on).

The rise/fall-times of a delay element j in a chain ($j \in \{1, 2, ...\}$) are denoted by $t_{r/f,j}$ (as shown on the delay elements/buffers, in the top of Fig. 2). The duty-cycle of Osc1 is denoted by D. The normalized values of the rise/fall-times parameters are denoted by $d_{r/f,i} = t_{r/f,i}/T01$. The regions in time in which Osc1's phase is sampled by Osc2, which concludes in a Raw bit={1, 0, non-valid}, are denoted by $\{S_1, S_0$ and $S_N\}$, respectively:

$$S_1 = \bigcup_{k=-\infty}^{\infty} \left\{[k, k + d_{r,1}) \cup [k + D, k + D + d_{f,1})\right\},$$

$$S_0 = \bigcup_{k=-\infty}^{\infty} \left\{[k + d_{r,1}, k + d_{r,1} + d_{r,2})\right\} \cup \left\{[k + D + d_{f,1}, k + D + d_{f,1} + d_{f,2})\right\},$$

$$\text{and, } \mathbb{R} = S_0 \cup S_1 \cup S_N,$$

where k is an integer index. The mapping $g(x) = 1$, when $x \in S_N$ and, $g(x) = 0$, when $x \in S_0 \cup S_1$ is used to denote these regions which are used by the model.

The Gaussian distribution in the first sampling event is denoted by ρ_0, $\rho_0(x) = f_{\mu_0,\sigma_{t_A}}(x)$. The definitions of the events discussed in the previous section, naturally follow: $\Pr(Y_0 = l) = \int_{S_l} \rho_0(x)dx,\ l \in \{0, 1\}$ and $\Pr(Ev_0) = 1 - \Pr(Y_0 = 0) - \Pr(Y_0 = 1)$.

Recalling the illustrative example of the ES-TRNG operation in the previous subsection, the distribution of the phase in the next cycle ($i = 1$), conditioning that Ev_0 has occurred, can be defined by:

$$\rho_{1|Ev_0}(x) = \frac{1}{\Pr(Ev_0)} \int_{-\infty}^{\infty} f_{0,\sigma_{T02}}(x - z) \cdot \rho_0(z - \frac{T02}{T01}) \cdot g(z - \frac{T02}{T01})dz.$$

The intuition behind this formula is that (referring now to the lower part of Fig. 2) if Ev_0 took place, the distribution will be shifted by T02/T01[2] and will be "cleared" of the $\Pr(Y_{i-1} = 0)$ and $\Pr(Y_{i-1} = 1)$ regions of the previous cycle (otherwise Ev_{i-1} would have taken place). This is the reason for the multiplication in the shifted $g(\cdot)$ function. The additional term $f_{0,\sigma_{T02}}$, and the convolution with it, are there to compensate for the fact that during the following period (of T02) more jitter was accumulated. Assuming that the distribution of this superimposed accumulated jitter is normal, its effect over the full distribution is taken into account by convolution. The rest of the conditional probabilities (for consecutive cycles, $i > 1$), $\rho_{i|Ev_{i-1}}(x)$, are derived recursively (and quite similarly to $\rho_{1|Ev_0}(x)$):

$$\rho_{i|Ev_{i-1}}(x) = \frac{\Pr(Ev_{i-2})}{\Pr(Ev_{i-1})} \int_{-\infty}^{\infty} f_{0,\sigma_{T02}}(x - z) \cdot \rho_{i-1|Ev_{i-2}}(z - \frac{T02}{T01}) \cdot g(z - \frac{T02}{T01})dz.$$

And naturally, the computation of $\Pr(Ev_i), i > 0$ is done by:

$$\Pr(Ev_i) = \Pr(Ev_{i-1}) \cdot \int_{S_N} \rho_{i|Ev_{i-1}}(x)dx$$

consequently, the joint probabilities, $\Pr(Ev_{i-1}, Y_i = 1)$, are derived by:

$$\Pr(Ev_{i-1}, Y_i = 1) = \Pr(Ev_{i-1}) \cdot \int_{S_1} \rho_{i|Evi-1}(z)dz.$$

From these, the probabilities of generating a one or a zero by the TRNG are computed as follows:

$$\Pr(b = m) = \Pr(Y_0 = m) + \sum_{k=1}^{\infty} \Pr(Ev_{k-1}, Y_k = m),$$
$$m \in \{0, 1\}.$$

With these at hand, it is immediate to compute the modeled Shannon-Entropy, H_1^m, and the modeled min-Entropy, H_∞^m.

[2] The 'real' shift is of T02 in [s]. In practice, we are only interested in its effect on μ_i which is ((T02/T01)-⌊(T02/T01)⌋)·T01. However, for simplicity we keep the notation of T02/T01 similar to [15].

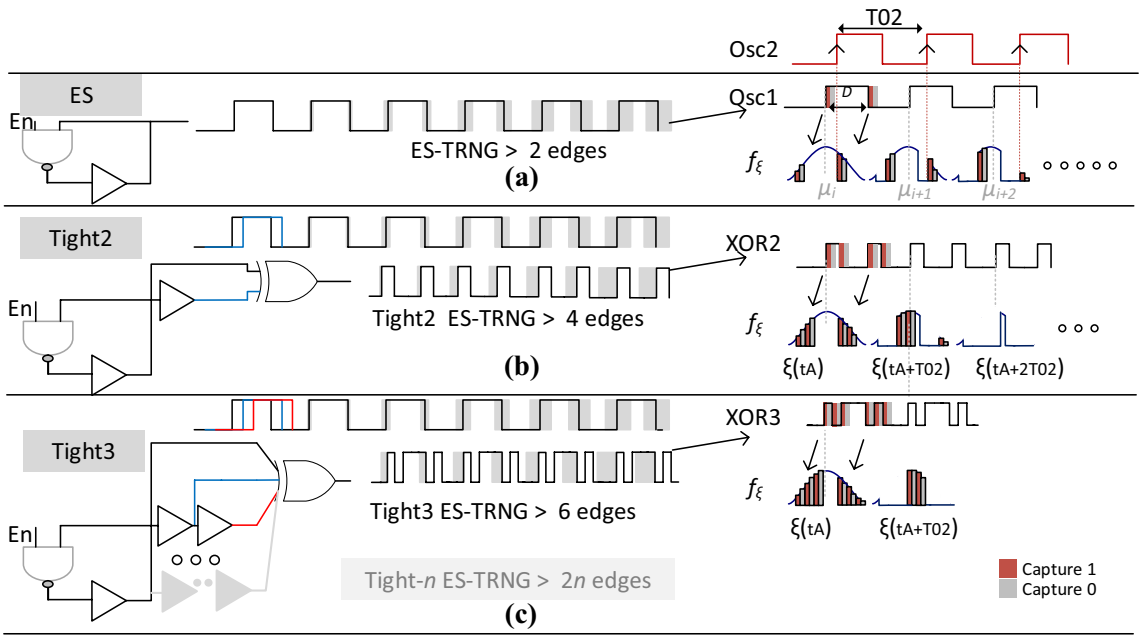

Fig. 3 ES-TRNG based schemes and illustrative phase sampling distributions: **a** initial proposal [15]; **b** Tight2-ES-TRNG; **c** Tight3-ES-TRNG

Tight-ES-TRNG: Simple Structure and Intuition

In this section we detail on the structure and operation mechanism of the Tight-ES-TRNG. As discussed in the introduction, our goal is to build a low-area generator with higher bit-rates and to improve the state-of-the-art and knowledge in this respect.

The logic-level modifications/additions we implement over the ES-TRNG structure are quite small and contribute to the attractiveness and low-area/resources of the Tight-ES-TRNG. The main novelty of the structure is in what we denote as "tightness": we now refer to Fig. 3a, which illustrates the sampled oscillator of the ES-TRNG, the output waveform with its associated accumulated phase-jitter (in gray regions), and the phase probability distribution (f_{ξ_i}). A shortcoming of the ES-TRNG is that when either the accumulated phase jitter is small or large in respect to the $S_1 \cup S_0$ width (of one cycle), a large amount of "repeated" samples would be require to reach a valid raw bit (on a worst-case scenario). Moreover, when the variance is relatively small, the bias might be considerable as $\Pr(Y_i = 0) \neq \Pr(Y_i = 1)$ (we come back to these important points when discussing the stochastic model in "Stochastic Model Update"). Therefore, to equalize the probabilities and remove bias as much as possible, we would need to either increase the accumulated jitter variance or to find a different mechanism for this purpose. This is exactly the goal of the Tight-ES-TRNG. In Fig. 3b we illustrate a simple instantiation of the Tight2-ES-TRNG which adds only one additional delayed replica of the oscillator output and then XORs it with the original. The resulting output waveform is illustrated on the figure. On each time instance in which the oscillator flips, we get a pulse whose width follows the amount of added delay. That is, instead of two edges in a cycle we get four edges. Clearly, this can be generalized as shown in Fig. 3c which displays the Tight3-ES-TRNG, implemented in this manuscript. On each time instance in which the original oscillator flips, we get three edges, where the width between edges is controlled by the added delay-line elements. That is, in this case, instead of two edges in a cycle we get six edges, or generally with n elements we get $2 \cdot n$ edges. When feeding these resulting signals (denoted by XOR2 or XOR3, respectively) to the repeated high-resolution sampling of the ES-TRNG, we in-fact "fill-the-gaps" of the probability distribution with regions of $\Pr(Y_i = 1)$ and $\Pr(Y_i = 0)$. This is illustrated in the right part of Fig. 3b and c, where now multiple "tight" comb-like alternating '1' and '0' regions exists. Clearly, this mechanism potentially has several positive outcomes:

1. Reduced bias. The tighter and more uniform (from both sides of the mean value) the distribution of the multiple $\Pr(Y_i = l), l \in \{0, 1\}$ regions is, the smaller is the bias of the generator.
2. Reduced required number of repeated-samples until a valid event occurs. The coverage of the phase-jitter distribution increases with the increase in the number of XOR inputs (denoted by xr). Consequently, the non-valid probability ($\Pr(Ev_0)$) decreases. In fact, this prob-

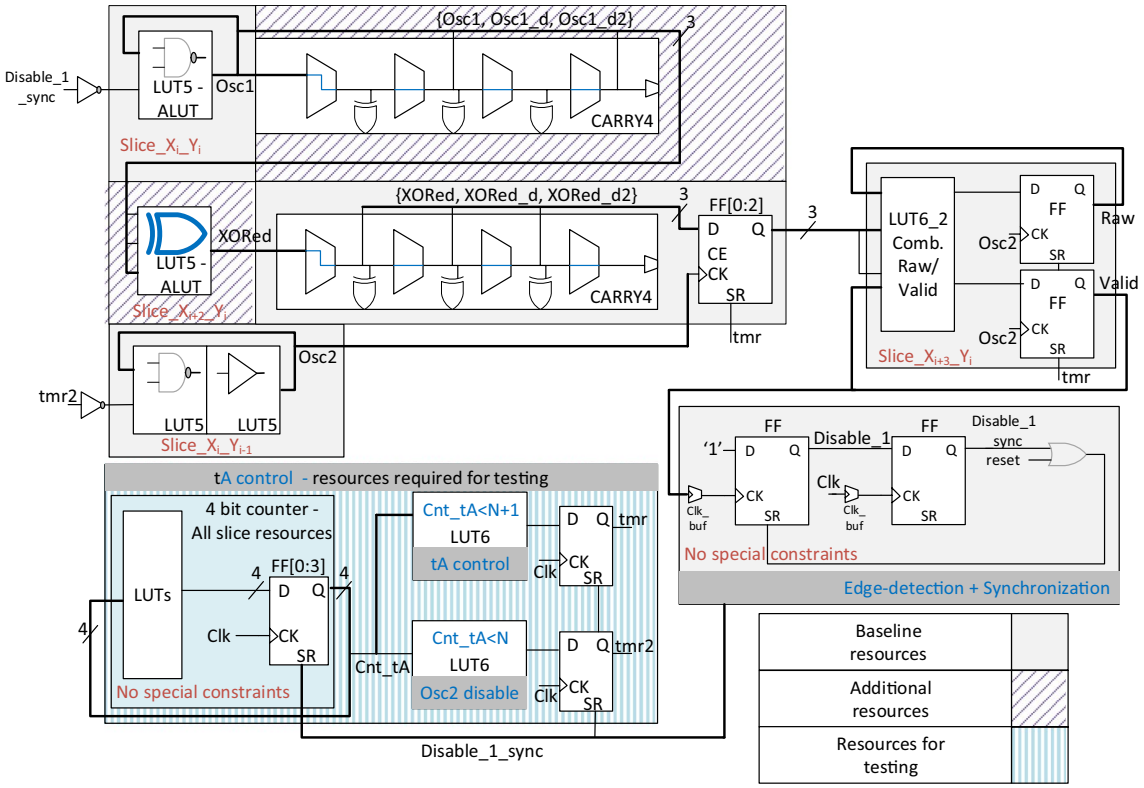

Fig. 4 Tight-ES-TRNG minimal design

ability will vanish to '0' faster (exponentially) with the number of tries as *xr* increases.

3. Smaller required jitter accumulation time (t_A). Due to the previous two points, it is possible to achieve a smaller t_A, either because of the reduced number of repeated samples or due to the reduced bias which can translate to a smaller required jitter variance (for the same bias).

Our approach was therefore to populate the distribution with alternating '1' and '0' regions which is required only in certain scenarios depending on typical electronic delay parameters such as the ones we expect with FPGAs. While this manuscript was evaluated we have also demonstrated that the ES-TRNG implementation on ASIC constructs does not strictly/necessarily require the Tight-ES-TRNG augmentation due the fast signals propagation and small delays [22].

Circuit Level Implementation and Improvements

In this section, we elaborate on the implementation details of the proposed Tight-ES-TRNG instance, on circuit-level improvements and optimization. Starting with a discussion on our *low-resource* and efficient implementation on FPGA, and moving on to discuss our attempts to delicately control the physical parameters and the oscillators periods.

Minimal Design

The logical abstraction of the Tight-ES-TRNG can be implemented in many ways on hardware platforms. However, the results obtained and the efficiency will highly depend on the selection of devices, on placement restrictions and on environmental and technological aspects; especially, on FPGA constructs which are more constrained and less flexible for designers than ASICs. In this manuscript, we elaborate on an implementation over a Xilinx Spartan-6 FPGA which is quite a popular device. In fact, most of the prior-art designs were implemented on the same platform and as a bonus, our results are also comparable with the ES-TRNG which was implemented on a similar device. Due to the sensitivity of the very high frequency signals we have in our system and the sensitive physical phenomena we make use of to extract randomness, it is of high importance to *pack* the design efficiently, constrain all the elements in the design and to reduce unknowns and variability. We next detail on how it was achieved.

We start by partitioning the design to three parts and then we elaborate on each. Referring to Fig. 4: (1) The parts in the implementation which are implemented similarly to [15]

in the logical abstraction, and not necessarily on the implementation specifics, appear in light gray fill (smooth), (2) the additions required to inhabit the Tight-ES-TRNG appear with diagonal purple stripes background and (3) the part of the implementation which was needed for testing and can perhaps be slightly simplified in an industry level implementation appear with vertical light blue stripes background.

1. Baseline—Osc1 was implemented using one 5-input Look-Up-Table (LUT5) of the A type (ALUT) which logically represent a NAND2 element. The delay line to sample the XORed signal (which serves as the Osc1 signal in the ES-TRNG construction) was constructed with the fast CARRY4 (dedicated carry propagation chains) with three taps ({XORed, XORed_d, XORed_d2}). These three taps were sampled with the four available flip-flops in the same slice by Osc2 which is asynchronously reset low by the *tmr* signal. Osc2 was implemented using only two LUT5 elements (a NAND2 and a buffer, BUF), placed in the same FPGA slice. These sampled signals were used by one LUT6_2 instance to generate the Valid and Raw signals which were again sampled by two available flip-flops (all in the next horizontal slice) by Osc2 and were also asynchronously reset low by the *tmr* signal. The sampled versions of the Valid and Raw signals were fed back to the same LUT6_2 to aid in computing the next Valid and Raw. The valid signal then passes directly to an edge-detection and synchronization module which was implemented as such: due to the (required) fast rise-time of the Valid signal and the fact that it needs to pass through the LUT6_2 in less than T02 (plus a setup-time) it was defined as a generated *clock* signal and was driven through a clock-buffer to a clock pin to detect a rise transition (asynchronous edge-detection) which triggered the Disable_1 signal. This is done to synchronize the extracted values with the system clock (or to further accumulate raw bits, utilize them and perform if needed more processing) it was again sampled to generate Disable_1_sync which was then ORed with the system reset strobe to reset back the edge-detection after more than 1 system clock cycles (and less than 2). Therefore the Raw bit is able to be captured with the system clock. The Disable_1_sync was used as a deterministic duration (one cycle) reset strobe of Osc1.
2. Tight-ES-TRNG additions—The Osc1 signal was delayed by another fast CARRY4 chain to generate two additional taps ({Osc1, Osc1_d, Osc1_d2}). We discuss these delays and the required balancing of these taps as compared to the other CARRY4 delay line in the next section. These three taps were directly routed (with minimal and constrained paths) to the ALUT available in the next horizontal FPGA Slice. In this LUT, a minimal delay XOR3 was implemented (by careful LUTs inputs selection) to generate the XORed signal.
3. Control and testing—The accumulated jitter control circuitry is needed for testing, design-modularity and flexibility. However, we do highlight that this mechanism is needed in full on a real-life system where additional health-tests and correction mechanisms (see "Temperature Dependence" and "External Voltage Dependence—Low and High Frequency Noise" for such cases) are embedded in the design to account for environmental noise such as temperature and voltage drifts, internal noise such as coupling-noise and signal-integrity, and physical variations such as stress and device degradation. Therefore, we elaborate on the implementation of which and the cost which are important and were not fully considered in [15]. The accumulated jitter (t_A) control circuitry was constructed with a 4-bit counter which was asynchronously reset by the Disable_1_sync signal to generate the Cnt_tA signal. The counter value was continuously monitored. While it was smaller than a determined value (say $N + 1$) it generated the *tmr* signal which resets the high-resolution edge sampling mechanism and the Raw/Valid circuitry. It is important to note that due to the pipeline nature of this data-path, these elements will reset only after the Raw bits were already captured. An illustrative waveform diagram of the control loop, with all the asynchronous and synchronous triggers, is provided in Appendix A— Fig. 16. Another comparator, which is set to the value of N, was implemented to trigger another signal, *tmr2*. This signal is used to reset Osc2. Basically, the combination of *tmr* and *tmr*2 control the t_A time with a resolution of a system clock cycle. The area resources needed to implement this feedback loop are four LUT6_2 elements and six Flip-Flops, which is significant as compared to the rest of the design. This cost can be reduced if a less fine-grain control resolution is required or a specific t_A value is known in advance. However, and as we elaborate in detail in "Entropy Comparison"'s cautionary-note, such a high resolution is in our opinion anyway needed along with additional control mechanisms to assure sufficient health, temper resistance and robustness of the generator.

Post-Place-and-Route Gate-Level Simulations on a Xilinx Spartan-6

All instances were manually placed, implemented with dedicated FPGA-instances, were predefined and specified to use dedicated primitives-inputs for specified signals and prevent the synthesizer, mapper and placer from performing optimizations, merging LUTs, removing logic, changing locations or swapping pins. To do so, a decent

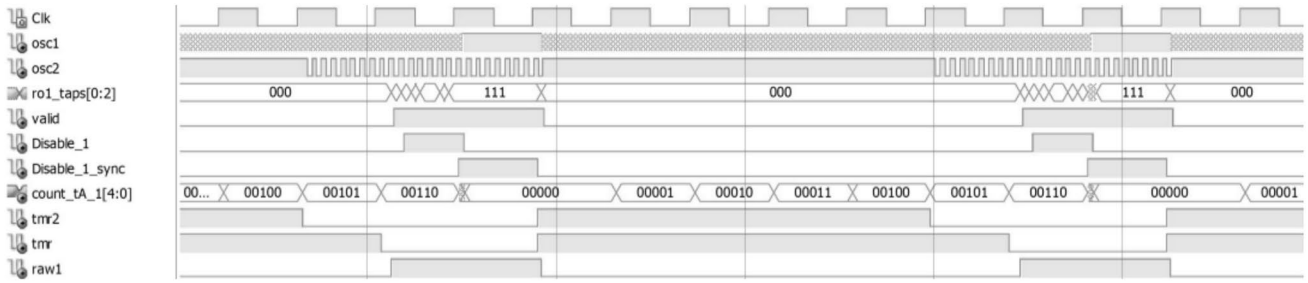

Fig. 5 Timing and control waveform from post-place-and-route simulation on a Xilinx Spartan-6 of the proposed Tight-ES-TRNG in nominal conditions

mixture of {*LOC*, *LOCK_PINS*, *BEL*, *INIT*, *MAP*} design constraints were used (both in the main hardware-description of the design and in the constraint *.ucf* file), side by side with additional regional placement constraints (e.g. to place other high-frequency system level counters and shifters further away from the oscillators), and also {*make hirarchy*, *boolean signal true*, *mark_debug*, *keep signal*, *keep_hierarchy*} design attributes and macro-hardening limitations [23].

In this subsection, we provide functional investigation of the proposed architecture. I.e., we verify the logical activity on gate-level post-route simulation environment with nominal conditions. Nevertheless, in this manuscript we stress more on experimental results ("FPGA Experiments"), which are anyway more indicative. This is due to the fact that it is impossible/not fully supported/not accurate to simulate most of the situations we test on the manuscript: that is, tuning the simulated power noise-level, sweeping the DC voltage and the temperature-range far outside of the nominal device spec. are not fully characterized by the simulation models and libraries vendor; such models only exist with spice-level ASIC simulators, noisetran-based models etc. Moreover, in the models-libraries for simulation typically worst-case delays are listed.

The presented stochastic model and circuit analysis in the manuscript are all done based on the assumption of perfect digital signals, but given the proposed types of digital circuits used (e.g. a very short combinatorial feedback loop), we need to make sure that the signal will behave close enough to a rail-to-rail digital signal, will not suppress short glitches or filter them out, especially at the output of the XOR gate. To validate the correctness of the stochastic model and the analysis, we perform such (analog) circuit simulation. Nevertheless, our results demonstrate full functionality in nominal conditions and as shown below on actual hardware, the design provide superior results (e.g. higher entropy and high performance) as compared to the prior-art.

As shown in Fig. 5 the above described mechanisms are fully functional on gate-level post-place-and-route (sdf-based) simulation-environment in nominal voltage and temperature conditions. Both oscillators, disable, reset and synchronization mechanisms are fully functional (digital) and operate correctly, with delay information provided for the gates and routing. Note that the behavior perfectly match the illustrative waveform diagram of the control loop (with all the asynchronous and synchronous triggers), which is provided in Appendix A— Fig. 16.

Fine-Grain Oscillators and Delay-Line Constraints

Several additional design mechanisms which we utilize to delicately control the physical parameters and the oscillators' periods include: (1) LUTs inputs control and (2) delay balancing.

As shown in Fig. 6, by combining constraints (i.e. {*LOC*, *BEL*, ...}) and setting several guided design attributes we were able to control specifically the in-out path through a LUT5 which was used to form an oscillator. In an abstract view, a LUT is comprised of some programmable storing elements and a MUX-tree. Together they form the required programmable functionality. On the physical layers this type of structure can be implemented with numerous device level elements and techniques (e.g. CMOS-based, Pass-Transistor based, PLA-like, with/without a pre/post drivers and with/without a restorer to enable fast slew-rates, and many other options). The actual schematic details and the layouts are unknown (and are in fact a very protected intellectual-property of the vendors). Therefore, we can only try to build general hypothesis and intuition such as: *as the index of a LUT input increases the delay of the LUT decreases*, and try to check and falsify it. This is exactly the case tested and shown on the figure. We build multiple oscillator chains, while using specific inputs, and feed the outputs back to the inputs, with minimal distances between elements, to form oscillators. We then use a clock-divider mechanism to down-sample the oscillator periods and tapped counter. From the values obtained we performed an average period calculation over many cycles. Figure 6 shows the projected and approximated average periods of oscillators with one and

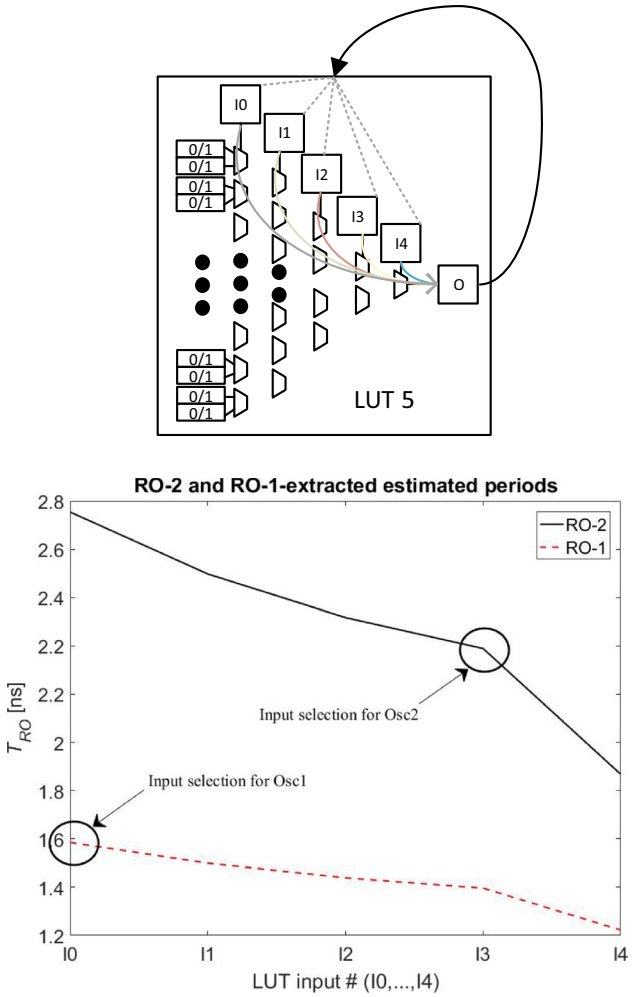

Fig. 6 Fine-grain oscillator constraints with constrained LUTs inputs

two elements (RO1 and RO2) while changing the selected inputs in the LUT5 (I0–I4). Looking at the trends of Osc2 and Osc1 (RO2 and RO1), clearly our hypothesis generally makes sense as we get a monotonic behavior. We further see that input selection is an efficient knob to set and balance oscillator periods. An important note is that in this experiment we tried to minimize routing delays, which can generally take up more than 70% of the delay. This was also the case with the actual oscillators constructed where routing-delay was estimated to range between 20 and 60% in our range of experiments. Broadly speaking these results are clearly not (and can never be) accurate, however, they do reveal concrete trends and values. The large impact of this factor on (e.g.) the entropy levels of the TRNG will be shown in the experimental section of the manuscript.

The second element which we aimed to carefully balance was the added delay chain taps ({Osc1, Osc1_d, Osc1_d2}), which should match the high-resolution edge-sampling delay-line mechanism (which taps {XORed, XORed_d, XORed_d2}). Figure 7 illustrates the two delay lines, the Osc1 oscillator implemented by an ALUT element, close to the beginning of the Slice's carry-chain, as well as the XOR3 (also an ALUT) which generated the XORed signal. The figure also illustrates a waveform diagram of all these signals. The main and clear observation is that delays on the Osc1 line should be larger than 2 times the delays on the XORed line. This is to achieve a "Tight" lattice of $\Pr(Y_i = 1)$ and $\Pr(Y_i = 0)$ regions (as discussed in "Tight-ES-TRNG—Simple Structure and Intuition"). Clearly, as these elements are *by-design* deterministic and repetitive across the FPGA mesh, a good starting point would be to tap the Osc1 line every two elements, and the XORed line every one element. After multiple examinations and attempts in the final design we thus tapped the outputs of the 1st carry chain every two elements to have some margin on both the delay from Osc1 to Osc1_d and from Osc1_d to Osc1_d2. To keep the XORed line delays minimal we tapped starting from the second element. A nice property is that as the two carry chains are close by and quite well matched (and in general a replica of the same circuitry) many environmental and logical influences will affect the two lines in the same way. Depending on the actual period of Osc1 this can lead to a very tight scheduling of Raw=1 and Raw=0 regions, as illustrated on the *Raw_interpret* illustrative waveform (Fig. 7)[3].

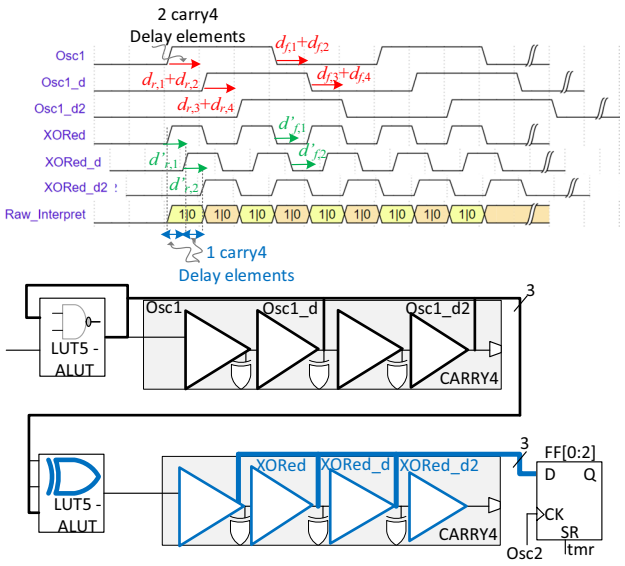

Fig. 7 Tight-ES-TRNG delay balancing

[3] However, it also induces tight timing margins, which under environmental changes should be maintained—see the cautionary-note in "Temperature Dependence" and "External Voltage Dependence—Low and High Frequency Noise".

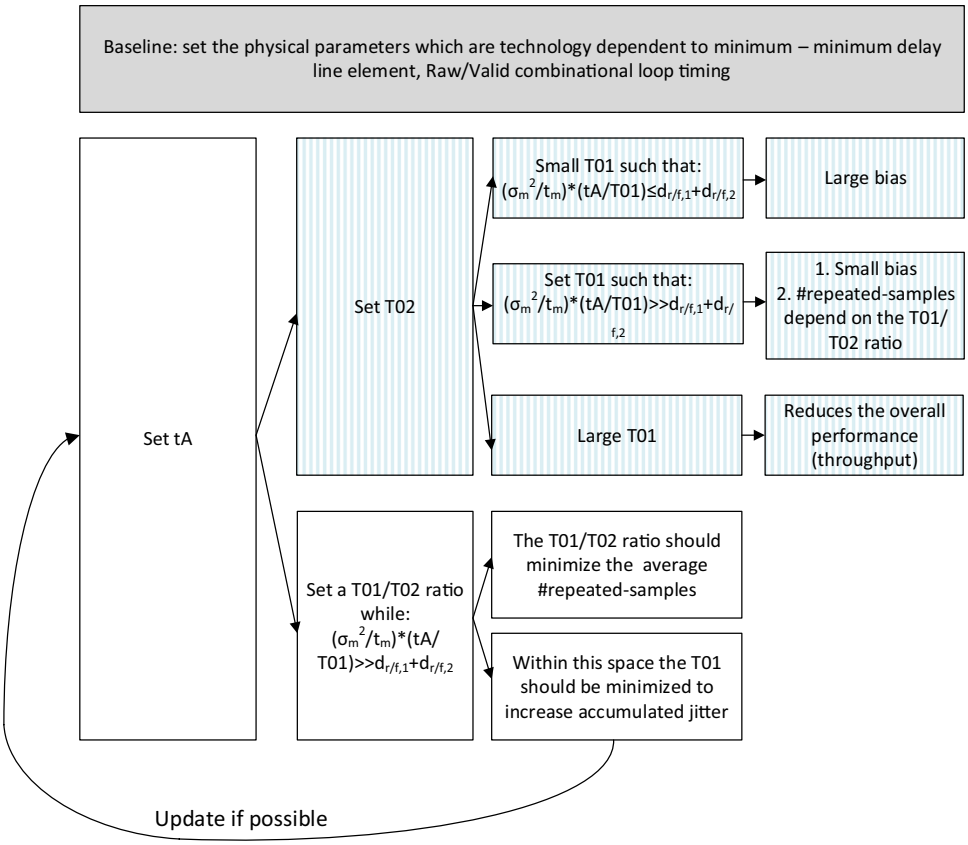

Fig. 8 The Tight/-ES-TRNG optimization space and parameters setting

Fine-Grain Period Control with Spatial Constraints

Our aim was to fine grain control the periods of the oscillators and the design parameters. To do so we first tried to load the oscillators (Osc2 and Osc1) with a controllable capacitive parasitic loading (by routing oscillator signals to other LUTs or by increasing routes lengths), hoping that it will subtly balance oscillators periods etc. However, for Osc1 it considerably reduced the accumulated noise and increased unwanted behavior due to small slew-rates and large capacitances; for Osc2 any attempt to load it with even the smallest capacitive loading failed and induced a considerable reduction in the entropy level (for the same reasons). We concluded that capacitive loading on such a platform is too gross to make the subtle changes required.

In the same direction, instead of loading the oscillators we tried to control the spatial locations of oscillator instances and the distance between them with the finest resolution we were able to achieve. I.e. attempting to maintain the drive-strength (current driving ability) of the chain elements albeit subtly increasing the delay. For this purpose, we focused on Osc2 (as in our instantiation it incorporates two instances, an AND and a BUF). While keeping the AND location set in an ALUT (placed in the lower part of one FPGA slice), we sweep the location of the BUF element LUT, first in the same slice (BLUT, CLUT and DLUT) and then in the proximate slice (ALUT, BLUT, CLUT and DLUT). We elaborate on the results of the experiment and the consequences in "FPGA Experiments".

Stochastic Model: Discussion and Updates

The ES-TRNG optimization problem is a multi-variable complex and non-trivial problem which is very much constraint-driven (especially over an FPGA non-flexible platform). Before specifying some of the design parameters selection and as a motivation for the chosen values, we discuss the main knobs and trade-offs they offer.

Figure 8 illustrates the design space and the methodology which was taken in this work to set and calibrate parameters. Starting from the top of the figure, in gray, we relate to parameters which are extremely technology-dependent and that we can affect only by element selection on our FPGA platform. That is, with a very limited resolution. Those include, delay line element, delays on combinational paths and sequential elements timing etc. Minimizing these parameters will imply that Osc1 and Osc2 can optionally be faster and with a smaller *foot-print* for our design, therefore they were set to minimum. Moving on from the baseline components, we assume we target a performance goal (i.e., we set a clear target for t_A). From this point, one approach

Table 1 Measured parameters, used by the statistical model, Xilinx Spartan-6 FPGA, room-temperature, V_{DD}=1.2[V]

Parameter	ES	Tight-ES	Param value	ES	Tight-ES
T01	1.595	1.595	T02	2.192	2.192
$t_{r,1}$	0.025	0.025	$t'_{r,1}$	–	0.032
$t_{f,1}$	0.040	0.040	$t'_{f,1}$	–	0.043
$t_{r,2}$	0.027	0.027	$t'_{r,2}$	–	0.036
$t_{f,2}$	0.043	0.043	$t'_{f,2}$	–	0.049
D	0.39	0.41	$t'_{r,3}$	–	0.035
σ_m^2/t_m	$14 \cdot 10^{-15}$ [s]	$14 \cdot 10^{-15}$ [s]	$t'_{f,3}$	–	0.048
–			$t'_{r,4}$	–	0.04
T_{clk}	20	20	$t'_{f,4}$	–	0.051

Values are in units of *ns* unless otherwise stated

(denoted by vertical blue stripes filling on the figure) would be to also set T02 period. Once these two parameters are set, we will need to choose T01. Three regions of values for T01 under this setting exists: (1) if we set T01 small enough such that, $(\sigma_m^2/t_m) \cdot (t_A/T01) \leq (d_{r/f,1} + d_{r/f,2})$ (where, $(\sigma_m^2 \cdot t_A)/(t_m \cdot T01)$ is the accumulated jitter noise variance [15]), it will lead to a large bias as the probabilities $\Pr(Y_i = 1)$ and $\Pr(Y_i = 0)$ will be unbalanced. In the extreme, it might be the case that $\Pr(Y_i = 1) = 1$, as the entire distribution fits within the $(0, d_{r/f,1}]$-region. (2) If we set T01 large enough such that, $(\sigma_m^2/t_m) \cdot (t_A/T01) \gg (d_{r/f,1} + d_{r/f,2})$, naturally the bias will reduce. However, the number of repeated samples required to reach a Valid (#repeated-samples), will be affected as it is mainly influenced by the T01/T02 ratio. In practice, very small changes in this ratio might induce relatively large changes in #repeated-samples (e.g. 1X-100X). In turn, this might have a considerable effect which will reduce the performance and bit-rate of the design. (3) Setting T01 larger will again reduce the performance.

An alternative approach that will minimize the danger of over-constraining parameters (and is the approach that we took in this manuscript) is illustrated in the blank filling boxes on the figure. We first set a "soft" target for t_A, we follow by making sure that $(\sigma_m^2/t_m) \cdot (t_A/T01) \gg (d_{r/f,1} + d_{r/f,2})$ takes place while setting a T01/T02 ratio. Our goal is to find a ratio that would minimize #repeated-samples. We stress that generally setting a delay ratio is easier than specific value on an FPGA device where the flexibility is limited (e.g. by ratio of number of elements). After finding a candidate ratio, we can try to further minimize T01 (in proportion to T02). This assists in increasing the accumulated jitter rate. In this case, we can go back and update (iteratively) the t_A goal.

While taking this approach we set parameters as illustrated in Table 1 and measured them according to a set of approaches from [15, 24–26]. It is important to note that exploring the entire parameters space is very exhaustive and time consuming: on the one hand using a set of parameters and evaluating the outcome of the stochastic model repeatedly is very slow as the model is iterative (serial) and on the other hand, every parameter change in the design induces a long process of characterizing the effective value of the parameter by experiments. Therefore, exhausting the entire design space is very hard and we merely provide one parameters-set scenario (after several iterations and optimizations). The methods which we used to measure the oscillators periods and the delay line rise/fall times are well described in [27]: for oscillators period we adopt the ripple-counter based approach and for the delay-lines we mimicked the long chain tapping approach with repeated sampling.

We further note, that the values we provide for oscillators periods are only estimations and that they can never be precisely evaluated. Any interference (sensing, buffering or sampling of the oscillators) induces a change in the evaluated parameters. In practice any mechanism we add for characterization should still exist in the circuit (or the main part of it) if we desire our estimations to resemble more to the actual values. Therefore, and as shown in the measurement results section we always evaluate the influence of a parameter change with the final outcome of the TRNG as a whole. Regarding the results of the stochastic model while asserting characterized parameters: we do highlight that as characterization issues are anyway common (in any digital system) and the bias which they might induce is not negligible for our application, a safe-margin on parameters values should be explored with the stochastic model, the resulting estimation, H_∞^m should represent the minimal achievable value.

Table 1 lists the measured parameters used by the stochastic model for the evaluated Spartan-6 Xilinx platform in room-temperature with the nominal power supply voltage. All the values in the table are in units of *ns* unless otherwise stated. According to the flow illustrated in Fig. 8 we first set a (reasonable) ratio for T01/T02=0.75. That is, one which we know we can get quite close to (see Fig. 6) with a small enough t_A and a rather small T01 (1.595 *ns*) to increase the accumulated jitter. The final periods of Osc1 and Osc2 were grossly set by minimizing routing and with the LUTs input selection. A fine tuning was made with LUTs distance for Osc2 and further minimizing routing loads. This resulted in a ratio of T01/T02≈0.72. The values are reported in the table. Note that for both the ES-TRNG (comparison) and for the Tight-ES-TRNG, the values are the same: this is done both for comparison purposes and in-fact also yielded the best results we achieved on both designs. The delay-line values of the XORed delay line were rather close to the values reported in [15]. However, a rather slight increase in all values exists which we associate with the different environmental parameters or

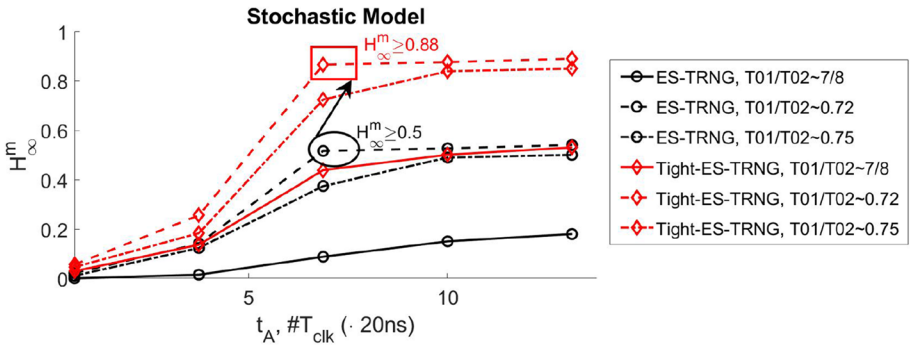

Fig. 9 Stochastic model estimate—Tight-ES-TRNG and ES-TRNG H^m_∞ comparison

measurement evaluation parameters (device mismatch and manufacturing corner, the differences in the measurement board and/or power supplies and regulation). On the other hand, relating to the Osc1 delay-line here we actually must modify (slightly) the circuit to perform measurements: Osc1 taps go into a combinational path (a XOR LUT) and are not directly sampled (as the case for the XORed delay line, see Fig. 7). Therefore, to evaluate this delay line we shortened the three XOR-ALUT inputs, through the slice (internally) directly to three flip-flops. We acknowledge this might induce a bias on the measurements however, we do believe: (a) such a bias in any case (and as the results show), would be small as the delays are slice-internal, (b) in case of fear from a large bias, a worst-case routing loading of the delay-line of Osc1 can take place.

Stochastic Model Update

The changes needed to be performed in the ES-TRNG stochastic model to take into account the modified scheme of the Tight-ES-TRNG design are limited. In fact, we only need to update the regions of time in which the XOR_ed delay-line is sampled by Osc2 (instead of the Osc1 delay-line) and concludes in a Raw bit={1, 0, *non-valid*}. We recall these regions were denoted by $\{S_1, S_0$ and $S_N\}$, respectively. We use the same notation of $d_{r/f,l}$, $l \in \{1:4\}$ for the first delay line (over Osc1) and we also define the notation $d'_{r/f,l}$, $l \in \{1:4\}$ for the second delay line, operating on XORed signal (please see Fig. 7 for a more visual illustration with the relevant timing parameters). In our case, starting with S_1, we add the relevant time regions to get:

$$S_1^T = S_1(t) + S_1(t + D + d'_{r,1} + d'_{r,2}) + S_1(t + \textstyle\sum_{h=1}^{4} d'_{r,h}).$$
$$S_0^T = S_0(t) + S_0(t + D + d'_{f,1} + d'_{f,2}) + S_0(t \textstyle\sum_{h=1}^{4} d'_{f,h}),$$

where, the T denotes the Tight-ES-TRNG construction. Naturally, S_N^T follows from these definitions. Note, that a special care should be taken to make sure that the following timing constraints are met:

$$d_{r/f,1} + d_{r/f,2} + \epsilon^1_{r/f} \geq d'_{r/f,1} + d'_{r/f,2}.$$
$$d_{r/f,3} + d_{r/f,4} + \epsilon^2_{r/f} \geq d'_{r/f,1} + d'_{r/f,2}.$$

Here, ϵ^i is a margin parameter which should be tuned with (e.g.) parasitic-, wiring- or logical loads. If these constraints are not met, the resulting bits bias will increase. The Gaussian distribution in the first sampling event, $\rho_0(x)$ and all the following conditional distributions ($\rho_{i|Ev_{i-1}}(x)$, $i > 0$) and the previously defined probabilities and events do not change (i.e. $\Pr(Y_0 = l), l \in \{0, 1\}$, $\Pr(Ev_i)$). That is, these are the only modifications required for the stochastic model.

The results obtained by the two models (ES-TRNG, and the slightly tweaked version of the Tight-ES-TRNG) are compared in the following for the same set of parameters (as defined in Table 1):

Figure 9 shows the stochastic model estimated min-entropy ($\hat{H}^m_\infty$) as a function of t_A. The circle-marked curves-set corresponds to the ES-TRNG model and the diamond-marked curves-set corresponds to the Tight-ES-TRNG updated model. In each set we show the results while varying T01 in a small range such that T01/T02={7/8, 0.72, 0.75} (T02 is set according to the value in Table 1). The actual (measured) value for T01/T02 on our setup was 0.72. The values shown on the figure reassures our expectations that the Tight-ES-TRNG construction increase the asymptotic min-entropy levels. An investigation of the trends also shows that a rather small variation in the periods-ratios will significantly affect the results. This is demonstrated experimentally in the following. In addition, the average number of repeated samples needed to capture a valid bit was shown to be smaller by a factor of 10–50 for the range of scenarios evaluated. This implies that with the Tight-ES-TRNG architecture the performance bottle-neck is never the repeated-sampling mechanism. The values in the figure were evaluated while sweeping over μ_0 and selecting the value which yields the worst-case (as discussed in [15], "ES-TRNG Construction and Operation" and more elaborated in "FPGA Experiments"); the actual value set was $\mu_0 = 0.13 \cdot T02$. In practice, the model results showed that small changes in μ_0 (of $< 10\%$ of T01) leads to concrete changes in the entropy

levels (of $10-20\%$). These sensitivities as well as the sensitivities of the periods-ratio are discussed more in details in "FPGA Experiments").

FPGA Experiments

In the experimental part of the paper we first layout the baseline conditions of our evaluation environment, discuss the evaluation tool used and follow with experimental results.

Our goal is first to evaluate a worst-case scenario for the TRNG and second, to clearly list the conditions of the experiments for comparability and repeatability. To evaluate a worst case scenario, we need to make sure the device embeds as minimal as possible interfaces and instances and that the TRNG is in-fact quite isolated (and not affected by capacitive coupling from other elements). Clearly the larger the logical-activity on the device, the noisier the power supply voltage become (or alternatively the current dissipation). In turn, this might result in optimistic evaluation of entropy levels of our TRNG (as we demonstrate at the end of this section). Moreover, such noise sources are not modeled in the stochastic model and might induce a large difference between the stochastic model (e.g. H_∞^m) and the experiments (e.g. $\hat{H}_\infty$) which is generally not wanted. In this respect, our device contained only the above specified logical elements of the Tight-ES-TRNG and an additional shift register and a minimal UART interface for communication (placed on the far-end of the device from the TRNG, to reduce voltage noise within the same power-region). For all experiments, unless otherwise stated, we operate the device with nominal power supply voltage ($V_{DD} = 1.2V$) and in room-temperature. In dedicated sections we clearly specify: (1) the monitored temperature (2) DC power supply voltage and (3) the added Gaussian noise variance. These, as we show next, might lead to a large entropy over-estimation.

For controlling the power supply voltage in room-temperature we use the on board regulator embedded in the Sakura-G evaluation board used [28], and in sections where we wanted a highly regulated and controlled environment, we connected an external low-noise power supply directly to the embedded Spartan-6 device.

Before discussing the experimental results we elaborate on the $\hat{H}_\infty$ evaluation methodology: we use the NIST SP (special publication) 800-90B test suite [21] which is one of the most common tools for entropy estimation and in addition it elaborates on how to build, test, and evaluate entropy sources. On the evaluation side the suite provides two tracks, the *i.i.d* (independent and identically distributed) and the non-*i.i.d* track. In this work due to the active reset of the oscillators we perform between extraction of subsequent raw-bits we have an *i.i.d* claim in the stochastic model. However, under many circumstances the resulting sequences

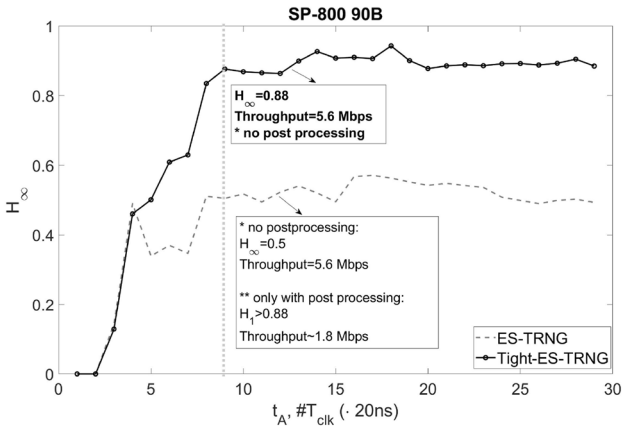

Fig. 10 Tight-ES-TRNG and ES-TRNG SP800-90B estimated min-entropy comparison

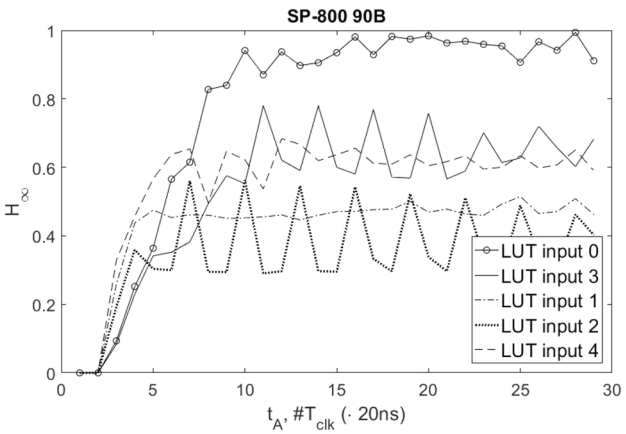

Fig. 11 Tight-ES-TRNG SP800-90B entropy comparison—changes in LUTs inputs

from an "internally"-*i.i.d* source can become dependent (e.g. due to slow and external physical and electrical phenomena, environmental effects and sampling effects). In our set of experiments and tests we passed the *i.i.d*-track tests for the final designs only (as expected) when t_A was sufficiently large such that the estimated entropy levels were rather stable and high. In all other cases (i.e. small values for t_A) the results in the figures below show the min-entropy estimate derived from the non-*i.i.d* track. For example, in Figs. 10, 11 and 12, in all scenarios, once $t_A > 8 \cdot T_{clk}$ all the values represents the *i.i.d* estimated values. Rarely did we find cases which did not pass the *i.i.d* tests for large enough t_A values, and only when the device temperature and supply voltage were not carefully monitored (i.e. in the temperature chamber and with an independent power supply regulator). In any case, even in these rare scenarios, when we performed minimal post-processing, such as XORing every two consecutive bits etc. as done in [29], we always passed the *i.i.d* tests

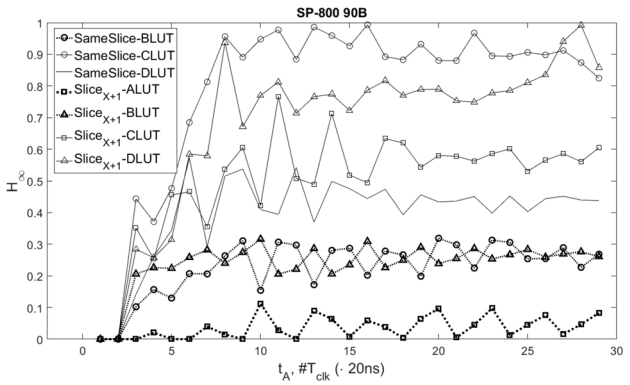

Fig. 12 Tight-ES-TRNG entropy comparison—changes in spacing and locations of ROs elements

which explains the results in [15]. As a general note, on the methodological point-of-view we highlight that we believe a fair comparison point between entropy sources is while comparing the outcome raw values without post-processing. In addition, we always perform both the SP 800-90B described (1) *restart* tests on 1000 independent sequences of 1000 sequential generated bits (with a full reset and power off between chunks) and (2) *sequential* tests of $1 \cdot 10^6$ sequences (from the same session). The values shown in the figures represent the outcome of the min-entropy worst-case estimation following both tests, and providing the more pessimistic value. As recommended in the latest standards which require a stochastic model to accompany the design, we take the minimal estimation between the model and the tests.

Finally, each of the tests results shown below represent the **minimum** value got over ten repeated experiments. Ideally, the SP 800-90B tests min-entropy estimation should represent a rather worst-case evaluation and one experiment should be enough for the purpose. However, to make sure the values we get do not deviate greatly we performed these ten iterations which confirmed a very small variance in estimations. In practice, the SP 800-90B tests lasted more than a day to process and output an estimation on a standard platform therefore achieving more than ten is already hard. An exemplary *i.i.d*-sequential test results of the SP 800-90B is provided in Table 3 in Appendix A.

Entropy Comparison

We start our evaluation from a base-line comparison of the Tight-ES-TRNG and the ES-TRNG in the following: In Fig. 10 we show the estimation $\hat{H}_\infty$ versus t_A for the two designs under the same setting: exactly the same Osc1 and Osc2 (same location, inputs and wiring), same device and surrounding logic and same environmental factors. We first see that (without post processing) both the Tight-ES-TRNG and the ES-TRNG achieves rather stable values with $9 \cdot T_{clk} = 180$ ns. However, the main achievement/improvement of the Tight-ES-TRNG is that the minimal raw values he reaches are $\hat{H}_\infty = 0.88$ when, $min_{t_A} > 9 \cdot T_{clk}$ with a throughput of 5.6 Mbps as compared to $\hat{H}_\infty = 0.5$ when, $min_{t_A} > 9 \cdot T_{clk}$ with the same throughput. This is clearly significant because the entropy levels of the ES-TRNG are asymptotic (an inherent bias exists) which can only be reduced with post-processing and significant performance reduction [29]. As an example, following the same line of post-processing done in [15] (by consecutive bits XORing) we were able to increase the Shannon-entropy to the same level of > 0.88 (by XORing every 3 consecutive bits) with a throughput reduction to 1.8 Mbps.

As discussed in "Fine-Grain Oscillators and Delay-Line Constraints" and "Fine-Grain Period Control with Spatial Constraints" we explored and tried to fine tune the physical parameters in our platform to improve our design. We now discuss the efficiency of the discussed knobs and the delicate nature of optimizing these designs. The resulting entropy levels of the Tight-ES-TRNG while changing the selected inputs of Osc2 in the LUT5 (I0–I5) are shown in Fig. 11. It is possible to observe that:

- T02/T01 ratio sensitivity: once a period is set for Osc1, changing the period of Osc2 (even slightly) greatly affect the resulting entropy levels.
- t_A sensitivity and periodicity: as discussed in "ES-TRNG Construction and Operation" the mean value of the jitter distribution in the first sampling cycle ($i = 0$), μ_0, is significant to affect the resulting entropy levels, the entropy-source bias and the #repeated-cycles. It depends on many factors, deterministic (routing delays etc.) and not-deterministic (time-dependent processes, coupling, etc.). In the figure we clearly see that first, generally as t_A increases the entropy level increases. However, with larger resolution on-top of this behavior, we see entropy periodicity versus t_A (more clearly on inputs I3 and I2). We relate these observations to μ_0 which depends on t_A: different t_A values are controlled by counter mechanisms (or any other timer circuitry) which will trigger different logical paths to enable Osc2 for different t_A values and by which vary μ_0.

We next inspect how delicate the tuning of Osc2 period is in regards with delay-elements location control. While keeping the AND of Osc2 location set in an ALUT (placed in the lower part of one FPGA slice), we sweep the location of the BUF(fer) element, first in the same slice (BLUT, CLUT and DLUT) and then in the proximate slice (ALUT, BLUT, CLUT and DLUT). The resulting estimated min-entropy levels of these designs is shown in Fig. 12. It is clearly observed that, as expected, careful element positioning is important

(in addition to concrete inputs selection to specified LUTs, as shown in Fig. 11) and can lead to a drastic changes in min-entropy levels. It is quite hard to draw general conclusions regarding the actual positions and how they affect the outcome as it depends on the interconnect routing between elements. In our case, we first performed a gross selection of LUT location (A- and C-LUTs), then LUTs inputs selection (I0 and I4) and only as a final stage carefully controlled interconnect routing to fine-tune the Tight-ES-TRNG parameters.

Temperature Dependence

In this and the following sections we evaluate the TRNG sensitivity to the external temperature, DC power supply voltage and to the added Gaussian noise variance. All are important security ingredients for many reasons:

1. Security under assumptions—we denote by "under assumptions" a scenario in which the device (TRNG embedded) is assumed to be controlled and regulated. In this manuscript we specifically discuss the next set of variables[4]: power-supply voltage (DC and variance), the local in-die and external temperature. In this case (regulation), the sensitivity-factors will considerably impact the cost of the regulation circuitry (and its required specifications), the monitoring, and correction mechanisms needed in the device. Therefore their analysis is very important.
2. Security without assumptions—clearly for such a security claim the device should be insensitive to the discussed factors and therefore such analysis is of paramount importance.

To control the power supply voltage DC and the noise variance we used the Agilent AG33250A waveform generator with the ranges of: $V_{DD} \in \{1.1, 1.2, 1.3\}$ DC [V] and the Gaussian noise standard deviation, $\sigma_n \in \{0 : 10 : 50\}$ [mV]. For temperature control we use the ESPEC SH-261 temperature and humidity chamber while sweeping across temperatures of $\{10 : 2.5 : 30\}°$ C. In order to keep the discussion and the results demonstrated in realistic temperature range which can be experienced in normal operation (and does not necessarily imply an adversarial activity), we examine the device on a rather restricted range. In addition, in this temperature range we are able to clearly state that ambiguity in the results is not due to problematic specs of other components in the system or on-board etc. (e.g. the SAKURA-G board used is not guaranteed to work with too

[4] Though, more variables do exist and they relate to the adversarial set of assumptions and the security model, e.g. the clock frequency.

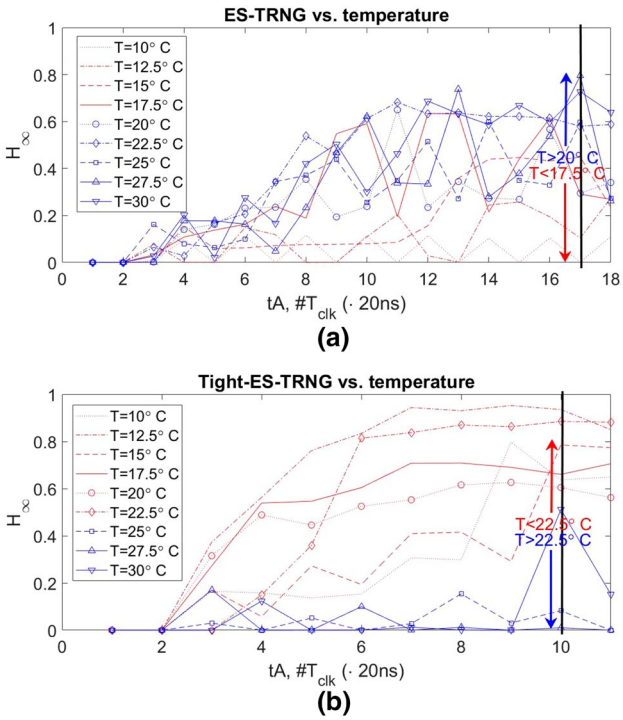

Fig. 13 $\hat{H}_\infty$ vs. temperature and t_A: **a** ES-TRNG, **b** Tight-ES-TRNG

extreme temperatures, although the Spartan-6 Xilinx device is characterized to work in quite extreme conditions).

Figure 13 shows the estimated min-entropy levels of the ES-TRNG and the Tight-ES-TRNG versus t_A and for the specified range of temperatures. While examining Fig. 13a we see that in low-temperatures, which imply low accumulated jitter values, larger t_A is required (a considerable change). In addition, in these low-temperatures (the red set of curves, $< 17° C$), another observation is that the min-entropy levels are also lower (as expected). In (relatively) high-temperatures, we observe higher min-entropy levels and a smaller required t_A (due to larger thermal noise and consequently larger jitter noise). Based on these observations we come to three conclusions:

- Either a concrete and reliable temperature compensation and control is required or a considerable margin on t_A should be taken (which might considerably reduce performance).
- Even if a margin on t_A is taken, the possible min-entropy degradation (with temperature lowering) will imply that in addition, a compensation is needed. For example, down sampling and filtering (e.g. XORing more and more bits etc.).
- Owing to the last two points we emphasize that continuous health-tests and temperature monitoring and compensation are required and it is very hazardous not to take

it into account (which was the case in most published prior-art).

Referring now to Fig. 13b of the Tight-ES-TRNG design temperature examination, it is possible to observe quite a reversed effect due to the time-domain "Tightness" and other timing constraints: in relatively low-temperatures (10° C) the min-entropy levels are low and large t_A is required (due to small phase jitter). As the temperature increase to the low-medium range, the min-entropy rapidly increases (exhibiting quite small t_A values). This phenomena is observed in much smaller temperatures than the ES-TRNG design, in-fact due to the tightness in time (smaller jitter variance is needed to reduce the bias). In high-temperatures (and rapidly), the min-entropy drops to the verge of failures due to timing-failures which we attribute to the delicate nature of balancing the two CARRY4 chains. The (now) very fast delays make the pulses smaller than the $\Pr(Y_i = 1) \cup \Pr(Y_i = 0)$ region required.

We reach one concrete and main conclusion and a corresponding cautionary-note: both designs exhibit temperature dependence which must be monitored, reacted upon and compensated. Under all adversarial scenarios and assumptions (and given the examined range of temperatures), the aforementioned mechanisms are required (i.e. a warm summer and a moderate winter put the design in jeopardy).

External Voltage Dependence: Low and High Frequency Noise

In this subsection we want to emulate two scenarios: (1) low frequency voltage drifts which might happen due to different (slow) work-loads and activity factors and environmental factors—to do so, we sweep the DC voltage level of the device which might also correspond to an adversarial activity; (2) high frequency voltage noise which is generated internally in electronic systems due to logic-activity near the TRNG (e.g. consider a typical scenario of an AES spread around the TRNG), to do so we sweep the standard-deviation of additive Gaussian noise on the power supply DC voltage.

Starting with the first scenario, Fig. 14 shows the results obtained while sweeping the power supply voltage in the range of $V_{DD} \in \{1.1,\ 1.2,\ 1.3\}$ [V] with a fixed temperature of $T = 21.5\,°C$. It is demonstrated that in the nominal level (1.2V) the TRNGs behave as expected (for comparison see Fig. 10 which the only difference is that now the temperature is monitored). However, reducing/increasing the power supply voltage by as little as 100mV, substantially reduces the min-entropy levels.

As discussed above another important factor we believe should be characterized is the TRNG's quality under power supply noise. This examination should be performed to

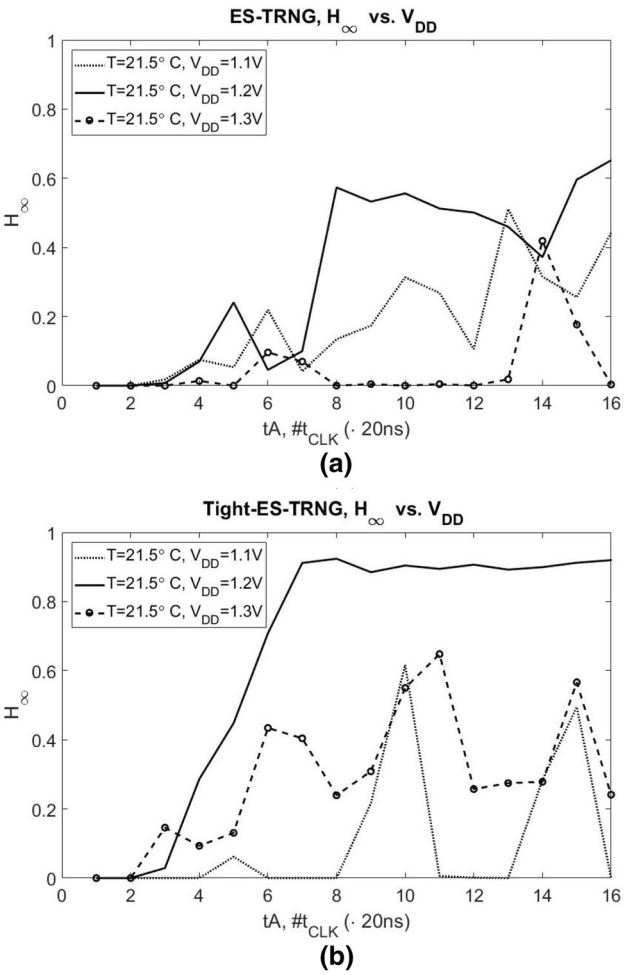

Fig. 14 $\hat{H}_\infty$ vs. V_{DD} and t_A: **a** ES-TRNG, **b** Tight-ES-TRNG

simulate scenarios of low/high logical activity within the device (different activity factors and workloads of other hardware components), we vary the additive Gaussian noise standard deviation, $\sigma_n \in \{0 : 10 : 50\}$ [mV] over the nominal DC of 1.2V. Figure 15 shows the results of this experiment. Clearly, for both designs (ES-TRNG in Fig. 15a and Tight-ES-TRNG in Fig. 15b), as σ_n increase, both t_A decrease and the min-entropy level increase. The important take-away message from this experiment is the importance of characterizing the TRNG with as little as possible additional elements and logic activity on the device to reduce the danger of min-entropy over-estimation.

ASIC solutions for voltage, temperature and noise variations are rather well explored. For monitoring meta-stable based TRNGs solutions one such example is the two-step coarse/fine-grained tune with a self-calibrating feedback loop which was proposed by Srinivasan et al. already in 2010 to compensate for temperature and voltage variations in 45nm ASIC technology [6]. Several quite recent examples for edge-sampling TRNGs are: configurable tune-able loop

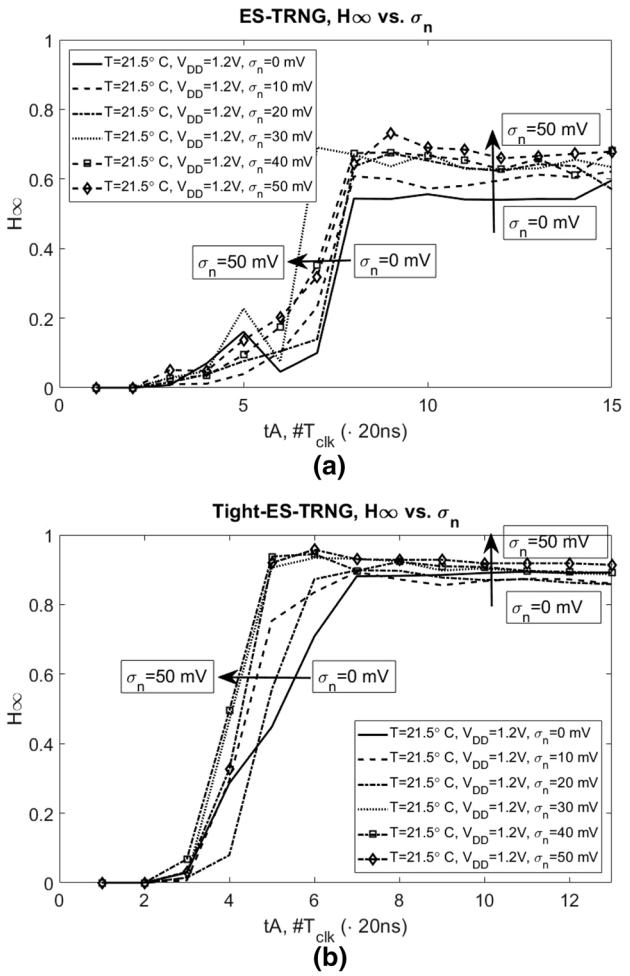

Fig. 15 $\hat{H}_\infty$ vs. σ_N and t_A: **a** ES-TRNG, **b** Tight-ES-TRNG

to provide robustness across a wide range of temperature and voltages, process variation in 28nm and 60nm processes by Yang et al. [34]. Later on they also proposed a low-cost noise and voltage monitoring on-chip in 40nm ASIC technology [35]. Lately a robust true-random-generator with voltage, process and temperature monitoring was introduced by Kim et-al. in 65nm technology [36]. It is important to note though that none of these solutions was well supported by a stochastic model which is required by the latest certification tests even though some of which did pass some entropy tests e.g. the SP 800-90B *sequential*-test.

The challenge or an open question that remains is how efficient it is (let alone possible) to implement such solutions on FPGA platforms which are more restricted and limited in design abilities. The goal of this section is to acknowledge and stress the importance of this line of research to answer these challenge on FPGA platforms. In this work we focus on FPGAs which are extensively investigated for a range of low-cost and high-performance applications which can considerably benefit from TRNGs (e.g. IoTs). Even-though, it is important to highlight that most of the limitations we face with FPGAs do not exist/can be further mitigated on ASICs, therefore stressing the attractiveness of our design for ASICs. i.e. on FPGAs we have less design granularity and flexibility, more predefined architecture and it is impossible to integrate low-cost analog/custom blocks for (e.g.) Process-Voltage-Temperature (PVT) compensation.

Comparison with the State-Of-The-Art

In Table 2 we list the most relevant state-of-the-art (SOTA) designs which target a (rather) low-cost implementation, have been implemented on FPGAs,[5] and are supported by a stochastic model. As such a comparison was already performed in full in CHES 2018 (ES-TRNG manuscript) for FPGA based solutions, we only highlight differences and stress important points.

The table lists ring-oscillator based designs: the two *matched* ring oscillators based ERO design with one freq.-divider([24, 30]), the two *matched* ring oscillators based COSO design with coherent sampling ([24, 31]), the PLL architecture which is a similar approach to COSO, albeit with two independent PLLs' instead of ROs' ([24, 32]), the transition-effect ring oscillator design TERO ([24, 33]), and the ES-TRNG.

For all designs we elaborate on the: resources required in terms of LUTs, FFs, dedicated PLL resources and CARRY4 elements, performance in Mbits/s, estimated min-entropy level and we indicate weather it comes from the same SP 800-90B evaluation suite as in this manuscript (denoted by *). The second column in the table relates to comments regarding the existence of a stochastic model, relative area resources requirements (Low, Medium and Large) and if the design require Manual-Routing (MR) or Manual-Placement (MP). Finally, in the last column we specify whether the design was evaluated on *other-than-nominal* conditions.

Several general observations are that the performance of the ERO, COSO, PLL and TERO are considerably smaller, especially the ERO and PLL designs which in-fact considerably down sample the oscillations frequency and require many FFs or use two expensive PLLs. One might claim that this performance reduction trades-off high entropy levels, however, the reported values in the table for these designs were evaluated through the procedure B of the AIS-20/31 standard which is quite different than the methodology taken in this manuscript (and for the ES-TRNG design). Namely, by using the SP 800-90B NIST standard. Therefore, for min-entropy estimation and comparison we mainly focus the discussion as compared to the ES-TRNG design.

Several specific observations are that the performance of the proposed design is ~5-fold than the SOTA while also

[5] In fact, all have been implemented on a Xilinx Spartan-6 device.

Table 2 Comparison With State-Of-The-Art TRNGs

All designs -Spartan 6	Comments	Resources	Bit-Rate [Mbits/s]	$\hat{H}_\infty$ *-SP 800-90B	Examined conditions
ERO[24, 30]	- Medium area - Stoch. Model - MP and MR	49 LUTs 19 FFs	0.0042	0.999	Nominal only
COSO[24, 31]	- Low area - Stoch. Model - MP and MR	18 LUTs 3 FFs	0.54	0.999	Nominal only
PLL[24, 32]	- Large area - Stoch. Model	34 LUTs 14 FFs 2 indep. PLLs	0.44	0.981	Nominal only
TERO[24, 33]	- Medium area - Stoch. Model - MP and MR	39 LUTs 12 FFs	0.625	0.999	Nominal only
ES-TRNG[15]	- Low area - Stoc. Model - MP	10 LUTs + 5FFs + 1 CARRY4 + (control) 6 LUTs + 6 FFs	1.15	0.5* - (raw) no post-process	Nominal only
Tight-ES-TRNG (this work)	**- Low area - Stoch. Model - MP**	**5 LUTs + 7 FFs + 2 CARRY4 + (control) 6 LUTs + 6 FFs**	**5.6**	**0.88* - (raw) no post-process**	**- Additive AC noise - DC Voltage sweep - Ext. temperature drift**

Bold highlights results achieved in this work

achieving considerably higher min entropy levels (on the raw and not post-processed samples) in nominal conditions. The resources cost of the Tight- and -ES-TRNG designs is quite close and they both require only manual-placement, which is quite standard and easy to achieve on all FPGA platforms. Finally, the sole design which was evaluated with extreme conditions (static and dynamic voltage noise, and temperature drifts), is the proposed design and the slightly modified ES-TRNG implemented both in this manuscript.

Conclusions

In this manuscript we introduce the Tight-ES-TRNG. We pick-up some of the remaining challenges of phase-jitter based TRNGs from prior-art. We demonstrate how it is possible to increase the number of sampled clock-edges in a cycle (with a very low cost) such that edges accommodate more and more from the full distribution of the phase jitter (denoted by "tightness"). By this mechanism we are able to substantially increase the achievable entropy levels and to reduce the number of required "repeated samples" to reach the so-called high-resolution region. We further demonstrate how it is possible to gross- and fine-grain balance design parameters such as the implemented Ring-Oscillators (ROs) periods on FPGAs by using specialized constraints. The proposed design is able to achieve 5.6 Mbps with an estimated (worst-case) min-entropy level of 0.88 bits— without post-processing (on the raw samples) which significantly advance the efficiency of the TRNG as compared to the ES-TRNG (while keeping the area attractiveness as compared to [9, 10]). We conclude the manuscript with a quite extensive robustness analysis of this TRNGs class. We further supplement our analysis by examining several practical scenarios of parameters drift (deliberate or unintentional) such as the external temperature, slow drifts in the power supply voltage, and transient noise (e.g. due to logic activity). In essence, we show how small drifts in these parameters concretely reduce both efficiency and the estimated min-entropy levels of the TRNG. The open question that we leave for further investigation is how efficient mechanisms of continuous health-tests, temperature monitoring and control (which were more explored for ASIC technology) are on FPGA platforms (which are clearly more restricted).

Appendix A: Additional Results and Illustrations

In this Appendix we give one example for the SP 800-90B *i.i.d*-track *sequential* test results which process $1 \cdot 10^6$ raw samples from the Tight-ES-TRNG in nominal conditions (as referenced in the manuscript). As discussed above, for each experimental point {voltage, noise-level, temperature, t_A} etc. up to 10 similar experiments were performed and the reports on the figures in the manuscript represent the worst (min) entropy levels. In addition, we provide here Fig. 16 which illustrate the timing diagram of the different control

Table 3 An exemplary SP 800-90B suite results *i.i.d*-track *sequential* test over a sequence of $1{\cdot}10^6$ binary elements

Statistic	C[i][0]	C[i][1]	Statistic	C[i][0]	C[i][1]
Excursion	651	0	Covariance(8)	573	0
numDir.Runs	6095	24	covariance(16)	6940	2
lenDir.Runs	2639	3836	covariance(32)	9161	0
numIncr.Decr.	7244	29	compression	1685	50
numRunsMed.	764	60	**Passed IID permutation tests		
lenRunsMed.	9343	12	Chi square independence		
avgCollision	4807	4	score=1931.9	DOF=2047	cut-off=2250.43
maxCollision	3696	900	**Passed chi-square independence test		
periodicity(1)	4128	35	Chi square goodness-of-fit		
periodicity(2)	7341	27	score=5.18988	DOF=9	cut-off=27.877
periodicity(8)	7476	15	**Passed chi-square goodness-of-fit test		
periodicity(16)	7810	27	LRS test		
periodicity(32)	8494	29	W: 37	Pr(E>=1): 0.973724	
covariance(1)	134	1	**Passed LRS test		
covariance(2)	8199	6	IID = True	min-entropy = 0.991657	

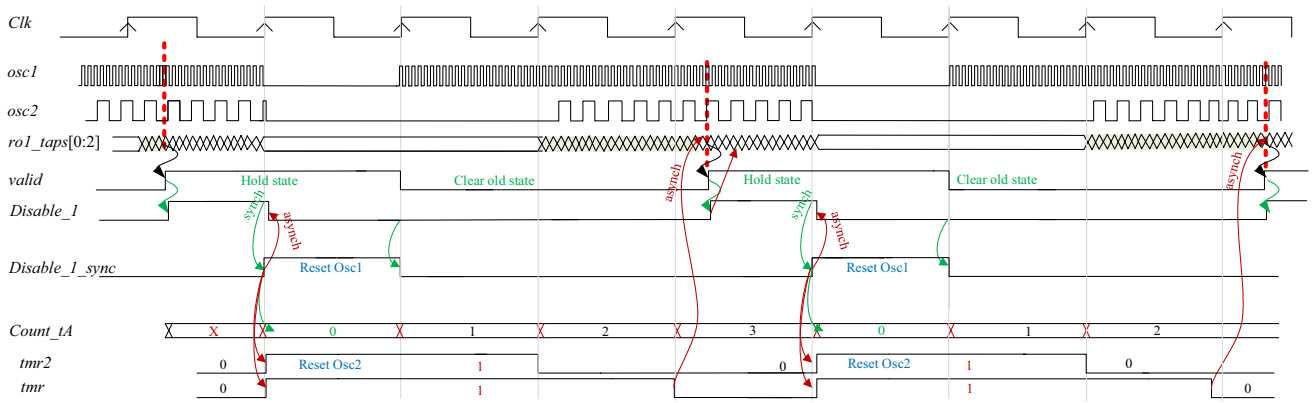

Fig. 16 Control circuitry illustrative timing-diagram

signals in the design (as discussed in "Tight-ES-TRNG—Simple Structure and Intuition").

Funding This research was supported by H2020 European Research Council (Grant 724725) and Israel Science Foundation (ISF) (Grant 2569/21).

Declarations

Conflict of Interest On behalf of all authors, the corresponding author states that there are no conflict of interest.

References

1. Parker RJ. Entropy justification for metastability based nondeterministic random bit generator. In 2017 IEEE 2nd International Verification and Security Workshop (IVSW), pp 25–30. IEEE, 2017.
2. Vasyltsov I, Hambardzumyan E, Kim Y-S, and Karpinskyy B. Fast digital TRNG based on metastable ring oscillator. In International Workshop on Cryptographic Hardware and Embedded Systems. Springer, 2008, pp 164–80.
3. Suresh VB and Burleson WP. Entropy extraction in metastability-based TRNG. In 2010 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST), pp. 135–40. IEEE, 2010.
4. Wieczorek PZ, Gołofit K. Dual-metastability time-competitive true random number generator. IEEE Trans Circ Syst I Regul Pap. 2014;61(1):134–45.
5. Tokunaga C, Blaauw D, Mudge T. True random number generator with a metastability-based quality control. IEEE J Solid-State Circ. 2008;43(1):78–85.

6. Srinivasan S, Mathew S, Ramanarayanan R, Sheikh F, Anders M, Kaul H, Erraguntla V, Krishnamurthy R, and Taylor G. 2.4 GHz 7mW all-digital PVT-variation tolerant true random number generator in 45 nm CMOS. In 2010 Symposium on VLSI Circuits, pp. 203–4. IEEE, 2010.
7. Mathew SK, Johnston D, Satpathy S, Suresh V, Newman P, Anders MA, Kaul H, Agarwal A, Hsu SK, Chen G, et al. μ RNG: a 300–950 mV, 323 Gbps/W All-Digital full-entropy true random number generator in 14 nm FinFET CMOS. IEEE J Solid-State Circ. 2016;51(7):1695–704.
8. Bucci M and Luzzi R. Design of testable random bit generators. In International Workshop on Cryptographic Hardware and Embedded Systems, pp. 147–156. Springer, 2005.
9. Sunar B, Martin WJ, Stinson DR. A provably secure true random number generator with built-in tolerance to active attacks. IEEE Trans Comput. 2007;56(1):109–19.
10. Wold K and Petrović S. Behavioral model of TRNG based on oscillator rings implemented in FPGA. In 14th IEEE international symposium on design and diagnostics of electronic circuits and systems, pages 163–6. IEEE, 2011.
11. Haddad P, Fischer V, Bernard F, and Nicolai J. A physical approach for stochastic modeling of TERO-based TRNG. In: International workshop on cryptographic hardware and embedded systems, pp. 357–72. Springer, 2015.
12. Callegari S, Rovatti R, Setti G. Embeddable ADC-based true random number generator for cryptographic applications exploiting nonlinear signal processing and chaos. IEEE Trans Signal Process. 2005;53(2):793–805.
13. Cherkaoui A, Fischer V, Fesquet L, and Aubert A. A very high speed true random number generator with entropy assessment. In: International Workshop on Cryptographic Hardware and Embedded Systems, pp. 179–96. Springer, 2013.
14. Fischer V and Drutarovskỳ M. True random number generator embedded in reconfigurable hardware. In: International Workshop on Cryptographic Hardware and Embedded Systems, pp. 415–30. Springer, 2002.
15. Yang B, Rožic V, Grujic M, Mentens N, and Verbauwhede I. ES-TRNG: a high-throughput, low-area true random number generator based on edge sampling. IACR Transactions on Cryptographic Hardware and Embedded Systems, pp. 267–92, 2018.
16. Cassiers G, Grégoire B, Levi I, Standaert F-X. Hardware private circuits: from trivial composition to full verification. IEEE Trans Comput. 2020;70(10):1677–90.
17. Salomon D and Levi I. On the performance gap of a generic C optimized assembler and wide vector extensions for masked software with an ascon-{\it {p} test case. Cryptology ePrint Archive, 2022.
18. Levi I, Bellizia D, Bol D, Standaert F-X. Ask less, get more: side-channel signal hiding, revisited. IEEE Trans Circ Syst I Regul Pap. 2020;67(12):4904–17.
19. Levi I, Bellizia D, Standaert F-X. Beyond algorithmic noise or how to shuffle parallel implementations? Int J Circ Theory Appl. 2020;48(5):674–95.
20. Bilgin B, De Meyer L, Duval S, Levi I, Standaert F-X. Low AND depth and efficient inverses: a guide on s-boxes for low-latency masking. IACR Trans Symmetric Cryptol. 2020;2020(1):144–84.
21. Turan MS, Barker E, Kelsey J, McKay KA, Baish ML, Boyle M. Recommendation for the entropy sources used for random bit generation. NIST Special Public. 2018;800:90B.
22. Klein N, Harel E, Levi I. The cost of a true random bit-on the electronic cost gain of ASIC time-domain-based TRNGs. Cryptography. 2021;5(3):25.
23. Xilinx. Constraints guide, UG625 (v. 14.5). 1 April 2013.
24. Petura O, Mureddu U, Bochard N, Fischer V, and Bossuet L. A survey of AIS-20/31 compliant TRNG cores suitable for FPGA devices. In 2016 26th international conference on field programmable logic and applications (FPL), pp. 1–10. IEEE, 2016.
25. Yang B, Rožić V, Mentens N, Dehaene W, and Verbauwhede I. TOTAL: TRNG on-the-fly testing for attack detection using lightweight hardware. In: Proceedings of the 2016 Conference on Design, Automation & Test in Europe, pp. 127–32. EDA Consortium, 2016.
26. Grujić M, Rožić V, Yang B, and Verbauwhede I. A closer look at the delay-chain based TRNG. In 2018 IEEE International Symposium on Circuits and Systems (ISCAS), pp. 1–5. IEEE, 2018.
27. Yang B, Rožić V, Grujić M, Mentens N, and Verbauwhede I. On-chip jitter measurement for true random number generators. In: 2017 Asian hardware oriented security and trust symposium (AsianHOST), pp. 91–96. IEEE, 2017.
28. Guntur H, Ishii J, and Satoh A. Side-channel attack user reference architecture board SAKURA-G. In: 2014 IEEE 3rd global conference on consumer electronics (GCCE), pp. 271–4. IEEE, 2014.
29. Dichtl M. Bad and Good ways of post-processing biased random numbers.
30. Baudet M, Lubicz D, Micolod J, Tassiaux A. On the security of oscillator-based random number generators. J Cryptol. 2011;24(2):398–425.
31. Kohlbrenner P and Gaj K. An embedded true random number generator for FPGAs. In: Proceedings of the 2004 ACM/SIGDA 12th international symposium on Field programmable gate arrays, pp. 71–8. ACM, 2004.
32. Bernard F, Fischer V, Valtchanov B. Mathematical model of physical RNGs based on coherent sampling. Tatra Mt Math Publ. 2010;45(1):1–14.
33. Varchola M and Drutarovsky M. New high entropy element for FPGA based true random number generators. In: International workshop on cryptographic hardware and embedded systems, pp. 351–65. Springer, 2010.
34. Yang K, Blaauw D, and Sylvester D. A robust -40 to 120°C all-digital true random number generator in 40 nm CMOS. In: 2015 Symposium on VLSI Circuits (VLSI Circuits), pp. C248–9. IEEE, 2015.
35. Yang K, Fick D, Henry MB, Lee Y, Blaauw D, and Sylvester D. 16.3 A 23Mb/s 23pJ/b fully synthesized true-random-number generator in 28 nm and 65 nm CMOS. In: 2014 IEEE international solid-state circuits conference digest of technical papers (ISSCC), pp. 280–1. IEEE, 2014.
36. Kim E, Lee M, and Kim J-J. 8.2 8Mb/s 28Mb/mJ robust true-random-number generator 65nm CMOS based on differential ring oscillator with feedback resistors. In: 2017 IEEE international solid-state circuits conference (ISSCC), pp. 144–5. IEEE, 2017.