Enhancing Quality and Security of the PLL-TRNG

Viktor Fischer, Florent Bernard, Nathalie Bochard, Quentin Dallison, Maciej Skórski

IACR Transactions on Cryptographic Hardware and Embedded Systems
ISSN XXXX-XXXX, Vol. 0, No. 0, pp. 1–27. DOI:XXXXXXXX

Enhancing Quality and Security of the PLL-TRNG

Viktor Fischer[1,3], Florent Bernard[1], Nathalie Bochard[1], Quentin Dallison[2] and Maciej Skórski[4]

[1] Hubert Curien Laboratory, Jean Monnet University, Saint-Etienne, France, (florent.bernard,nathalie.bochard,fischer)@univ-st-etienne.fr
[2] THALES, Gennevilliers, France, quentin.dallison@thalesgroup.com
[3] FIT, Czech Univ. of Technology, Prague, Czech republic, viktor.fischer@fit.cvut.cz
[4] University of Warsaw, Warsaw, Poland, maciej.skorski@gmail.com

Abstract. Field Programmable Gate Arrays (FPGAs) are used more and more frequently to implement cryptographic systems, which need random number generators (RNGs) to be embedded in the same device. The main challenge related to the implementation of a generator running inside FPGAs is that the physical source of randomness, such as jittered clock generator, is implemented in the configurable logic area, i.e. in the close vicinity of noisy running algorithms, which can have significant impact on generated numbers or even serve to attack the generator. A possible approach to prevent such influence is the use of Phase-Lock Loops (PLLs), which are separated from the re-configurable logic area inside the FPGA chip. In this paper, we propose a new architecture of the PLL-based TRNG including a method to avoid correlation in the output through control of timing in the sampling process, as well as new embedded tests based on the enhanced stochastic model. We also propose a workflow to help find the best parameters, such as output bitrate and entropy rate. We show that bitrates of around 400 kb/s or more can be achieved, while guaranteeing min-entropy rates per bit higher than 0.98 as required by the latest security standards.

Keywords: Random number generation · Parameterized stochastic models · Dedicated statistical tests · Randomness monitoring

1 Introduction

Random number generators are essential in cryptography and data security as they guarantee unpredictability of generated numbers, used as confidential keys, nonces (numbers used once), padding data or as masks in side-channel attacks countermeasures. To avoid possible manipulations, random numbers should never be generated outside the protected cryptographic system, which is usually embedded inside an integrated logic device. However, logic devices are developed to have predictable behavior to implement algorithmic systems. Finding a robust physical source of randomness related to analog phenomena inside logic devices remains a challenge.

The list of physical sources of randomness that can be exploited in digital devices is quite limited. The most commonly used source is the jitter of clock signals generated in ring oscillators [BLMT11, YRG+18], self-timed rings [CFFA13] or PLLs [FD03]. The frequency spectrum of this jitter is often complex and composed of random components coming from the thermal noise and low-frequency noises [HTBF14], but also from pseudo-random noises or even data dependent noises [VABF10]. The designer must therefore analyze and

quantify the jitter, as well as understand the possible manipulations with noise sources to have a way of detecting them systematically or even creating a resilient design.

The unpredictability of generated numbers as one of the most important parameters of TRNGs is guaranteed by a sufficient entropy rate at the generator output. Since the entropy is a property of random variables and not of their realizations, it cannot be measured (e.g. by observing the generated numbers). It can only be estimated using a stochastic model, which has become a strict requirement by the AIS 31 standard [PV22]. The stochastic model consists of a family of probability distributions that describe the theoretical behavior of the raw random signal allowing the verification of a (lower) entropy bound for the output data (see [PV22], paragraph 45).

Ideally, the model should be based on a set of measurable parameters, e.g. the clock jitter in the case of the ring oscillator based TRNGs [BLMT11] or the length of a randomly lasting time intervals as it is the case in the TRNG using noise from two Zener diodes [KS08]. The measurement of input parameters of the model can then constitute a basis for the dedicated tests of randomness, which can be used to verify the entropy rate in real time. For example, jitter measurement methods that can be embedded in logic devices are presented in [FL14, MLC+14].

Many papers proposing TRNG designs implementable in logic devices exist, but not all TRNGs are suitable for implementation in both FPGAs and ASICs. Publications [PMB+16, GGFZ22] contain surveys of TRNG implementable in FPGAs, and a recent example of a generator dedicated to implementation in ASIC can be found in [PV22].

The challenge of TRNG implementation in logic devices is well illustrated by the availability of the TRNG IP cores respecting stringent security criteria. Indeed, four IP cores available currently on the market claim to be compliant only with NIST SP 800-90B [DR23, IPC23, Lat23, Ram23], two other IP cores also claim to pass AIS31 tests [Sil23, Ber23] and only three more claim to be fully compliant with NIST SP 800-90B and AIS31 [IC23, Arm23, Syn23]. However, even for these last three IP cores, the availability of a stochastic model and of efficient dedicated tests based on the model is not clearly stated.

FPGA devices and especially reconfigurable systems on chip (SoC) are used very frequently (and will be used even more in the future) as cryptographic systems on chip [ZGS+21], in which random number generators (RNGs) constitute an essential and unavoidable part. For security reasons, RNGs cannot be implemented outside the cryptographic system; we therefore have to implement them inside the FPGA (when we use FPGA as an implementation technology). Currently, PLLs are located in the only part of the FPGA device which has independent power supply and which is outside the configurable logic area. These elements are highly beneficial as they reduce to a minimum the risk of crosstalk and thus possibilities of active attacks.

Surveys presented in [PMB+16, GGFZ22] show the main advantages of the TRNG using PLLs (PLL-TRNG) compared to other TRNG designs suitable for FPGAs: a source of randomness (PLLs) powered independently from the rest of the device, simple design, high entropy rates, availability of the stochastic model and dedicated tests. The two main weaknesses are a relatively low speed and possible correlations between output bits.
In this paper:

- we propose to use an m-bit time-to-digital converter (TDC) instead of a 1-bit decimator used in the original PLL-TRNG, obtaining more information regarding the source of randomness;
- we introduce new dedicated tests taking advantage of the information offered by the m-bit TDC for an overall gain in speed, robustness and precision;
- we study correlations between random sampled bits before the TDC, and identify the time distance between such bits as a determining factor for the entropy rate;
- we define a procedure to obtain suitable time distances for a given PLL setup and

include the notion of time distances between random bits in the parameterized stochastic model;
- we share all the necessary material (raw data as well as data processing scripts) to reproduce the results of this paper in the GitHub repository [SDB+23].

The paper is organized as follows: after the Introduction, we first present the PLL-TRNG principle and the enhanced architecture in Section 2. In Section 3 we describe the stochastic model of the generator taking into account time distances between contributing samples (samples increasing the entropy rate). Section 4 is dedicated to the proposed dedicated statistical tests based on the parameterized PLL-TRNG statistical model. Sections 5 and 6 contain results of hardware implementation as well as statistical validation of the model assumptions. We finally discuss the obtained results in Section 7 before closing the paper in Section 8.

2 PLL-TRNG architecture and its evolution

In this section, we will first present and discuss the original PLL-TRNG architecture. We will then introduce the modified PLL-TRNG and the proposed dedicated embedded tests.

2.1 PLL-TRNG basic architecture

The PLL-TRNG, which uses a jittered clock signal generated in a phase-locked loop (PLL) as a source of randomness, was first published in [FD03]. The stochastic model of the generator and corresponding dedicated embedded tests were then introduced in [FBB19]. The original PLL-TRNG architecture is depicted in Fig. 1. The generator is based on a coherent sampling principle: the clock signal clk_1 is sampled in a D flip-flop (DFF) at rising edges of the reference clock signal clk_0. The mean frequencies $f_0 = 1/T_0$ and $f_1 = 1/T_1$ of both clock signals have some fixed ratio determined by the PLL block.

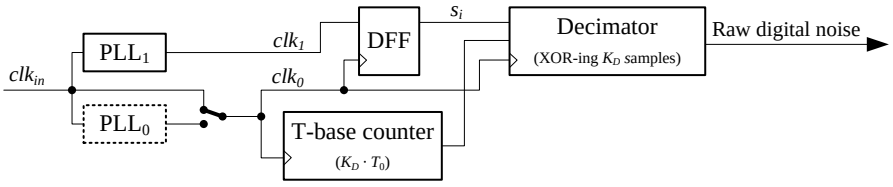


Figure 1: Architecture of the PLL-based TRNG (PLL-TRNG).

In the simpler version of the generator, only one PLL (PLL$_1$) is used and the PLL-generated clock signal has the frequency $f_1 = (K_{M1}/K_{D1}) \cdot f_{in} = (K_M/K_D) \cdot f_0$, where K_{M1} and K_{D1} are multiplication and division factors of the PLL$_1$ clock generator. If K_M and K_D are mutually prime and K_D is odd[1], the generator's output bitrate is equal to $R = f_0/K_D$ and the sensitivity to the jitter is $S = \Delta^{-1}$, where $\Delta = T_1/K_D$ is the time resolution in the reconstructed period T_1, as it is presented in Fig. 2.

If the clock signals clk_1 and clk_0 were jitter-free, the signal s_i (the sampler output) would feature a pattern with period $T_P = K_D \cdot T_0$. To remove the pseudo-random behavior of this pattern, a time-base signal is generated in a synchronous counter to restart the decimator after each period T_P. The decimator is thus XOR-ing K_D samples to output one bit.

In practice, both clk_1 and clk_0 are jittered, and we denote by σ_0 (resp. σ_1) the period jitter of clk_0 (resp. clk_1). These jitters are mainly due to local noises originating from the

[1]Note that choosing K_D odd increases sensitivity to the jitter by a factor of two as explained in [FD03, Sect. 3.3].

transistors inside each PLL and thus independent from one another. In this regard, we can apply the technique detailed in [BLMT11, Appendix C] to simplify further computations by transferring the jitter of clk_0 to clk_1. The period of clk_1 is then considered to be a random variable of the mean value T_1 and standard deviation:

$$\sigma_1' \simeq \sqrt{\sigma_1^2 + \frac{T_1}{T_0}\sigma_0^2}, \tag{1}$$

and the period T_0 of clk_0 is supposed to be constant.

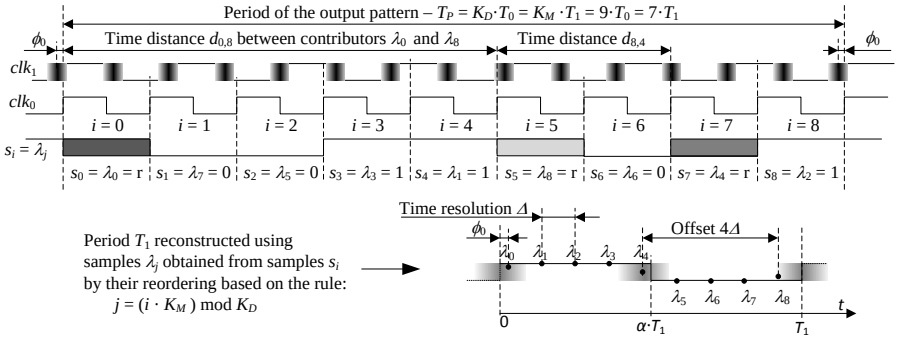


Figure 2: Reconstruction of one sampled clock period T_1 by reordering successive K_D samples s_i for $K_D = 7$ and $K_M = 6$, ϕ_0 is the initial relative phase between clk_1 and clk_0.

Thanks to the coherent sampling principle, the period of the sampled signal clk_1 can be reconstructed by reordering K_D original samples s_i (obtained at the DFF output) to a new set λ_j, where $s_i = \lambda_j$, $j = (i \cdot K_M) \bmod K_D$, as shown in Fig. 2. By increasing K_D, the designer can reduce the time resolution (Δ) in the reconstructed period T_1, increasing the sensitivity to the jitter and consequently the entropy rate.

A majority of samples s_i most likely have a constant value: about half of them are equal to zero (s_1, s_2, s_6 in Fig. 2) and about half are equal to one (s_3, s_4, s_8); depending on the initial phase ϕ_0, the time resolution Δ and the standard deviation σ_1', few samples (s_0, s_5, s_7) will have a random value. The decimator thus outputs one pattern-free random bit per pattern period T_P.

As usually, the output bitrate and entropy rate have contradictory requirements: increasing K_D will decrease the output bitrate and increase entropy rate and *vice versa*. However, since the sensitivity to the jitter is equal to $S = K_D \cdot f_1 = K_M \cdot f_0$, the entropy rate can also be increased by increasing the multiplication factor K_M and the reference clock frequency f_0.

If the multiplication and division factors of a unique PLL are not suitable to obtain sufficient output bitrate and/or entropy rate, the second PLL can be used (see Fig. 1). The new multiplication and division factors will be $K_M = K_{M1} \cdot K_{D0}$ and $K_D = K_{D1} \cdot K_{M0}$, respectively. Since the main difficulty in the design of the PLL-TRNG is the choice of suitable multiplication and division factors from the huge design space, the authors in [APFB18] proposed an algorithm, which helps in finding all feasible configurations. This algorithm was further enhanced in [CBBB20] by the use of possible optimisation criteria.

Based on this analysis, we can highlight the following advantages of the original PLL-TRNG:

1. a simple architecture and a design essentially related to the choice of f_{in}, K_M, K_D;
2. modern FPGA families always contain several PLLs, placed in a delimited area and thus independent from activities in the rest of the circuitry inside the same device;
3. the pseudo-random pattern at the output is erased by the decimator, which facilitates the detection of entropy failures.

2.2 Design rationale of the modified PLL-TRNG

The enhanced PLL-TRNG architecture is presented in Fig. 3.

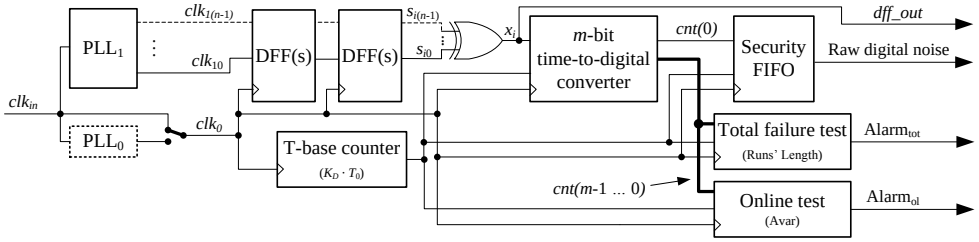


Figure 3: Architecture of the enhanced PLL-based TRNG including m-bit time-to-digital converter, security FIFO and new dedicated tests exploiting the m-bit converter output.

We introduced the following modifications aimed at increasing security and performance of the PLL-TRNG:

1. the output of the PLL1 is sampled by two consecutive D flip-flops (DFFs) connected in series, the second being used to resolve from possible metastable events;
2. the sampler output x_i is converted to a pattern-free digitized analog signal (the das signal mentioned in [KS11]) using an m-bit time-to-digital converter (TDC) – by counting the number of samples x_i equal to 1 during the pattern period $T_P = K_D \cdot T_0$, the TDC converts the pseudo-random pattern to an m-bit random number;
3. the Total failure test and Online test rely on the m-bit converter output;
4. the least significant converter bit $cnt(0)$ is used as the raw random signal;
5. the raw signal is temporarily saved in a FIFO memory, whose depth depends on the latency of the Total failure test;
6. to further increase the output entropy rate, we consider the sampled PLL (PLL1) having up to n outputs delayed by $180/n$ degrees – if $n = 1$, then $s_{i0} = s_i = x_i$.

Note that the T-base counter serves the same purpose as in the original PLL-TRNG.

3 Stochastic model

In this section, we will introduce the assumptions that the stochastic model of the modified PLL-TRNG relies on. Their validity will be verified and discussed later in Section 6.

3.1 Model assumptions

Assumption 1 (Stationarity of the pattern). *Within each pattern period T_P, the sampler (DFF) outputs a vector of K_D bits (bits x_i from Fig. 3). We assume that this vector is (time-locally) stationary.*

Given the state-space of the random variable (a vector of 20 random bits can have 2^{20} or about 1 000 000 different values), we are unable to perform a fully rigorous statistical analysis. However, we can still be confident in our claim of stationarity of the DFF output since the position in time of all the random bits of a given period T_P is constant, as illustrated in more detail in Section 6.1. In addition, we developed a local variant of the stationarity test which gives (moderate) support of the claim.

Assumption 2 (Output stationarity). *Raw random numbers at the TRNG output*[2] *can be interpreted as realizations of stationarily distributed binary-valued random variables.*

[2]We recall that we do not post-process the raw random numbers, so in our case raw random numbers correspond to internal random numbers.

This assumption is explored in detail and validated in Section 6.2. We also explore the stationarity of the full counter value, as it can be used to produce a generator output with higher bitrate.

Assumption 3 (Small intra-pattern correlations). *Although consecutive sampled bits are correlated, we will assume that this correlation decreases sharply over time and that bits that are far enough from each other can be considered independent.*

This claim is backed by observations made in Section 6.4. Moreover, since we can somewhat control the time distances between random bits (as explained in Section 3.3), this allows us to decrease any observable correlations down to acceptable levels.

Assumption 4 (Independence of least significant bits of the counter output). *We assume that the subsequent values of the counter least significant bit are uncorrelated.*

This claim is backed by observations made in Section 6.3. However, more work should be done to assess the independence of subsequent counter values.

3.2 Stochastic model of the modified PLL-TRNG

According to Fig. 3, the PLL block generates one reference clock (clk_0) and n sampled clocks (clk_{1k}, $k \in [\![0, n-1]\!]$). These clocks have the same mean frequency $f_1 = 1/T_1$ and differ only by a phase shift. Even if the formulas presented in this section can be extended for any number n of phase shifted clock signals, we will first study the case of only one clock signal ($n = 1$) $clk_1 := clk_{10}$ sampled by clk_0 and discuss in Section 7 the use of more phase shifted clock signals ($n \geq 2$). In this case, samples x_i in Fig. 3 correspond to samples s_i in Fig. 2 and are realizations of a random variable X_i that will be characterized in Prop. 2.

In general, the phase of a clock signal is influenced by different noise sources, but only the thermal noise can be considered to be independent of other noise sources and non-manipulable. This thermal noise is a white noise which can be modelled by a Gaussian distribution law. In the case of the PLL-TRNG, low frequency noise sources (such as $1/f^\beta$ noises, e.g. flicker noise) are filtered out by the control loop of the PLL [DDS18]. Furthermore, the jitter due to thermal noise does not accumulate and remains bounded.

The consequence is that the phases of both rising and falling edges of the sampled signal have the same standard deviation σ'_1 caused mainly by the bounded accumulation of the jitter coming from the thermal noise and denoted $\sigma'_1 \approx \sigma'_{1,th}$ according to Eq. (1).

According to this remark and following notations introduced in Fig. 2, we can state the following proposition.

Proposition 1 (Phases Z_j as random variables in the reconstructed period)**.** *For each $j \in [\![0, K_D - 1]\!]$, $Z_j \sim \mathcal{N}\left(\mu_j, {\sigma'_1}^2\right)$ where $\mu_j := \phi_0 + j \times \Delta \mod T_1$, $\Delta = \frac{T_1}{K_D}$ and $\mathcal{N}\left(\mu, \sigma^2\right)$ denotes the Normal law of mean μ and variance σ^2.*

This proposition gives a set of input parameters for the stochastic model. We can classify them into two categories:

1. Configurable parameters, chosen by the designer (helped by the stochastic model):
   - the sampling frequency f_0,
   - the PLL coefficients K_M and K_D.
2. Intrinsic parameters, which can be observed or measured but not altered:
   - the initial phase ϕ_0 of the sampled clock,
   - the duty cycle α of the sampled clock,
   - the bounded accumulated jitter σ'_1.

In our case study, according to Fig. 2, the clk_1 signal with period T_1 and duty cycle α starts in state 1, with a phase shift ϕ_0 from clk_0. Given the knowledge of the distribution of each phase Z_j ($j \in [\![0, K_D - 1]\!]$) from Prop. 1, it is possible to compute the probability that the $(j+1)^{th}$ bit X_j sampled on the clk_1 signal is equal to 1.

Although we have $\mu_j \in [0, T_1[$, because of the jitter $\sigma'_1 > 0$, realizations of Z_j can lie outside of $[0, T_1[$. For example, for $\varepsilon \gtrsim 0$ and $\mu_j = T_1 - \varepsilon$, a realization of Z_j can be equal to $\mu_j + 2\varepsilon > T_1$. Furthermore, because $\sigma'_1 \ll T_1$, it is unlikely that $Z_j < -T_1 + \alpha \cdot T_1$ (previous falling edge of clk_1) or $Z_j > T_1 + \alpha \cdot T_1$ (next falling edge of clk_1). For this reason, we can assume that: $\Pr(-T_1 + \alpha \cdot T_1 < Z_j < T_1 + \alpha \cdot T_1) = 1$, and can state the following proposition:

Proposition 2 (Sampled bit X_j on the reconstructed period)**.** *For each $j \in [\![0, K_D - 1]\!]$,*

$$\begin{aligned}\Pr(X_j = 1) &= \Pr(0 < Z_j < \alpha \cdot T_1) \quad + \Pr(T_1 < Z_j < T_1 + \alpha \cdot T_1) \\ &= \Pr(0 < Z_j < \alpha \cdot T_1) \quad + \Pr(T_1 < Z_j) \\ &= \frac{1}{\sqrt{2\pi}\sigma'_1} \int_0^{\alpha \cdot T_1} e^{-\frac{(t-\mu_j)^2}{2{\sigma'_1}^2}} dt + 1 - \frac{1}{\sqrt{2\pi}\sigma'_1} \int_{-\infty}^{T_1} e^{-\frac{(t-\mu_j)^2}{2{\sigma'_1}^2}} dt,\end{aligned}$$

which can also be expressed with the cumulative density function Φ of the standard normal distribution:

$$\Pr(X_j = 1) = \Phi\left(\frac{\alpha \cdot T_1 - \mu_j}{\sigma'_1}\right) - \Phi\left(-\frac{\mu_j}{\sigma'_1}\right) + 1 - \Phi\left(\frac{T_1 - \mu_j}{\sigma'_1}\right).$$

Since counter values contain more information than their least significant bit, we will study the probability distribution of these values. This is one of the main benefits of replacing the decimator in the original TRNG architecture by the m-bit converter. Indeed, this study will be used to define and configure the Online and Total failure tests.

Definition 1 (Counter output N_p and *raw random numbers* R_p)**.** For each pattern period T_P, the counter output value of Fig. 3 is represented by the stochastic process $(N_p)_{p \in \mathbb{N}}$ defined below for each p:

$$N_p := \sum_{i=0}^{K_D - 1} X_i,$$

i.e. it is the sum of the random variables that are characterized by Prop. 2.
The *raw random numbers* are the random variables R_p corresponding to the least significant bit of N_p:

$$R_p := N_p \mod 2.$$

To study the probability distribution of N_p (and, consequently, that of R_p), one has to assume stationarity of these processes and particularly the stationarity of X_j among many periods T_P. This question is addressed in Section 6.1.

Once stationarity is confirmed, we want to ensure that random bits $(X_j)_{j \in [\![0, K_D - 1]\!]}$ are uncorrelated. This is not true in general for all these bits, but we can simplify the problem by considering what we call contributors:

Definition 2 (Contributors)**.** We call contributors, or contributing bits, the bits X_j such as $0 < \Pr(X_j = 1) < 1$. They correspond to sampling points on either edge of the sampled clock, and their value is random due to jitter.

In order to have a practical way to take into account these contributors, we can set more precise thresholds such as $0.02275 \leq \Pr(X_j = 1) \leq 0.97725$ corresponding to a phase Z_j falling in the interval $[\mu_j - 2\sigma'_1; \mu_j + 2\sigma'_1]$ of length $4\sigma'_1$. Moreover, using $\Pr(X_j = 1)$ from Prop. 2, for given σ'_1, ϕ_0, α, it is possible to compute the theoretical number of contributors for a given configuration.

According to these thresholds, we can then split the set of K_D indices in three subsets:
1. S_0 the set of indices j such that $\Pr(X_j = 1) \approx 0$ (< 0.02275) (X_j is almost always 0)
2. S_1 the set of indices j such that $\Pr(X_j = 1) \approx 1$ (> 0.97725) (X_j is almost always 1)
3. S_c the set of indices j for contributing bits

We can then rewrite the counter output as

$$N_p = \sum_{j \in S_c} X_j + \#S_1,$$

where $\#S_1$ denotes the number of elements in the set S_1, and

$$R_p = \bigoplus_{j=0}^{K_D-1} X_j = \left(\bigoplus_{j \in S_c} X_j \right) \oplus B,$$

where $B = 1$ if and only if $\#S_1$ is odd.

As we want to express the probability that $R_p = 1$, we can use the formula from [Dav02] to compute $\Pr(X \oplus Y = 1)$ in the case where X and Y are not fully independent. It is given in Eq. (2) where $\mu := \Pr(X = 1)$, $\nu := \Pr(Y = 1)$ and $\rho := \mathrm{Corr}(X, Y)$

$$\Pr(X \oplus Y = 1) = \frac{1}{2} - 2\left(\mu - \frac{1}{2}\right)\left(\nu - \frac{1}{2}\right) - \rho\sqrt{\mu(1-\mu)\nu(1-\nu)}. \tag{2}$$

The expression $\rho\sqrt{\mu(1-\mu)\nu(1-\nu)}$ is negligible if at least one of the two following conditions is fulfilled:

Condition 1: The correlation ρ between bits is very small.

Condition 2: The probability that a bit is equal to 1 is very close to 0 or very close to 1.

If we can guarantee at least one of these two conditions, the formula becomes:

$$\Pr(X \oplus Y = 1) = \frac{1}{2} - 2\left(\Pr(X = 1) - \frac{1}{2}\right)\left(\Pr(Y = 1) - \frac{1}{2}\right), \tag{3}$$

which corresponds to assuming the independence of X and Y.

So according to Condition 2, only correlations between bits X_j where $j \in S_c$ have to be checked. They will form groups of adjacent bits (one group per clock edge) in this reconstructed period, but not necessarily adjacent in time. The distance in time between two samples X_j where $j \in S_c$ should be long enough to reduce correlations between sampled bits in time and satisfy Condition 1. We will present the methodology to find such configurations based on this distance in Section 3.3 and show in Table 2 that such configurations exist for all devices studied in this work.

Thus, it is possible to compute the probability that R_p is equal to 1.

Proposition 3 (Probability that R_p is equal to 1.)**.**
If the distance between samples $(X_j)_{j \in S_c}$ is high enough to reduce correlations between contributors, then

$$\Pr(R_p = 1) = \Pr\left(\bigoplus_{j=0}^{K_D-1} X_j = 1 \right) = \frac{1}{2} + (-2)^{K_D-1} \prod_{j=0}^{K_D-1} \left(\Pr(X_j = 1) - \frac{1}{2} \right).$$

Finally, according to Assumption 4, both Shannon entropy and min-entropy can be computed from the probability of R_p to be equal to 1.

Proposition 4 (Entropy rate of the PLL-TRNG)**.**
In the case of a PLL-TRNG, the Shannon entropy rate H_1 and the min-entropy rate H_∞ follow:

$$H_1 := -\Pr(R_p = 1)\log_2(\Pr(R_p = 1)) - (1 - \Pr(R_p = 1))\log_2(1 - \Pr(R_p = 1)),$$
$$H_\infty := -\log_2(\max(\Pr(R_p = 1), 1 - \Pr(R_p = 1))).$$

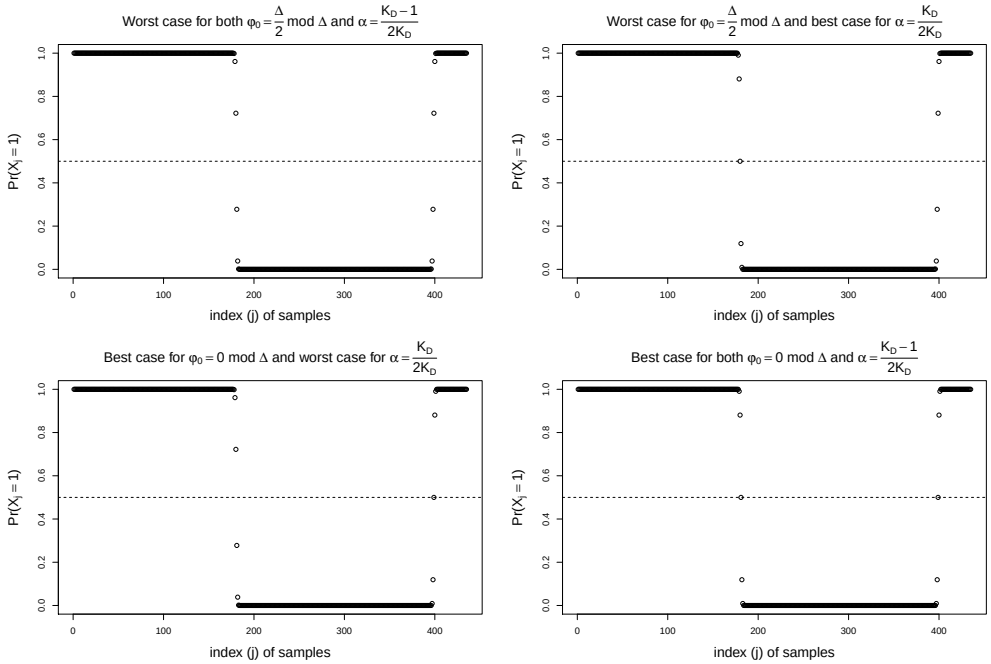


Figure 4: Position of contributors in the reconstructed period T_1 depending on ϕ_0 and α.

For a given configuration (f_0, K_M, K_D are fixed), the formulas above depend on three intrinsic parameters: the initial phase ϕ_0, the duty cycle α and the jitter σ'_1.
Some of them are difficult to measure precisely in the device (e.g. ϕ_0 and α). Since we want to compute a lower bound $\underline{H}_1$ (resp. $\underline{H}_\infty$) for the Shannon (resp. min-) entropy rate of the generator, a conservative approach is to set ϕ_0 and α in the case where the contributors have the lowest contribution to entropy. This worst case is achieved when the position of contributors along both the rising and falling edges (i.e. the probability $\Pr(X_j = 1)$) are the farthest away from the value 0.5.

The problem is illustrated in Fig. 4, where the choice of ϕ_0 determines the position of the rising edge, and the choice of α controls the falling edge through the length of the high state. Setting the initial phase $\phi_0 = \frac{\Delta}{2} \mod \Delta$ ensures a worst case scenario by positioning the contributors in the rising edge as far away from the 0.5 threshold as possible. Similarly, setting the duty cycle $\alpha = \frac{K_D - 1}{2K_D} \mod \frac{1}{K_D}$ ensures that the situation on the falling edge is analog to that on the rising edge. This overall worst case is pictured in the top-left panel, where the contributors are as far away from 0.5 as possible.

As it is illustrated in Fig. 5, it means that $\underline{H}_1$ (resp. $\underline{H}_\infty$) is a strictly increasing function of the jitter σ'_1 and for all ϕ_0 and α, $H_1(\sigma'_1) \geq \underline{H}_1(\sigma'_1)$ (resp. $H_\infty(\sigma'_1) \geq \underline{H}_\infty(\sigma'_1)$). Thanks to this lower bound, it is possible to set a threshold on the expected entropy rate of the generator and to compute the corresponding minimum required jitter value.

Figure 6 displays the importance of the minimal jitter obtained in that way. The curves show the min-entropy given by the model, either as a function of ϕ_0 or α, in two situations: the left panels consider a jitter $\sigma'_1 < \sigma_{min}$, while the right panels consider a sufficient jitter $\sigma'_1 > \sigma_{min}$. We can note that all the curves show the periodicity of the entropy mentioned earlier. The dotted curves correspond to an arbitrary choice of the fixed intrinsic parameter ϕ_0 or α. They mostly satisfy the requested entropy level, but not with certainty under $\sigma'_1 < \sigma_{min}$ for unlucky values of ϕ_0 or α. However, when these parameters are set to be in the worst situation described above (solid curves), we are at a minimum of the entropy; only by ensuring that $\sigma'_1 > \sigma_{min}$ we can state that the entropy

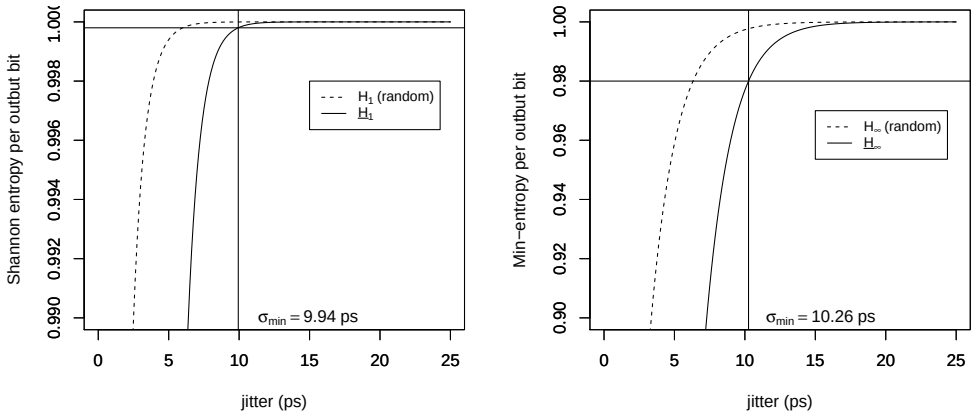


Figure 5: Entropy (left: Shannon, right: min-) per output bit as a function of the jitter. The dotted curves are for a random ϕ_0 and α whereas the solid one corresponds to the worst case. These curves are generated for Configuration A in Table 2. In this case, a minimum value of 9.94 ps (resp. 10.26 ps) for the jitter is computed in order to get 0.9998 Shannon entropy rate (resp. 0.98 min-entropy rate) required by [PS22].

level is always satisfactory.

This minimum value of the jitter will also be used to define, in Section 4, embedded tests based on the expectation and variance of counter values. Thanks to Assumptions 2 and 4, for each p, the counter value N_p can be modelled as a Poisson binomial distribution and its variance can be computed according to the distribution $p_i := \Pr(X_i = 1)$ given in Prop. 2.

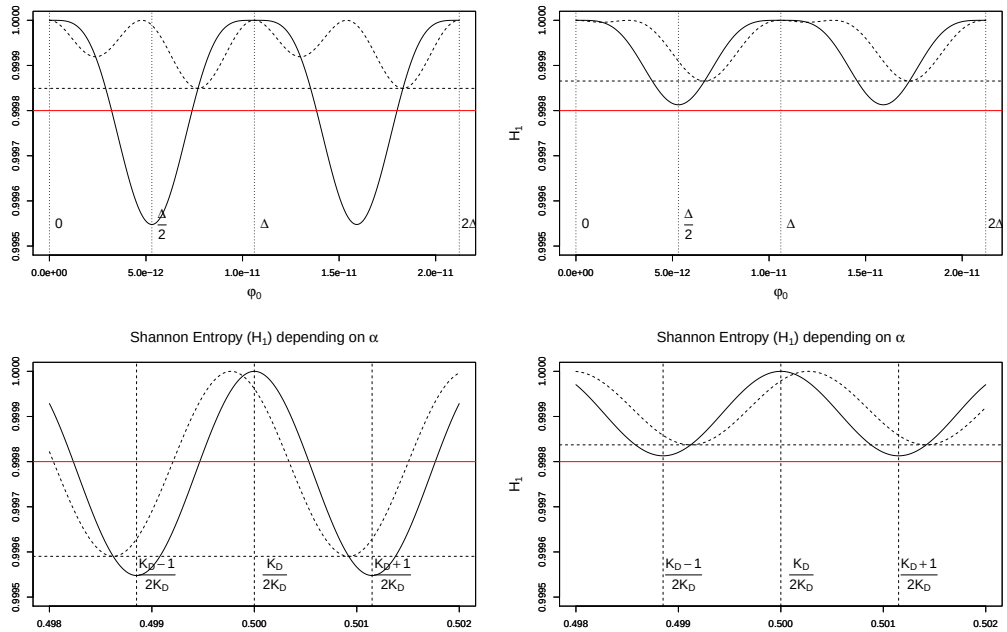


Figure 6: Shannon entropy per output bit depending on ϕ_0 (top) and α (bottom). Left panels are based on $\sigma_1' < \sigma_{min}$, right panels are based on $\sigma_1' > \sigma_{min}$. Solid curves correspond to the worst value of the fixed parameter, dotted curves correspond to an arbitrary value.

Proposition 5 (First moments of the counter output)**.**

$$\mathbb{E}(N_p) = \sum_{i=0}^{K_D-1} p_i = \sum_{i \in S_c} p_i + \#S_1,$$

$$\mathbb{V}\mathrm{ar}(N_p) = \sum_{i=0}^{K_D-1} p_i(1-p_i) = \sum_{i \in S_c} p_i(1-p_i).$$

Due to the possible presence of low frequency noises (e.g. flicker noise), it is generally preferable to use the Allan variance $\mathrm{AVAR}(N_p)$, which has the property of converging despite the presence of low frequency noises. Moreover, it is less expensive to use the Allan variance in terms of required hardware resources [APFB18].

Definition 3 (Allan variance of the counter output)**.** The Allan variance of the counter output is defined as follows:

$$\mathrm{AVAR}(N_p) := \frac{1}{2}\mathbb{E}\left((N_{p+1} - N_p)^2\right), \tag{4}$$

where $(N_p)_{p \in \mathbb{N}}$ are realizations of the counter value.

In practice, we observed that the Allan variance and classical variance are very close to each other – more details in the data log files in [SDB+23]. It goes to show that the PLL naturally filters out low frequency noises through its feedback loop. Due to this correspondence, we are able to keep the less expensive implementation of Allan variance while the model still uses classical variance.

3.3 Computation of the distance in time between contributors

Our goal in this section is to ensure minimal correlation between contributors by somewhat controlling the time distance between them. Indeed, the longer the time between two contributors, the lower the probability of one influencing the other. We recall that i denotes the index of samples in the pattern period T_P and j the index of samples in the reconstructed period T_1 obtained through coherent sampling. They are related through the following equation:

$$\forall i \in [\![0, K_D - 1]\!], \quad j = i \times K_M \mod K_D.$$

Because K_M and K_D are coprime numbers, we can consider the reciprocal function:

$$\forall j \in [\![0, K_D - 1]\!], \quad i(j) = (j \times K_M^{-1}) \mod K_D.$$

Given two samples λ_j and $\lambda_{j+\tau}$ that have an offset of $\tau > 0$ positions in the reconstructed period T_1, we are interested in their distance in time, i.e. the non-negative quantity $|i(j+\tau) - i(j)|$.

$$|i(j+\tau) - i(j)| = \begin{cases} (\tau \times K_M^{-1}) \mod K_D & \text{if } i(j+\tau) - i(j) > 0, \\ K_D - (\tau \times K_M^{-1} \mod K_D) & \text{if } i(j+\tau) - i(j) < 0. \end{cases}$$

This idea can then be generalized for a given offset τ and an arbitrary pair of samples with the following definition.

Definition 4 (Minimal distance in time between samples with a given offset τ)**.** Let $\tau \in [\![1, K_D - 1]\!]$. We then define:

$$d_{min}(\tau) := \min((\tau \times K_M^{-1}) \mod K_D, K_D - (\tau \times K_M^{-1} \mod K_D)).$$

Let us consider all pairs of samples $(\lambda_j, \lambda_{j+\tau})$ in the reconstructed period T_1: $d_{min}(\tau)$ is the minimal distance in time (expressed as a number of clk_0 periods) between any of those pairs. In other words, all pairs of samples with an offset τ will have a time distance of *at least* $d_{min}(\tau)$ in the pattern period T_P.

In order to control the distance in time between contributors, we would need the list of all offsets between contributors. Since this knowledge is not accessible, we want to find a meaningful set of offsets $\mathcal{T} := \{\tau_1, \ldots, \tau_q\}$ to compute $\{d_{min}(\tau)\}_{\tau\in\mathcal{T}}$. This list of time distances will be a great tool to examine possible correlations between contributors.
We consider two kinds of offsets between those contributing samples:

1. Between consecutive samples on one edge (rising or falling), such as samples λ_0 and λ_8 in Fig. 2. Let τ_1 be one less than the number of samples influenced by the jitter in one edge; we want to consider all values $\tau \in [\![1, \tau_1]\!]$ in order to account for all offsets between samples within a given edge. For example, in Fig. 2, $\tau_1 = 1$.
2. Between samples on different edges (rising and falling), such as samples λ_0 and λ_4 in Fig. 2. Given two samples λ_j and $\lambda_{j'}$ on the two edges, the offset τ_2 is either $\alpha \cdot T_1$ or $(1-\alpha)\cdot T_1$ in the reconstructed period T_1. It means that:

$$\begin{aligned} \lambda_j - \lambda_{j'} \equiv \alpha \cdot T_1 \mod T_1 &\Leftrightarrow j\cdot\Delta - j'\cdot\Delta \equiv \alpha\cdot T_1 \mod T_1 \\ \Leftrightarrow \quad j - j' \equiv \alpha \cdot \frac{T_1}{\Delta} \mod \frac{T_1}{\Delta} &\Leftrightarrow \qquad j - j' \equiv \alpha \cdot K_D \mod K_D. \end{aligned}$$

   Similarly, if $\lambda_j - \lambda_{j'} \equiv (1-\alpha)\cdot T_1 \mod T_1$, then $j - j' \equiv (1-\alpha)\cdot K_D \mod K_D$. Because the duty cycle α is in general difficult to know precisely, we consider that for a given configuration, α can be in $[\alpha_{min}, \alpha_{max}]$, then we have to consider offsets between :

$$\begin{aligned} \tau_2^{min} &= \lfloor \min(\alpha_{min}\cdot K_D, (1-\alpha_{max})\cdot K_D) \rfloor, \\ \tau_2^{max} &= \lceil \max(\alpha_{max}\cdot K_D, (1-\alpha_{min})\cdot K_D) \rceil. \end{aligned}$$

In the end, the set $\mathcal{T}$ of possible offsets between contributors to consider in the reconstructed period T_1 is:

$$\mathcal{T} = [\![1, \tau_1]\!] \bigcup [\![\tau_2^{min}, \tau_2^{max}]\!].$$

Example 1 (Determination of time distances for a given configuration)**.**
Let $K_D = 435$, $K_M = 728$ (corresponding to Configuration A in Table 2) and let $\tau_1 = 3$, $\alpha_{min} = 0.47$ and $\alpha_{max} = 0.52$, then the set $\mathcal{T}$ will be given by:

$$\mathcal{T} = [\![1, 3]\!] \bigcup [\![204, 231]\!].$$

Using Definition 4, we obtain the following list of distances: {193, 49, 144, 213, 20, 173, 69, 124, 118, 75, 167, 26, 216, 23, 170, 72, 121, 121, 72, 170, 23, 216, 26, 167, 75, 118, 124, 69, 173, 20, 213}.

A first approach could be to filter out configurations where a pair of contributors is too close – essentially setting a "minimal distance" constraint. A configuration with a high enough minimal distance would ensure an absence of correlation between any contributors. However, as we explored this possibility, it appeared to be too restrictive; it filters out many acceptable configurations and we sometimes end up with no option left. Typically, a configuration with eight well distributed contributors and two that are too close would be filtered out, but is exploitable at the cost of ignoring the one correlated pair.

This observation motivated our current approach of analyzing the whole set of distances between contributors for a given configuration. As stated in Definition 2, we estimate that we need at least eight uncorrelated contributors in the worst case to meet the entropy

requirements. Moreover, the correlation analysis made in Section 6.4 shows that a time distance of roughly $30 \cdot T_0$ is enough to neglect correlations.

Building on the precedent remarks, we propose the following analysis: for every distance that would be lower than the given threshold of 30, we consider the corresponding pair as one contributor, and make the conservative assumption that the second one does not contribute to entropy at all. We then check if we still have the required eight contributors.

We upgraded the code published in [CBBB20] to compute the set of distances between contributors when exploring possible configurations. This workflow is explained in Section 5.1 in more detail.

4 Dedicated embedded tests

In this section, we will present new embedded tests dedicated to the PLL-TRNG: the Total failure test and the Online test.

4.1 Total failure test

The Total failure test observes the length of runs of identical subsequent counter values N_p at the output of the time-to-digital converter. In the event of a total failure of the source of randomness, there are no contributors in T_P so the counter values remain constant. However there is always a small probability that some successive counter values are equal even in the presence of jitter. The stochastic model can be used to define more precisely how many times it can occur in normal operating condition within given period of time (a day, a week, a month...).

Equation 5 describes the probability that the counter value remains constant over $l \geq 2$ consecutive periods T_P.

$$\Pr(N_1 = \ldots = N_l) = \sum_{k=1}^{K_D} \left(\sum_{\substack{\mathcal{I} \in [\![0, K_D - 1]\!] \\ \mathcal{I} = \{i_1 < \cdots < i_k\}}} \left(\prod_{j=1}^{k} \Pr\left(X_{i_j} = 1\right) \cdot \prod_{j \in [\![0, K_D - 1]\!] \setminus \mathcal{I}} \left(1 - \Pr\left(X_j = 1\right)\right) \right) \right)^l . \quad (5)$$

Unfortunately, this formula is very hard to compute in practice, mainly due to the number of combinations of indices chosen among $[\![0, K_D - 1]\!]$.

We will use an approximation of this Poisson binomial distribution by a normal law. Indeed, for any integer $k \geq 1$, $\Pr(N_p = k) = F_{N_p}(k) - F_{N_p}(k-1)$ where F_{N_p} is the Cumulative Density Function (CDF) of the Poisson binomial distribution.
According to [Hon13], $F_{N_p}(k) \approx \Phi\left(\frac{k+0.5-\mathbb{E}(N_p)}{\sqrt{\mathbb{V}\mathrm{ar}(N_p)}}\right)$, so $\Pr(N_p = k) \approx \Phi\left(\frac{k+0.5-\mathbb{E}(N_p)}{\sqrt{\mathbb{V}\mathrm{ar}(N_p)}}\right) - \Phi\left(\frac{k-0.5-\mathbb{E}(N_p)}{\sqrt{\mathbb{V}\mathrm{ar}(N_p)}}\right)$. Thus, $\Pr(N_1 = \cdots = N_l)$ can be approximately computed by

$$\Pr(N_1 = \cdots = N_l) \approx \sum_{k=1}^{K_D} \left(\Phi\left(\frac{k + 0.5 - \mathbb{E}(N_p)}{\sqrt{\mathbb{V}\mathrm{ar}(N_p)}}\right) - \Phi\left(\frac{k - 0.5 - \mathbb{E}(N_p)}{\sqrt{\mathbb{V}\mathrm{ar}(N_p)}}\right) \right)^l .$$

Then, for any $\beta \in [0, 1]$, it is easy to compute the minimum integer l such that $\Pr(N_1 = \cdots = N_l) \leq \beta$. We denote $l_{min}(\beta)$ this value that will be used as a threshold for the Total failure test when too many subsequent counter values are identical. The pseudo-code below returns such $l_{min}(\beta)$:

Algorithm 1: Computation of $l_{min}(\beta)$.

$\ell \leftarrow 1$;
while $\sum_{k=1}^{K_D} \left(\Phi\left(\frac{k+0.5-\mathbb{E}(N_p)}{\sqrt{\mathbb{V}\mathrm{ar}(N_p)}} \right) - \Phi\left(\frac{k-0.5-\mathbb{E}(N_p)}{\sqrt{\mathbb{V}\mathrm{ar}(N_p)}} \right) \right)^{\ell} > \beta$ **do**
  $\ell \leftarrow \ell + 1$;
end
return ℓ

The specific value of β will be related to the probability of false alarm we can accept with this test during an operating time of t sec during which the generator produces $\frac{t}{K_D T_0}$ counter values (i.e. the probability that an ideal generator produces indeed l identical consecutive values during time t sec) and is computed as $\beta = \frac{K_D T_0}{t}$.
As an example for Configuration A, Table 1 presents the thresholds $l_{min}(\beta)$ for different false alarm scenarios and the latency ($l_{min}(\beta) K_D T_0$) of the test that is very small as it is expected for a Total failure test.

Table 1: Total failure test thresholds for configuration A ($K_M = 728$, $K_D = 435$, $T_0 = 4.61$ ns) for different false alarm scenarios and corresponding latency.

False alarm parameters	Once per day	Once per week	Once per month
β	$2^{-34.58}$	$2^{-37.38}$	$2^{-39.49}$
$l_{min}(\beta)$	24	26	28
Latency (as the number of periods T_0)	10 440	11 310	12 180
Latency (in μs)	80.643	87.363	94.083

To compare these results to the old design of the Total failure test, recall that it was based on 255 output bits; for this configuration with $K_D = 435$, it corresponds to the latency of 110 925 periods T_0. This new test design is therefore roughly ten times faster to raise an alarm in case of Total failure. Note that counting repetitions of the least significant bit of the time-to-digital converter (the equivalent of the decimator output bit in the original architecture) would also be possible, but this would significantly reduce both agility and robustness of the test. Note also that other additional sub-tests can be used to reinforce the Total failure test. Namely, the "locked" flag usually available as a PLL output can be used to verify that the PLL is indeed locked, since only locked PLLs ensure proper behavior of the PLL-TRNG.

4.2 Online test

The Online test is based on an estimation of the Allan variance according to Eq. 4, and its comparison with a predefined threshold. We compute the Allan variance based on 4096 counter values, as a compromise between precision and latency. If higher precision is needed, the number of counter values can be easily increased with small impact on the cost. Recall that we can compare the estimated Allan variance to a threshold on statistical variance, as the PLL naturally filters out low frequency noises.

The threshold value acts as a minimal required variance V_{min} derived from a minimal jitter σ_{min} (see Fig. 5) that is required to obtain at least the lower bound of entropy per bit (e.g. $H_\infty = 0.98$ according to [PS22]). Besides the jitter size (σ_{min}), the variance V_{min} also depends on the value of the initial clock phase ϕ_0 and the duty cycle α. To account for all possibilities, V_{min} is derived considering the best and worst possible case for the pair (ϕ_0, α) as it is illustrated in Fig. 7.

Note that, if the variance is below 0.5, it is not possible to have precise information about the jitter. Indeed, a configuration with a very small jitter but with a best case scenario for ϕ_0 and α will produce two contributors in the middle of both rising and falling

edge and give a variance of at least 0.5. So there is an absolute minimum threshold of 0.5 for $\mathbb{V}\mathtt{ar}(N_p)$ that can also be used as a complementary test to the Total Failure test.

The model can also be used to set an upper bound of the variance. Indeed in normal operating conditions, the jitter shouldn't be too high. As the variance is an increasing function of the jitter, this upper bound could be used to detect any attempt (e.g. in corner conditions) of manipulation of the source of randomness. Because the jitter will be configuration-dependent, it is not possible to set an absolute value for σ_{max} as it was done for the lower bound.

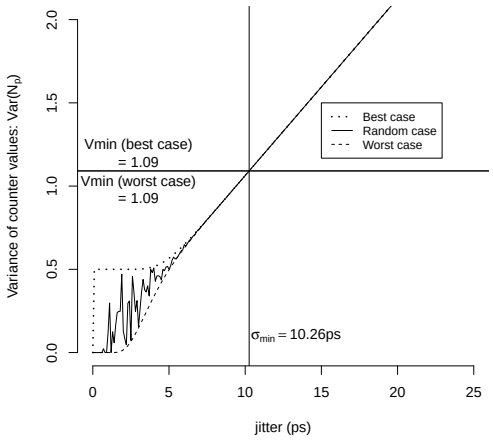


Figure 7: Variance of counter values as a function of the jitter for Configuration A, computed with Prop. 5 in both best (dotted curve above) and worst (dotted curve below) cases and for a random choice of ϕ_0 and α (solid curve). Note that for the smallest jitter value required in this configuration, the best case and worst case V_{min} values are identical.

5 Hardware implementations

5.1 PLL-TRNG setup

The PLL-TRNG design is restricted essentially to the choice of the design space parameters $\varsigma \in \{f_{in}, M_1, N_1, C_1, PVCO_1, f_1, n, M_0, N_0, C_0, PVCO_0, f_0\}$, where f_{in} is the input frequency, M_i, N_i, C_i, and $PVCO_i$, $i = \{0, 1\}$ are division factors of the PLL as presented in Fig. 8, f_1 and f_0 are output frequencies of PLL_1 and PLL_0, respectively and n is number of outputs of the PLL_1. Namely, the objective of the design is to select the parameter

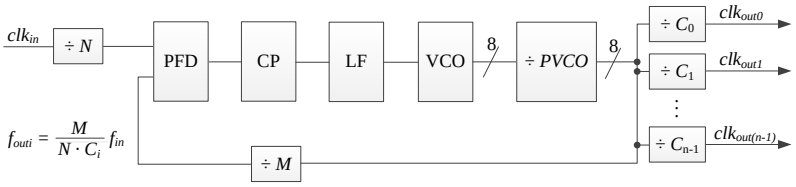


Figure 8: Typical phase-locked loop (PLL) architecture (PFD – Phase-frequency detector; CP – Charge pump; LF – Loop filter; VCO – Voltage-controlled oscillator; N, M, $PVCO$, C_i – input, loop, Post-VCO, and output frequency division factors, respectively).

values, which are within the intervals supported by the selected technology (e.g. [Int19]) and which will guarantee suitable bitrate R and sensitivity to jitter S (and thus entropy).

In our experiments, we used Intel Cyclone V FPGA device 5CEBA4F17C8N, Xilinx Spartan 6 FPGA device XC6SLX16-2FTG256C, and Microchip SmartFusion2 FPGA device M2S025-FG484, referred to in the following tables as CV, S6 and SF respectively. To find all possible PLL-TRNG configurations under the selected design constraints, we use the Python code presented in [CBBB20], which we upgraded as explained in Section 3.3.

In our experiments, we used the two-PLL TRNG configuration with the input frequency $f_{in} = 125$ MHz. While searching for possible configurations, we limited the reference clock frequency to $f_0 < 250$ MHz to guarantee timing constraints in counters and embedded tests. These requirements limited the design space for each device to hundreds of thousand possible configurations: more than 500 thousand configurations in Intel Cyclone V and Cyclone 10 FPGAs, about 100 thousand in Xilinx Spartan 6 and Spartan 7 FPGAs, and more than one million in Microchip SmartFusion2 FPGA. The exact values depending on input and output frequency constraints can be obtained using the Python code available at the GitHub repository [SDB+23]. We can observe in particular that some configurations propose very good performance but display just one low distance; this is the main motivation for our refined analysis of all time distances rather than just filtering by minimal value.

From the available configurations, we first selected the one which was identical for all three devices – Configuration A. Next, Configuration B is chosen to give a high sensitivity to the jitter, high distances between contributors, and a satisfying bitrate. Finally, Configuration C aims to show the impact of small time distances between contributors on correlations, while giving bitrates comparable with Configuration B. Table 2 presents selected configurations in the three available FPGA families. For all configurations, $f_{in} = 125$ MHz, K_M and K_D are the TRNG multiplication and division factors, R is the bitrate, S the jitter sensitivity.

Table 2: Selected PLL-TRNG configurations for the Intel Cyclone V FPGA, -C8 speed grade (configurations marked CV), Xilinx Spartan 6 FPGA, -2 speed grade (configurations S6), and Microchip SmartFusion2 FPGA, standard speed grade (configurations SF).

Config. name	M_0	N_0	C_0	P_0	f_0 [MHz]	M_1	N_1	C_1	P_1	f_1 [MHz]	K_M	K_D	R [Mb/s]	S [ps^{-1}]
CV_A S6_A SF_A	29	4	7	1	129.46	26	5	3	1	216.67	728	435	0.30	0.094
CV_B	99	13	4	1	237.98	8	1	5	1	200.00	416	495	0.48	0.099
S6_B	19	4	4	1	148.44	29	5	5	1	145.00	464	475	0.31	0.069
SF_B	31	4	4	1	242.19	23	3	3	1	319.44	368	279	0.87	0.089
CV_C	5	1	3	2	208.33	147	19	5	1	193.42	441	475	0.44	0.092
S6_C	33	4	7	1	147.32	17	5	3	1	141.67	476	495	0.30	0.070
SF_C	35	11	2	2	198.86	17	3	3	1	236.11	374	315	0.63	0.074

5.2 Implementation results

For a fair comparison, we realized three hardware modules, one for each family, with an identical architecture. The modules were powered from linear power supplies to avoid switching noises. Low noise quartz oscillators were used to generate the clock signal clk_{in}.

One of the timing-critical parts of the PLL-TRNG design from Fig. 3 are the two counters: the T-base counter and counter of samples equal to one representing the time-to-digital converter. Both count up to $K_D - 1$ at the frequency of clk_0. According to Table 2, K_D is always smaller than 511, so 9-bit counters are sufficient. These counters are fast enough even in the slowest family (SmartFusion2) to run at frequencies up to 250 MHz. The embedded tests receive m-bit input data in time intervals equal to $K_D \cdot T_0$, which are largely sufficient to perform more complex multiplication and addition operations. Another problem concerning placement and routing is the position of the first flip-flop of the sampler and the way the clock signal clk_1 arrives to its data input. Indeed, the PLLs

are aimed at generating clock signals (and not data) and are thus preferably connected to clock trees. We placed the first flip-flops as close as possible to the PLL1 output and used a regional clock tree for the Cyclone V family, and local routing resources for Spartan 6 and SmartFusion2.

We used the signal x_i from Fig. 3, which was output from the device via dedicated LVDS pins, to acquire 32 Mb of sampler output bits at high speed. The acquired files were used to reconstruct images of the period T_1 (see the right panels in Fig. 10 and 11 as two examples), to compute counter values and their mean values, Allan variance and the variance based on the probability distribution of contributors used in the model. Finally, from the same file, we generated the raw binary signal (TRNG output) and applied NIST SP 800-90B statistical test [SBK+18] to confirm that the generated values are IID and to estimate their min-entropy H_∞. All generated files are available in the repository [SDB+23].

Implementation results for the three PLL-TRNG configurations with one and two PLL1 outputs implemented in three available FPGA families are presented in Table 3. The mean counter values correspond approximately to $K_D/2$ confirming that the duty cycle is close to the expected value of 0.5. Note that counter values close to 0 or K_D would indicate possible loss of entropy (some contributors could be missing). We present only the Allan variance of counter values, since the Allan variance and traditional variance were always practically the same as can be seen in the log files in the GitHub repository. This observation confirmed the ability of the PLL control loop to largely mitigate the impact of the auto-correlated low frequency noises.

Table 3: PLL-TRNG implementation results for configurations A-C from Table 2 with one and two PLL1 outputs, implemented in selected devices: CV, S6 and SF.

		One PLL1 output					Two PLL1 outputs				
Config. name	R [Mb/s]	# contr.	Cnt mean	Cnt Avar.	Model var.	H_∞ (tests)	# contr.	Cnt mean	Cnt Avar.	Model var.	H_∞ (tests)
CV_A	0.30	11	218.3	1.78	1.79	0.999	23	235.6	2.38	3.62	0.994
S6_A	0.30	11	217.8	1.85	1.84	0.996	24	218.3	3.20	3.81	0.995
SF_A	0.30	20	218.2	2.17	3.24	0.995	40	214.6	2.96	6.45	0.996
CV_B	0.48	8	250.2	1.16	1.21	0.993	15	264.3	1.70	2.41	0.995
S6_B	0.31	9	237.3	1.76	1.56	0.996	18	236.7	2.32	2.97	0.996
SF_B	0.87	9	140.4	1.11	1.37	0.996	18	135.9	1.68	2.75	0.995
CV_C	0.44	8	239.9	1.70	1.41	0.913	19	252.9	12.2	3.02	0.988
S6_C	0.30	11	247.3	1.01	1.79	0.962	24	248.7	8.09	3.84	0.993
SF_C	0.63	20	157.9	0.55	3.09	nonIID	39	153.3	11.3	6.16	0.729

Last but not least, we can observe differences between variances computed from the counter values and the variance computed from the model, which is based on probabilities of contributors assuming they are independent. While all the variances should ideally be the same, these differences outline some noticeable correlation between the contributors. Specifically, we could expect stronger correlations in Configurations C.

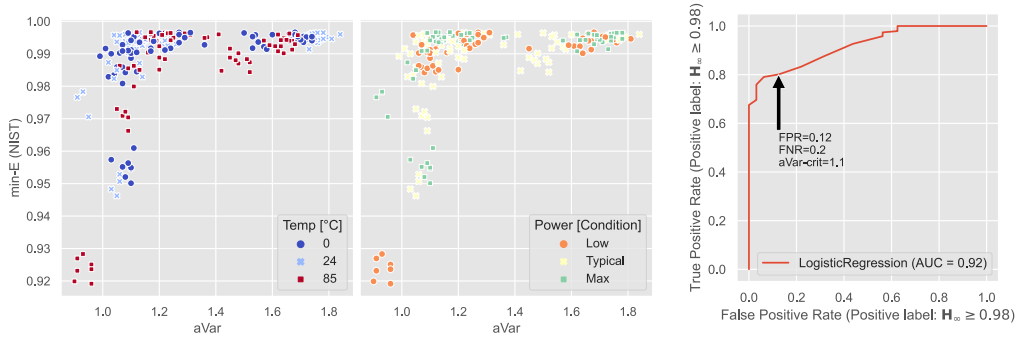
We also implemented the same configurations, but with two PLL1 outputs as a way of improving the entropy rate; results are presented in the right part of Table 3. As could be expected, the number of contributing bits and the variance computed from the model are doubled. However, the difference between variances of counter values and the variance computed from the model are bigger than in configurations with one PLL1 output. Once again, this indicates an even stronger correlation in the generated signals, explained mostly by the reduced distance between most contributors.

5.3 Effectiveness of the Online test and threshold selection

As we explained in Section 4.2, the result of the Online test depends on the required threshold value for the Allan variance of the counter values, derived from the required entropy rate (e.g. $H_1 = 0.9998$ or $H_\infty = 0.98$ according to [PS22]). We use the stochastic

model from Section 3 to compute this threshold to obtain 1.06 for the required Shannon entropy (H_1) and 1.09 for the required min-entropy (H_∞). The developer will have the responsibility to claim either one or the other (or both) of these required entropy rates.

To assess the effectiveness of the Online test, we collected data samples over a grid of environmental parameters regulated by temperature and power supply. We considered three values for the temperature, which correspond to the used commercial version of devices (low – 0°C, nominal – 24°C and high - 85°C [Int19, Xil15, Mic18]) and three values for the power supply (low, nominal, and high) depending on the device specification ($1.07, 1.10, 1.13$ for Cyclone V [Int19], $1.14, 1.20, 1.26$ for Spartan 6 [Xil15] and SmartFusion 2 [Mic18]). For each combination of card, temperature and power supply, we collected 8 data files containing 32 MB of continuous samples. We then computed the NIST min-entropy estimate and the Allan variance value for each of them, for a total of $216 = 8 \times 3 \times 3 \times 3$ acquisitions and measurements. Figure 9a presents the min-entropy estimates against the Allan variance for different temperature and power values.



(a) Min-entropy rates against the Allan variance. **(b)** Classification results.

Figure 9: Classification of high/low min-entropy rates with Allan variance (all families, all voltage and temperature conditions, Config. B).

We framed the problem as a binary classification task of high (at least 0.98) and low (below 0.98) min-entropy rate, based on the Allan variance. The Online test is expected to pass under high entropy and raise an alarm under low entropy. Various decision thresholds give rise to a trade-off between the false-positive rate (FPR) and false-negative rate (FNR). To evaluate the overall optimal accuracy, we followed the framework of Receiver Operating Characteristic (ROC) curves typically used in binary classification [Nah22].

The Allan variance value was passed through a logistic regression classifier, which achieved very high accuracy (ROC-AUC bigger than 90%). We empirically find that the variance value of 1.1 strikes a good balance across variety of families and environmental conditions (see Figure 9b). More information and the detailed version of this analysis can be found in the project repository [SDB+23].

6 Evaluation of the stochastic model assumptions

This section discusses results of experiments and statistical tests that were run to further confirm the model assumptions from Section 3.1: stationarity and correlations, both in the sampler output (in the pattern observed in periods T_P) and in the counter values. The codes and all input data are available from the GitHub repository [SDB+23].

6.1 Stationarity of the sampler output (contributors in the pattern)

We first confirmed the stationarity of the sampler output by hardware experiments. Fig. 10 and 11 show the evolution in time of sets of K_D original samples featuring the pattern and followed by the set of reordered samples representing the image of subsequent periods T_1. During the whole observation, the time position of contributing bits located at clock edges remains constant, which backs our claim of stationarity of the sampler output; at least for a sufficiently long time that the Online test can consider it stationary.

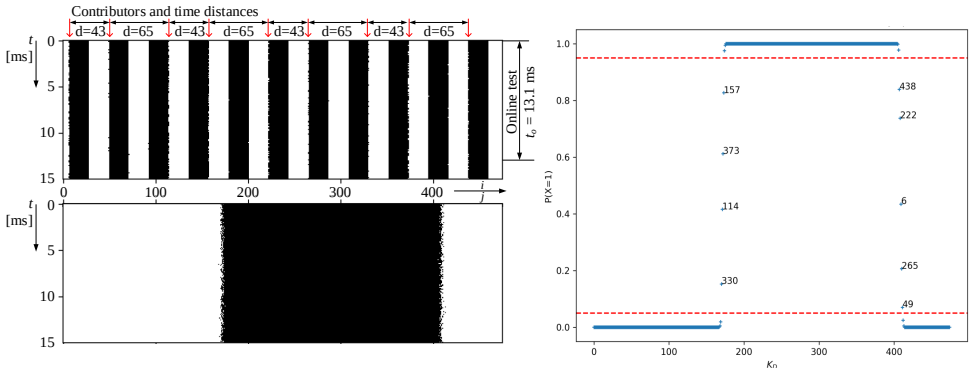


Figure 10: Evolution of K_D samples before (above) and after (below) reordering to obtain the image of period T_1 during 15 ms, black pixels are bits equal to zero, white pixels are bits equal to one (left) and reconstruction of period T_1 with indices i of contributors in the period T_P, for Configuration B within the Spartan 6 device (right).

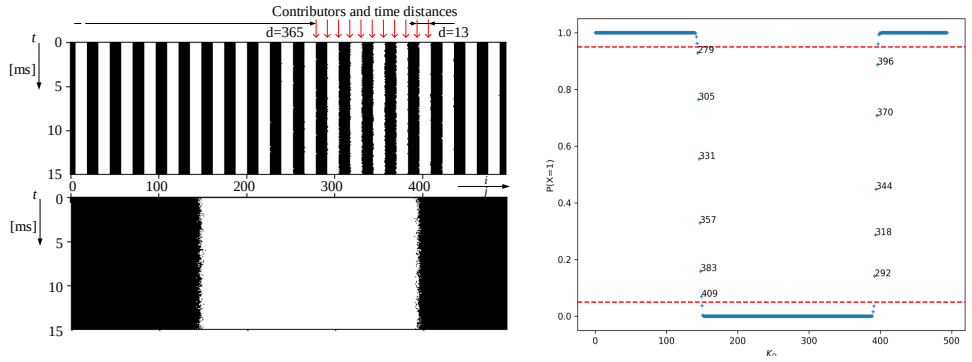


Figure 11: Evolution of K_D samples before (above) and after (below) reordering to obtain the image of period T_1 during 15 ms, black pixels are bits equal to zero, white pixels are bits equal to one (left) and reconstruction of period T_1 with indices i of contributors in the period T_P, for Configuration C within the Spartan 6 device (right).

We also performed a statistical test to confirm local stationarity of the sampler output for all three families. To this end, we run the multivariate version of the KPSS test [NH00] (referred to as loc-KPSS) over blocks of 10 000 subsequent vectors of the sampler output. We then compared the obtained test statistics against the critical value (see Table 4 for results and the code repository for details).

6.2 Stationarity of raw bits and counter values

We evaluated stationarity of raw bits and counter values using two established tests: Augmented Dickey-Fuller test [DF79] (ADF) as well as Kwiatkowski, Phillips, Schmidt,

Table 4: Results of stationarity and serial correlation tests (p-values). Tests confirming stationarity, respectively independence, at the confidence level $\alpha = 0.01$ (following [SBK+18]) are marked in green. Recall that KPSS checks the hypothesis of stationarity, ADF tests non-stationarity, and Ljung-Box asserts lack of serial correlation.

		Stationarity					Autocorrelation	
Config	Test card	Contributors loc-KPSS	Raw bit KPSS	Raw bit ADF	Counter KPSS	Counter ADF	Counter LjungBox	Raw bit LjungBox
CV_A	CV	**0.10**	**0.10**	**0.00**	0.01	**0.00**	0.00	**0.07**
S6_A	S61	**0.10**	**0.06**	**0.00**	**0.07**	**0.00**	**0.63**	**0.36**
	S62	**0.10**	**0.10**	**0.00**	**0.05**	**0.00**	**0.38**	**0.16**
SF_A	SF	**0.10**	**0.10**	**0.00**	**0.02**	**0.00**	0.00	**0.34**
CV_B	CV	**0.10**	**0.10**	**0.00**	0.01	**0.00**	0.00	**0.36**
S6_B	S61	**0.10**	**0.10**	**0.00**	0.01	**0.00**	0.00	**0.64**
	S62	**0.10**	**0.10**	**0.00**	**0.10**	**0.00**	**0.04**	**0.63**
SF_B	SF	**0.10**	**0.10**	**0.00**	0.01	**0.00**	0.00	**0.65**
CV_C	CV	**0.10**	**0.05**	**0.00**	0.01	**0.00**	**0.61**	**0.93**
S6_C	S61	**0.10**	**0.10**	**0.00**	0.01	**0.00**	**0.38**	**0.84**
	S62	**0.10**	**0.10**	**0.00**	0.01	**0.00**	**0.12**	**0.63**
SF_C	SF	0.01	0.01	**0.00**	0.01	**0.00**	0.00	0.00

and Shin test [KPSS92] (KPSS) from the package `statsmodels` [SP10]; note that in the KPSS test, reported p-values are rounded to the available range of 0.01-0.10.

The results for our three cards are presented in Table 4. Based on these results, we can make the following observations:

- Only *local stationarity* of counter values - the counter variance shows a slight stochastic drift over time, which is likely due to the presence of some low frequency noise,
- *Strict stationarity* of output bits, as confirmed by both KPSS and ADF tests.

To apply our new Online and Total Failure tests, we only need local stationarity of the counter values, so these results confirm Assumption 2.

6.3 Dependencies in raw bits and counter values

The stationarity of output bits (Section 6.2) enables testing of possible long-term dependencies in the counter output. In each experiment, 32 MB of data from the relevant configuration were used to run Ljung-Box test on both counter values and output bits, implemented in the package `statsmodels` v0.13.5 [SP10]. The results are presented in detail in the right part of Table 4; they demonstrate that:

- there are *no short or long-term dependencies* in the output bits, except on configuration SF_C where Table 3 shows the most correlation,
- there *may be dependencies* in the counter values pointed out by the LjungBox test.

Assumption 4 is therefore validated. The tests used default implementation parameters. Note that the confidence in the claim of possible dependencies in the counter values is weak, as the test may falsely reject data due to the long-term fluctuations in stationarity.

6.4 Correlation in the source of randomness

We evaluated intra-pattern dependencies in the chosen configurations, by estimating lagged correlations between contributing bits (cf. Definition 2). In each experiment, the correlation matrix was estimated using 32 MB of data; the resulting matrices have been compared between different configurations. The findings are as follows:

- We observe that correlation of contributing bits *sharply decays with distance*. This pattern is consistent across device families, cards, and even the number of PLL outputs as shown in Fig. 12.

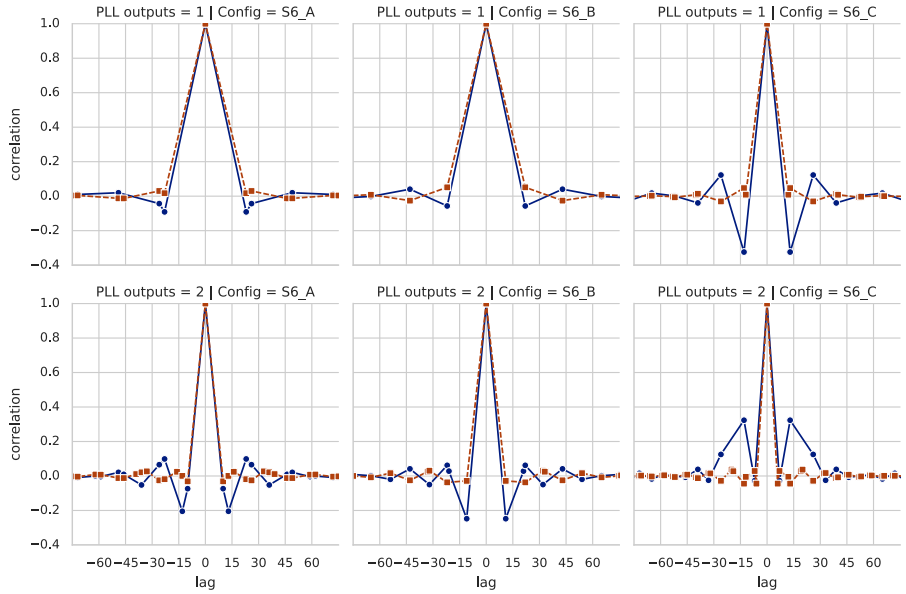


Figure 12: Correlations within the pattern period T_P in Configurations A, B, and C with one (above) and two (below) PLL1 outputs in two Spartan 6 devices (blue and red curve, respectively).

- Configuration B features the smallest correlations, while configuration C tends to have bigger ones. Although it has more contributors, most of them are close – we see a peak at distances 13 and 26. A full correlation matrix is shown in Fig. 13.
- While adding a second output to PLL1 to try and increase the bitrate of the TRNG, we see that it also has the downside of reducing the correlation decay; this limitation must not be overlooked.
- An identical analysis performed on a second Spartan 6 card shows lower correlations across the range of lags – see the red dashed graphs in Fig. 12. We attribute this to random luck, as controlling the time distances between contributors is a sufficient condition for absence of correlation but not a necessary one.

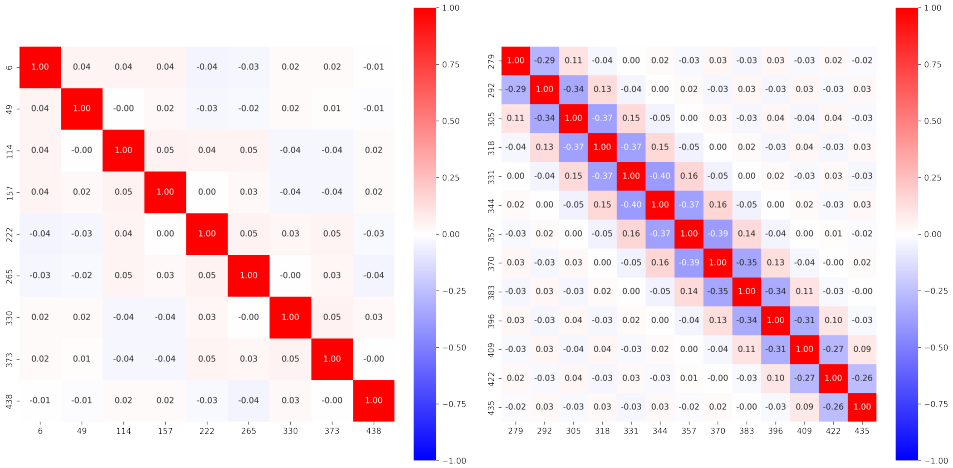
These empirical results verify Assumption 3 under the condition stated in Section 3.3 for the distance threshold between contributing bits.

7 Discussion

As we could see in Section 5.1, different FPGA families differ in degrees of freedom in the choice of the PLL-TRNG configuration. Forcing a minimal sensitivity to the jitter (S) is helpful in reducing the number of solutions, but the quality of the selected configuration has to be further verified in hardware for two reasons: first, the jitter corresponding to the selected configuration is not known beforehand; secondly the correlations between contributing bits depend on the configuration and on the jitter and may not be negligible.

Indeed, we observed that the intra-pattern correlations are hard to avoid completely. Since the available model assumes negligible correlations between samples, the designer has to find a configuration in which this assumption can be verified.

To recover some bitrate, we explored going from 1 to 2 PLL outputs – we double the number of random bits, but introduce unavoidable correlations. Empirically, the counter variance rises, but the theoretical +100% rise is not obtained because, as random bits are too close to each other, correlation reduces the individual impact of each contributor.



(a) Config S6_B **(b)** Config S6_C

Figure 13: Correlations between contributing bits within the pattern period: examples of a "good" (a) and a "bad" (b) configuration, in the Spartan 6 device S6_1.

Nevertheless, we still could find some configurations with negligible correlations (for example configuration S6_B with two PLL1 outputs).

We have to stress that thanks to the coherent sampling principle, we can observe and avoid correlations before randomness extraction (before the TDC in our case). This is not possible in any TRNG based on free-running oscillators, in which this kind of correlations certainly exist, but cannot be observed or quantified.

We observed that once a good configuration is found, the results are repeatable for the same type of hardware (i.e. card, device, and its configuration, as it can be seen in Table 4 and Fig. 12 for similar results in both Spartan 6 devices). In addition, the stationarity of the TRNG output is not an issue, primarily thanks to the behavior of the PLLs. Last but not least, since PLLs occupy a separate device area and are powered from independent power supplies, the logic running in the programmable area has practically no impact on the generated numbers.

The cost of the new PLL-TRNG version including the TDC and new embedded tests is very similar in the three tested FPGA families, especially concerning the use of PLLs, registers and DSP modules: besides the two PLLs, about 58 registers are needed to implement the core of the PLL-TRNG including two sampling registers per each PLL1 output, two times 9 registers for counters, and up to 34 registers for the FIFO. The number of required combinational blocks depends on the family: 41 ALUTs in Cyclone V, 14 LUT slices in Spartan 6 and 55 LUTs in SmartFusion2. The two embedded tests require 131 registers and one DSP module in each family and an additional 74 ALUTs in Cyclone V, 69 LUT slices in Spartan 6, and 133 LUTs in SmartFusion2.

We also tested the speed (latency) and efficiency of embedded tests. To test performance of the Total failure test, we modified the frequency of the reference clock signal clk_0 – the PLL0 was replaced by an external clock signal generator. The left panel in Fig. 14 depicts the oscilloscope screenshot when the frequency of clk_0 was modified in configuration CV_B from the nominal frequency of 237.98 MHz to 200 MHz (to simulate locking with clk_1). The Total failure alarm was triggered in 103.5 μs. In this case, the Online test alarm was triggered much later (not visible in the figure). The right panel in Fig. 14 depicts the case when the nominal frequency was changed to 231 MHz (not sufficient to trigger the Total failure alarm). The Online test was triggered in 8.5 ms, i.e. in time in which the TRNG parameters have been shown to be stationary (at least 15 ms as shown in Fig. 10 and 11).

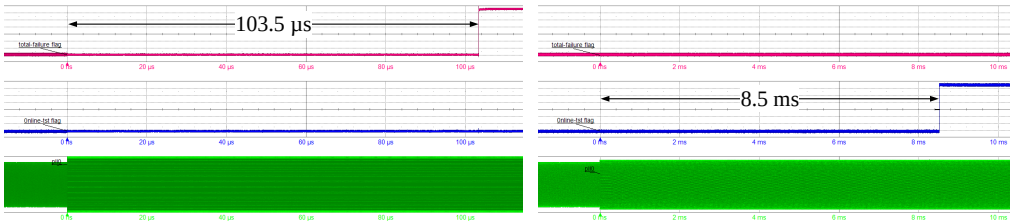


Figure 14: Total failure test (red waveform on the top) and online test (blue waveform on the bottom) alarms during total failure (left) and online (right) attacks, on the CV_B configuration featuring one PLL1 output.

8 Conclusions

In this paper, we proposed a new PLL-TRNG architecture with enhanced security. We believe that the new design approach and corresponding results have a great importance in the TRNG security in general and in the PLL-TRNG design in particular. Indeed, some ideas have a general scope and are applicable to most TRNG designs:

1. Evaluating the randomness as close to the source as possible gives much more information about the source itself, allowing for an enhanced model and increased security. Namely, we evaluate correlations before the time-to-digital conversion (TDC);
2. Evaluating the randomness using a multi-bit resolution gives more information, which our design takes advantage of – the tests can be faster and even more efficient than standard general-purpose black-box statistical tests. Namely, we model the 9-bit counter value by a Poisson binomial probability law, which we use to define meaningful thresholds for repetition counts and Allan variance with better response time and accuracy.

Some new ideas are also specific to the PLL-TRNG:

1. The study of correlation at the source of randomness is possible thanks to the knowledge of a pattern period T_P and isolation of so-called random samples – or contributors. Note that in generators using freely running oscillators the sampler output features a dynamically changing pattern and the correlations between contributors cannot be therefore evaluated. This evaluation is possible only in PLL-based TRNG, which we do for the first time in this paper;
2. We introduced a new parameter to allow for some control of the intra-pattern correlations – the time-distance between contributing bits in the pattern period T_P. Through an appropriate search in the design parameter space, we can ensure that this distance is high enough that random bits are uncorrelated, assuring in turn that the model is a better fit to the implemented design.

Acknowledgements

This work was supported by the ARSENE project funded by the “France 2030” government investment plan managed by the French National Research Agency, under the reference “ANR-22-PECY-0004”. It was also supported by the PSPC Regions AAP 1 – AURA 2019 call, funded by “Pôle de compétitivité MINALOGIC”, France. The fourth author was supported by the French National Research and Technology Agency under the CIFRE program N° 2021/0413.

References

[APFB18] Elie Noumon Allini, Oto Petura, Viktor Fischer, and Florent Bernard. Optimization of the PLL configuration in a PLL-based TRNG design. In Jan Madsen and Ayse K. Coskun, editors, *2018 Design, Automation & Test in Europe Conference & Exhibition, DATE 2018, Dresden, Germany, March 19-23, 2018*, pages 1265–1270. IEEE, 2018.

[Arm23] Arm. True Random Number Generator (TRNG). https://www.arm.com/products/silicon-ip-security/random-number-generator, 2023. Accessed: July 2023.

[Ber23] BertendSP. Physical True Random Number Generator (TRNG). https://www.bertendsp.com/products/trng-p200/, 2023. Accessed: July 2023.

[BLMT11] Mathieu Baudet, David Lubicz, Julien Micolod, and André Tassiaux. On the security of oscillator-based random number generators. *Journal of Cryptology*, 24(2):398–425, April 2011.

[CBBB20] Brice Colombier, Nathalie Bochard, Florent Bernard, and Lilian Bossuet. Backtracking Search for Optimal Parameters of a PLL-based True Random Number Generator. In *2020 Design, Automation & Test in Europe Conference & Exhibition, DATE 2020, Grenoble, France, March 9-13, 2020*, pages 1–6. IEEE, 2020.

[CFFA13] Abdelkarim Cherkaoui, Viktor Fischer, Laurent Fesquet, and Alain Aubert. A Very High Speed True Random Number Generator with Entropy Assessment. In Guido Bertoni and Jean-Sébastien Coron, editors, *Cryptographic Hardware and Embedded Systems - CHES 2013 - 15th International Workshop, Santa Barbara, CA, USA, August 20-23, 2013. Proceedings*, volume 8086 of *Lecture Notes in Computer Science*, pages 179–196. Springer, 2013.

[Dav02] Robert B. Davies. Exclusive OR (XOR) and hardware random number generators. http://www.robertnz.net/pdf/xor2.pdf, February 2002. Accessed: July 2023.

[DDS18] Nicola Da Dalt and Ali Sheikholeslami. *Understanding Jitter and Phase Noise*. Cambridge University Press, 2018.

[DF79] David A. Dickey and Wayne A. Fuller. Distribution of the estimators for autoregressive time series with a unit root. *Journal of the American statistical association*, 74(366a):427–431, 1979.

[DR23] Design-Reuse. True Random Number Generator (TRNG) IP. https://www.design-reuse.com/sip/true-random-number-generator-trng-ip-ip-47583/, 2023. Accessed: July 2023.

[FBB19] Viktor Fischer, Florent Bernard, and Nathalie Bochard. Modern Random Number Generator Design - Case Study on a Secured PLL-based TRNG. *it Inf. Technol.*, 61(1):3–13, 2019.

[FD03] Viktor Fischer and Milos Drutarovský. True random number generator embedded in reconfigurable hardware. In Burton S. Kaliski Jr., Çetin Kaya Koç, and Christof Paar, editors, *CHES 2002*, volume 2523 of *LNCS*, pages 415–430. Springer, Heidelberg, August 2003.

[FL14] Viktor Fischer and David Lubicz. Embedded evaluation of randomness in oscillator based elementary TRNG. In Lejla Batina and Matthew Robshaw, editors, *CHES 2014*, volume 8731 of *LNCS*, pages 527–543. Springer, Heidelberg, September 2014.

[GGFZ22] Davide Galli, Andrea Galimberti, William Fornaciari, and Davide Zoni. On the effectiveness of true random number generators implemented on FPGAs. In *Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS2022)*, pages 315–326. Springer International Publishing, 2022.

[Hon13] Yili Hong. On computing the distribution function for the Poisson binomial distribution. *Computational Statistics & Data Analysis*, 59:41 – 51, 2013.

[HTBF14] Patrick Haddad, Yannick Teglia, Florent Bernard, and Viktor Fischer. On the Assumption of Mutual Independence of Jitter Realizations in P-TRNG Stochastic Models. In *Design, Automation & Test in Europe Conference & Exhibition - DATE 2014*, pages 1–6. IEEE, 2014.

[IC23] Secure IC. True Random Number Generator (TRNG) IP. https://www.secure-ic.com/products/issp/security-ip/key-management/true-random-number-generator-ip/, 2023. Accessed: July 2023.

[Int19] Intel. Cyclone V Datasheet, CV-51002. https://www.alldatasheet.com/datasheet-pdf/pdf/1179747/INTEL/CV-51002.html, December 2019. Accessed: July 2023.

[IPC23] IPCores. TRNG1 – True Random and Pseudorandom Number Generator Core. http://www.ipcores.com/True_Random_Generator_TRNG_IP_core.htm, 2023. Accessed: July 2023.

[KPSS92] Denis Kwiatkowski, Peter CB. Phillips, Peter Schmidt, and Yongcheol Shin. Testing the null hypothesis of stationarity against the alternative of a unit root: How sure are we that economic time series have a unit root? *Journal of Econometrics*, 54(1-3):159–178, 1992.

[KS08] Wolfgang Killmann and Werner Schindler. A design for a physical RNG with robust entropy estimators. In Elisabeth Oswald and Pankaj Rohatgi, editors, *CHES 2008*, volume 5154 of *LNCS*, pages 146–163. Springer, Heidelberg, August 2008.

[KS11] Wolfgang Killmann and Werner Schindler. A proposal for: Functionality classes for random number generators, version 2.00. https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Zertifizierung/Interpretationen/AIS_31_Functionality_classes_for_random_number_generators_e.pdf?__blob=publicationFile, 2011. Accessed: July 2023.

[Lat23] LatticeSemiconductors. XIP8001B: True Random Number Generator (TRNG). https://www.latticesemi.com/products/designsoftwareandip/intellectualproperty/ipcore/xipheracores/xip8001b, 2023. Accessed: July 2023.

[Mic18] Microsemi. DS0128: IGLOO2 and SmartFusion2 Datasheet, Rev. 102. https://dtsheet.com/doc/1866696/ds0128--igloo2-and-smartfusion2-datasheet, 2018. Accessed: July 2023.

[MLC+14] Yuan Ma, Jingqiang Lin, Tianyu Chen, Changwei Xu, Zongbin Liu, and Jiwu Jing. Entropy evaluation for oscillator-based true random number generators. In Lejla Batina and Matthew Robshaw, editors, *CHES 2014*, volume 8731 of *LNCS*, pages 544–561. Springer, Heidelberg, September 2014.

[Nah22] Francis Sahngun Nahm. Receiver operating characteristic curve: overview and practical use for clinicians. *Korean journal of anesthesiology*, 75(1):25–36, 2022.

[NH00] Jukka Nyblom and Andrew Harvey. Tests of Common Stochastic Trends. *Econometric Theory*, 16(2):176–199, 2000.

[PMB+16] Oto Petura, Ugo Mureddu, Nathalie Bochard, Viktor Fischer, and Lilian Bossuet. A Survey of AIS-20/31 Compliant TRNG Cores Suitable for FPGA Devices. In *26th International Conference on Field Programmable Logic and Applications (FPL2016)*, pages 1–10, 2016.

[PS22] Matthias Peter and Werner Schindler. A proposal for Functionality Classes for Random Number Generators, version 2.35 - DRAFT. https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Certification/Interpretations/AIS_31_Functionality_classes_for_random_number_generators_e.pdf?__blob=publicationFile&v=5, 2022. Accessed: July 2023.

[PV22] Adriaan Peetermans and Ingrid Verbauwhede. An energy and area efficient, all digital entropy source compatible with modern standards based on jitter pipelining. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2022(4):88–109, August 2022.

[Ram23] Rambus. TRNG-IP-76 FIPS-Certified True Random Number Generators. https://www.rambus.com/security/crypto-accelerator-cores/trng-ip-76/, 2023. Accessed: July 2023.

[SBK+18] Meltem Sonmez, Elaine Barker, John Kelsey, Kerry McKay, Mary Baish, and Mike Boyle. NIST SP 800-90B: Recommendation for the Entropy Sources Used for Random Bit Generation. https://doi.org/10.6028/NIST.SP.800-90b, October 2018.

[SDB+23] Maciej Skórski, Quentin Dallison, Nathalie Bochard, Florent Bernard, and Viktor Fischer. Enhancing Quality and Security of the PLL-TRNG: Code and Data. https://github.com/maciejskorski/enhanced-pll-trng, July 2023.

[Sil23] SilexInsights. TRUE RANDOM NUMBER GENERATOR (TRNG). https://china.origin.xilinx.com/content/dam/xilinx/publications/solution-briefs/partner/silexinsights-solution-brief.pdf, 2023. Accessed: July 2023.

[SP10] Skipper Seabold and Josef Perktold. Statsmodels: Econometric and Statistical Modeling with Python. In *9th Python in Science Conference*, pages 92–96, 2010.

[Syn23] Synopsys. Synopsys True Random Number Generators. https://www.synopsys.com/dw/ipdir.php?ds=security-random-number-generator, 2023. Accessed: July 2023.

[VABF10] Boyan Valtchanov, Alain Aubert, Florent Bernard, and Viktor Fischer. Characterization of randomness sources in ring oscillator-based true random number generators in FPGAs. *13th IEEE Symposium on Design and Diagnostics of Electronic Circuits and Systems*, pages 48–53, 2010.

[Xil15] Xilinx. Spartan-6 FPGA Data Sheet: DC and Switching Characteristics (DS162). https://docs.xilinx.com/v/u/en-US/ds162, 2015. Accessed: July 2023.

[YRG+18] Bohan Yang, Vladimir Rožić, Miloš Grujić, Nele Mentens, and Ingrid Verbauwhede. ES-TRNG: A high-throughput, low-area true random number generator based on edge sampling. *IACR TCHES*, 2018(3):267–292, 2018. https://tches.iacr.org/index.php/TCHES/article/view/7276.

[ZGS+21] Alexander Zeh, Andy Glew, Barry Spinney, Ben Marshall, Daniel Page, Derek Atkins, Ken Dockser, Markku-Juhani O. Saarinen, Nathan Menhorn, and Richard Newell. RISC-V cryptographic extension proposals. https://github.com/riscv/riscv-crypto, 2021. Accessed: July 2023.