ES-TRNG: A High-throughput, Low-area True Random Number Generator based on Edge Sampling

Bohan Yang, Vladimir Rožić, Miloš Grujić,
Nele Mentens and Ingrid Verbauwhede

COSIC, KU Leuven, Belgium
{firstname.lastname}@esat.kuleuven.be

Abstract. In this paper we present a novel true random number generator based on high-precision edge sampling. We use two novel techniques to increase the throughput and reduce the area of the proposed randomness source: variable-precision phase encoding and repetitive sampling. The first technique consists of encoding the oscillator phase with high precision in the regions around the signal edges and with low precision everywhere else. This technique results in a compact implementation at the expense of reduced entropy in some samples. The second technique consists of repeating the sampling at high frequency until the phase region encoded with high precision is captured. This technique ensures that only the high-entropy bits are sent to the output. The combination of the two proposed techniques results in a secure TRNG, which suits both ASIC and FPGA implementations. The core part of the proposed generator is implemented with 10 look-up tables (LUTs) and 5 flip-flops (FFs) of a Xilinx Spartan-6 FPGA, and achieves a throughput of 1.15 *Mbps* with 0.997 bits of Shannon entropy. On Intel Cyclone V FPGAs, this implementation uses 10 LUTs and 6 FFs, and achieves a throughput of 1.07 *Mbps*. This TRNG design is supported by a stochastic model and a formal security evaluation.

Keywords: Hardware random number generators · ring oscillators · entropy · FPGA · stochastic model

1 Introduction

True Random Number Generators (TRNGs) are essential building blocks of modern embedded security systems. They enable various cryptographic algorithms, protocols and secured implementations by providing secret keys, initialization vectors, random challenges and masks. The security of these applications relies on the uniformity and unpredictability of the utilized random numbers. A cause of failure in today's security systems is often traced to a design flaw or an active attack on the used TRNG [Mar03, BCC+13] rather than to a broken cryptographic algorithm or an unprotected implementation.

True randomness cannot be obtained via computational methods. Instead, physical phenomena such as noise in electronic devices should be the source of the unpredictable nature of TRNGs. Due to their importance for security, TRNGs are subjected to strict evaluations in the process of industrial certification. The SP 800-90B [TBK+18], a special publication of the national institute for standards and technology (NIST), contains requirements for the design and evaluation of TRNGs. According to this document, a theoretical rationale for the unpredictable behavior of the entropy source is required. A min-entropy estimation of the generated output and effective online tests are also mandatory. The German BSI standard called AIS-31 [KS11] put forward a stricter requirement for the

IACR Transactions on Cryptographic Hardware and Embedded Systems ISSN 2569-2925,
Vol. 2018, No. 3, pp. 267–292
DOI:10.13154/tches.v2018.i3.267-292

design and the evaluation of TRNGs. This standard requires a formal security analysis of the TRNG design based on a stochastic model of the entropy source. The generated random bits need to provide a Shannon-entropy level of at least $0.997/bit$.

Examples of TRNGs in literature include Intel's embedded RNG core called μRNG [MJS+16], ASIC designs [PSR06] and FPGA implementations (e.g. [CFFA13, MKD11]). Several types of physical phenomena are explored to design an entropy source for TRNGs, including thermal noise [BPPT06], chaos [Gol06], timing jitter [SMS07, SPV06, WT08, RYDV15] and metastability [VHKK08, DGH09, GP09, VD10]. Many of these TRNGs are designed without accounting for AIS-31 compliance, so they are not supported by stochastic models and formal security analysis. In addition, some designs use unrealistic assumptions of the platform parameters (e.g. overestimated strength of the jitter) thereby resulting in an overly optimistic entropy assessment.

Our goal is to develop a lightweight, AIS-31 compliant TRNG with conservative entropy estimation. The contributions of this paper are the following:

1. We propose a novel, lightweight randomness generator called ES-TRNG: Edge-sampling based True Random Number Generator. ES-TRNG is a timing jitter based TRNG, suitable for both ASIC and FPGA implementations.

2. We present two design techniques to achieve a better sampling efficiency, while keeping the simplicity of implementation. These two techniques, called variable-precision phase encoding and repetitive sampling, are used for optimizing our ES-TRNG but they possibly have applications in other TRNG designs.

3. We analyze the security of ES-TRNG using a mathematical model based on the stochastic nature of the timing jitter and the sampling procedure.

4. We implement ES-TRNG on two commercial FPGAs as a case study to prove the design concept and validate the stochastic model.

5. We discuss the dangers of using unrealistic assumptions of the jitter strength in ring-oscillator-based TRNGs.

This paper is organized as follows. In Section 2, we present the preliminaries and notation used in the paper. Section 3 presents the architecture of the proposed entropy source. The security analysis based on the stochastic model of the entropy source is presented in Section 4. A link between the stochastic model and experimental results is provided in Section 5. Section 6 summarizes the FPGA implementation and application of the stochastic model for finding the optimal design parameters. Section 7 presents a comparison with the state of the art. Section 8 concludes the paper.

2 Preliminaries

2.1 TRNG design procedure

Various design criteria need to be taken into account when designing a TRNG. Conventional design goals include resource consumption, throughput, latency, feasibility in the target platform and design effort. An ongoing trend started more than a decade ago towards following security criteria during the TRNG design procedure. Some essential requirements are resistance against attacks, a stochastic model to prove unpredictability, and inner testability of the entropy source. The indispensable design criterion is a security assessment based on a realistic and applicable stochastic model.

Figure 1 shows both the old and the modern TRNG design procedure. An old and obsolete design approach relies on statistical evaluations on the output of the proposed TRNG. Most commonly used statistical test suites are NIST SP 800-22 [RSN+10], FIPS

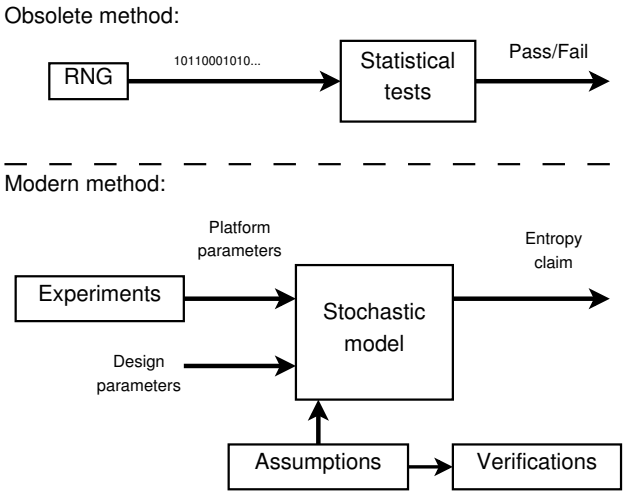


Figure 1: TRNG Design methods.

140-1 [FIP94] and DIEHARD [Mar98]. However, these test suites are only checking statistical parameters of the generated bits. A completely deterministic pseudo-random sequence could pass those test suites, despite having no randomness.

In a modern design approach, a stochastic model is used to evaluate the unpredictability of the proposed TRNG design and, furthermore, is a compulsory requirement of BSI AIS-31 [KS11]. Statistical test suites only function as a sanity check or a prototype evaluation. The non-determinism of a TRNG should be rooted in an unpredictable physical process that evolves over time. A notable example is the timing phase jitter in a free-running ring oscillator. A stochastic model is a simplified abstraction of this process based on clearly stated and verifiable assumptions. Platform and design parameters serve as inputs to a stochastic model to enable the entropy estimation of a TRNG. Platform parameters, such as the delays of logic gates and the jitter strength, should be evaluated for the target platform. Design parameters, such as the sampling frequency, should be determined according to the design requirements.

Figure 2 shows a generic architecture of a TRNG. The entropy source is the only component in the architecture with non-deterministic behavior. All randomness is generated by the entropy source, in some cases in the form of analog signals. A digitization module is needed when converting these analog signals into a digital form. Implementations of the entropy source and digitization module comprise the *Digital Noise Source*.

The *raw random numbers* are the output of the digital noise source and should be available for inner testability. Raw random numbers are usually subject to statistical defects, such as the bias from an ideal probability of ones and the auto-correlation between output bits.

The post-processing module is utilized to enhance statistical and security characteristics of the TRNG. There are two types of post-processing: the algorithmic post-processing and cryptographic post-processing. The algorithmic post-processing is utilized to extract entropy from the raw random numbers to increase the entropy per bit. The cryptographic post-processing is used to provide additional security properties, such as the backtracking resistance. We note that, according to AIS-31 [KS11], there is a minimal requirement of entropy level on the input data of cryptographic post-processing. The output data of the post-processing function are referred to as *the internal random numbers*. The post-processing is optional and not required if the entropy of the raw random numbers is sufficient.

The online tests, also called embedded tests or continuous tests, are used to detect

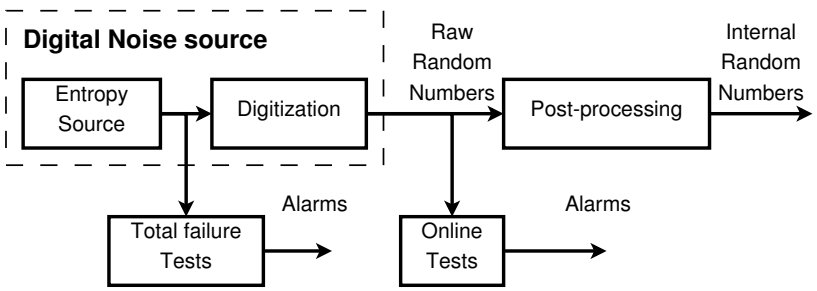


Figure 2: The generic architecture of a TRNG.

failures of the entropy source. They operate on the raw random numbers rather than the internal random numbers for faster response and more reliable attack detection. Online tests can be derived and implemented based on the stochastic model [KS11] or empirically [YRM+16]. Total failure tests are implemented to detect the total breakdown of the entropy source. They are intended to check the working status of the entropy source and trigger an alarm immediately after the breakdown.

2.2 Notation and definitions

- $Pr(a)$ – The probability of the event a.
- $E(X)$ – The expected value of the random variable X.
- $\sigma^2(X)$ – The variance of the random variable X.
- $\sigma(X)$ – The standard deviation of the random variable X.
- $\mathcal{N}(\mu, \sigma^2)$ – Normal distribution with mean μ and standard deviation σ.
- $f_{\mu,\sigma}(x)$ – The probability density of $\mathcal{N}(\mu, \sigma^2)$.

$$f_{\mu,\sigma}(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} . \tag{1}$$

- $\rho_X(x)$ – The probability density of the random variable X defined over the domain $\mathbb{R}$.

$$\int_{-\infty}^{\alpha} \rho_X(x)dx = Pr(X \leq \alpha), \forall \alpha \in \mathbb{R}. \tag{2}$$

- $\rho_{X|a}(x)$ – The conditional probability density of the random variable X defined over the domain $\mathbb{R}$, given that the event a occurred .

$$\int_{-\infty}^{\alpha} \rho_{X|a}(x)dx = Pr(X \leq \alpha|a), \forall \alpha \in \mathbb{R}. \tag{3}$$

- $H_1(X)$ – The Shannon entropy of a discrete random variable X with outcomes 0 and 1. If p denotes the probability $Pr(X = 1)$, then the Shannon entropy is computed as follows:

$$H_1(X) = -p \cdot log_2(p) - (1-p) \cdot log_2(1-p). \tag{4}$$

- $H_\infty(X)$ – The min-entropy of a discrete random variable X with outcomes 0 and 1. If p denotes the probability $\max_{i\in\{0,1\}}(Pr(X=i))$, then the min-entropy is computed as follows:
$$H_\infty(X) = -log_2(p). \tag{5}$$

- Given a set $S = [a_1, b_1) \cup [a_2, b_2) \cup \cdots \cup [a_i, b_i)$, where the $[a_1, b_1) \cdots [a_i, b_i)$ are mutually disjoint real intervals, an integral of function $f(x)$ over S is the sum of integrals of $f(x)$ over all intervals:
$$\int_S f(x)dx = \sum_{k=1}^{i} \int_{a_k}^{b_k} f(x)dx. \tag{6}$$

- Given a set of discrete distributions $F(\Theta)$ defined over the same domain and parameterized by Θ which is distributed according to the distribution G (referred to as *the mixing distribution*), then *the compound distribution* J of $F(\Theta)$ and G is given by integrating parameter Θ against the probability density ρ_G of the mixing distribution G.
$$Pr(J=i) = \int_G Pr(F(\Theta)=i)\rho_G(\Theta)d\Theta\,, \tag{7}$$
where the integration is done across the the whole domain of G and i denotes an arbitrary element from the domain of $F(\Theta)$.

- The probability distribution of the sum of two independent random variables is equal to the convolution of their distributions. If $f(x)$ and $g(x)$ are the probability distributions of two independent random variables, their convolution is:
$$(f*g)(x) = \int_{-\infty}^{\infty} f(x-\mu)g(\mu)d\mu. \tag{8}$$

3 ES-TRNG Architecture

In this section, we present the architecture of ES-TRNG and the design rationale over conventional design criteria. These design criteria include: a compact implementation, a reasonably high throughput, feasibility on various implementation platforms and a low engineering effort. Our ES-TRNG architecture is based on high-precision edge sampling. The randomness source of the ES-TRNG is the timing phase jitter from a free-running ring oscillator. Two novel techniques are used to improve the throughput and reduce the resource consumption. The first technique is called variable-precision phase encoding. By using the selective high-precision sampling process, this technique enables a compact implementation and a short jitter accumulation time. The second technique is repetitive sampling, which allows multiple sampling within a single system clock cycle. Due to the repetitive sampling, ES-TRNG can obtain a higher throughput. By using a fully digital architecture and not relying on any technology specific components, we obtain a design feasible on a wide range of implementation platforms.

The architecture of the digital noise source is shown in the left part of Figure 3. The entropy source denoted by *RO1* is implemented as a free-running ring oscillator with an enable signal. The average period of *RO1* is denoted by T_{01}. The output signal of *RO1* propagates through the *Digitization* module.

The digitization module consists of three main components, namely, a tapped delay chain, a sampling free-running ring oscillator *RO2* and a bit extractor. The average period of *RO2* is denoted by T_{02}. The used tapped delay chain consists of four cascaded delay

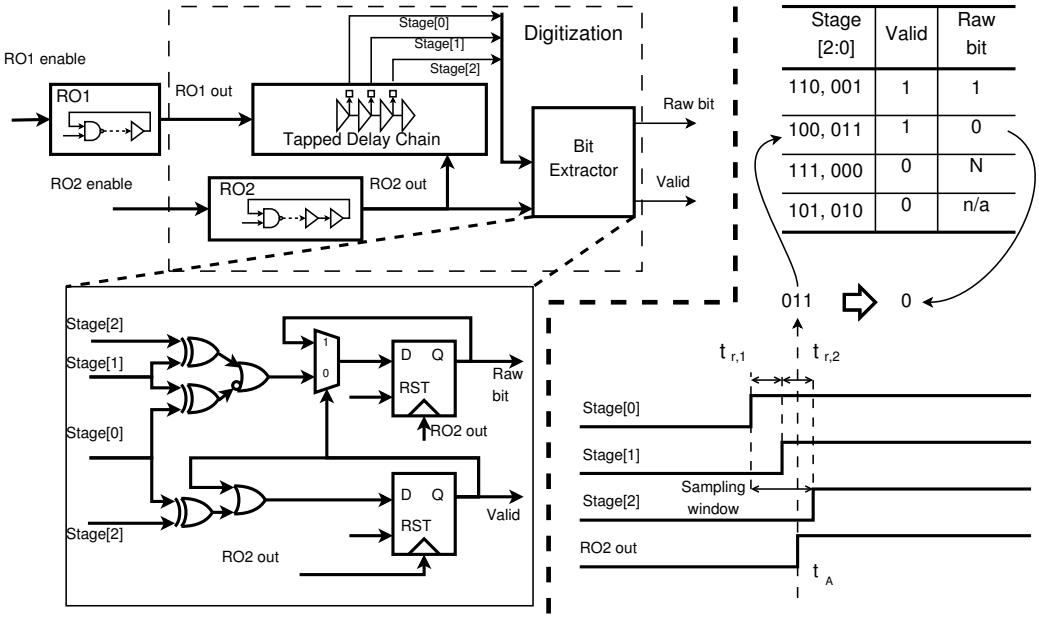


Figure 3: The architecture and operation principle of the digital noise source.

elements. The first and last delay elements are used as isolation buffers to provide similar input and output loads for the two delay elements in the middle. These middle elements form a two-stage delay chain. The $0 \rightarrow 1$ delay and the $1 \rightarrow 0$ delay of the first stage in the tapped delay chain are denoted by $t_{r,1}$ and $t_{f,1}$ respectively. Similarly, notations $t_{r,2}$ and $t_{f,2}$ are used for the rising delay and the falling delay of the second stage. The output of this two-stage tapped delay chain is sampled using three FFs (Flip-Flops) at the rising edge of the output signal of RO2.

As depicted on the right part of Figure 3, the sampled three-bit signal $Stage[2:0]$ is processed using the *Bit Extractor*. According to the truth table in Figure 3, the bit extractor encodes the three-bit input $Stage[2:0]$ to a single data bit called *Raw bit* and a strobe signal *Valid*. In the notation used in this truth table, N stands for non-valid value which is not used to generate output bits.

The lower part of Figure 3 provides the architecture of the bit extractor. This module has a compact implementation using a single LUT and two FFs on modern FPGAs. The simplicity of the bit extractor also results in a low latency which enables correct operation at high clock frequencies.

3.1 Platform and design parameters

Table 1 summarizes the relevant parameters. Platform parameters are those parameters that are specific to an implementation platform and that cannot be changed by the designer. Their values have to be obtained experimentally. Parameter σ_m^2/t_m reflects the strength of the white noise in the timing jitter. Parameters $t_{r/f,1/2}$ are properties of hardware primitives on FPGA fabric. Design parameters of the proposed TRNG are the periods T_{01} and T_{02} of *RO1* and *RO2*, the system clock frequency, and the jitter accumulation time.

3.2 Two novel techniques

In order to achieve a compact implementation with high throughput, the proposed design utilizes two novel techniques, namely variable-precision phase encoding and repetitive

Table 1: Platform and design parameters.

Platform parameters.	
$t_{r,1}$	The $0 \rightarrow 1$ delay of the 1^{st} stage in the tapped delay chain
$t_{r,2}$	The $0 \rightarrow 1$ delay of the 2^{nd} stage in the tapped delay chain
$t_{f,1}$	The $1 \rightarrow 0$ delay of the 1^{st} stage in the tapped delay chain
$t_{f,2}$	The $1 \rightarrow 0$ delay of the 2^{nd} stage in the tapped delay chain
D	The duty cycle of the free-running RO1
σ_m^2/t_m	The variance of a white-noise jitter accumulated during the measurement time t_m
Design parameters.	
T_{01}	The average free-running RO1 period
T_{02}	The period of the sampling clock RO2
T_{CLK}	The period of the system clock
t_A	Jitter accumulation time

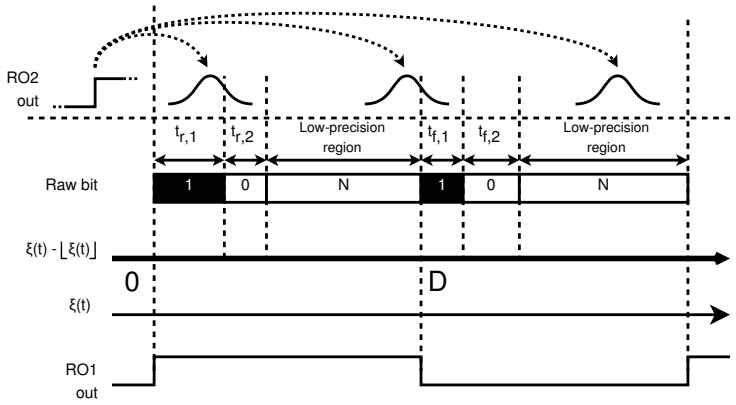


Figure 4: Variable-precision phase encoding.

sampling. The central idea is to repeat the sampling and ignore all samples from the low-precision region (the region where values 000 and 111 are sampled in the tapped delay chain).

3.2.1 Variable-precision phase encoding

The variable-precision phase encoding technique is shown in Figure 4. This technique is enabled by using both the tapped delay chain and the bit extractor. The bottom part of the figure shows a single period of *RO1*. Symbol $\xi(t)$ denotes the normalized time in the unit of T_{01}. The phase of the oscillator *RO1*, denoted by $\xi(t) - \lfloor \xi(t) \rfloor$, changes periodically increasing from 0 to 1 within a single period.

The position of the captured edge is encoded into a raw bit. Since the delays $t_{r/f,1}$ and $t_{r/f,2}$ of the tapped delay chain are much smaller than the oscillation period T_{01}, the digitization module captures the oscillator phase with high precision around signal edges when the phase value is around 0 or D. This region is called the *high precision region*. The samples from this region are encoded by either 0 or 1 depending on the phase. In the remainder of the cycle, the edge is captured with low precision, i.e. only the correct half-period can be determined from the captured data. This region is called the *low precision region*. The samples from this region are not used (encoded as N).

The sampling of the delay chain is triggered by the rising edge of the signal *RO2 out*. Due to the accumulated timing jitter, the relative sampling position follows a Gaussian distribution. Increased accumulation time t_A leads to a wider jitter distribution. The

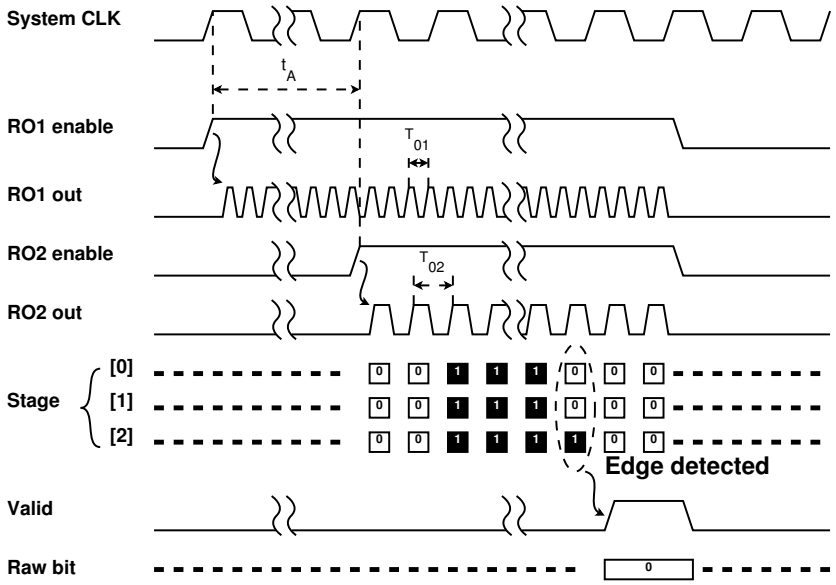


Figure 5: Timing diagrams of repetitive sampling.

expected value of the sampling time is determined by t_A, the number of ignored low-precision samples and the initial phase offset between $RO1$ *out* and $RO2$ *out*. This expected value (the center of the Gaussian distribution) can appear at any position, as indicated at the top of the figure.

We note that using a delay line with more than two elements would extend a region that is sampled with high precision, at the cost of increased area and energy consumption. A design using the precise sampling for all phase values is presented in [RYDV15].

3.2.2 Repetitive sampling

The second technique used in the proposed design is called repetitive sampling. The proposed digitization module has a small critical path, which enables the digital noise source operating at a higher frequency than other components. Repetitive sampling is synchronized to the high frequency signal $RO2$ *out*, aiming to reduce the time needed to hit the high-precision region, thereby improving the throughput. Once the high-precision region is hit, a *Valid* signal is generated.

The timing diagram of the digital noise source is depicted in Figure 5 to illustrate the repetitive sampling. The entropy source *RO1* is reset before generating a bit and then enabled for a period of time t_A. The parameter t_A is chosen to allow accumulating enough jitter at the entropy source. After time t_A has passed, the sampling oscillator *RO2* is enabled. The $RO1$ *out* signal edge propagating through the tapped delay chain is sampled using $RO2$ *out*. This sampling can be repeated multiple times within a single system clock cycle, because the frequency of a free-running *RO2* can be higher than the system clock frequency available on FPGAs. Once the tapped delay chain stages $Stage[2:0]$ reach the high-precision region, the *Valid* signal will be set, the raw bit will be encoded and *RO1* and *RO2* will be reset.

4 Security Analysis

The formal security analysis provides the theoretical guarantees for the quality of the generated bits. These guarantees are achieved by developing an entropy estimator based

on the stochastic model of the TRNG. The conservative entropy estimator provides a lower bound on the entropy based on the design parameters and the platform parameters. This estimator is used early in the design stage to guide the choice of the design parameters.

4.1 Assumptions

The assumptions about the physical processes in the entropy source are the starting point of the security analysis.

In this design the entropy is extracted from the timing jitter of a free-running ring oscillator. Our first assumption is that some amount of white (Gaussian) noise is present in the entropy source. The property of the white noise is that its observations are independent of each other and independent of any other noise source in the system. According to the central limit theorem, the variance of this noise increases linearly with the jitter accumulation time. The platform parameter σ_m^2/t_m is obtained experimentally. Our second assumption is that the value of this parameter is not overestimated. This means that a conservative estimation of this parameter's value must be made during the measurement procedure.

We further assume that other noise sources are present in the entropy source. These include flicker noise (low frequency noise), telegraph noise (popcorn noise) and the global noise from the power supply. None of these sources are exploited in this design because either their observations are correlated, the noise parameters are not measured or the noise source can be manipulated by the attacker. In order to provide a conservative estimation of entropy, the impact of these noise sources will be treated as deterministic and known to the attacker.

In addition, we assume that the sampled raw bits are independent because the circuit is reset before generating each bit [BL05]. We note that this doesn't imply that the raw bits are identically distributed.

Finally, we assume that the frequencies of the two free-running ring oscillators and the delays of the elements in the tapped delay line are measured with sufficient precision, such that the measurement error doesn't significantly affect the entropy estimation.

4.2 Entropy Source.

We will analyze the entropy source using the phase model of the ring oscillator with duty cycle D (in practice $D \approx 0.5$). The phase $\xi \in \mathbb{R}$ of the RO increases over time; integer phase values correspond to the rising edges of the output signal and the values $i + D, i = 0, 1, \ldots$ correspond to the falling edges. The entropy source is reset before generating a new bit, so we assume that the oscillations always start from $\xi(0) = 0$. Phase ξ is affected by both deterministic and white noise sources.

$$\xi(t) = \frac{t}{T_{01}} + \xi_{Deterministic}(t) + \xi_{Gaussian}(t)\,. \tag{9}$$

Since the average value of the white noise is zero, the expected value of the phase after time t is

$$E(\xi(t)) = \frac{t}{T_{01}} + E(\xi_{Deterministic}(t))\,. \tag{10}$$

The contribution of the deterministic noise sources $E(\xi_{Deterministic}(t))$ cannot be estimated with reasonable precision. Therefore, for entropy estimation we will always consider the worst-case value of this term.

The standard deviation of the phase is greater than the standard deviation of the white noise:

$$\begin{aligned}
\sigma^2(\xi(t)) &= \sigma^2\left(\frac{t}{T_{01}} + \xi_{Deterministic}(t) + \xi_{Gaussian(t)}\right) \\
\text{(by independence)} \qquad &= \sigma^2\left(\frac{t}{T_{01}} + \xi_{Deterministic}(t)\right) + \sigma^2\left(\xi_{Gaussian}(t)\right) \\
&\geqslant \sigma^2(\xi_{Gaussian}(t)) \\
&= \frac{\sigma_m \cdot t}{T_{01}^2 \cdot t_m}\,. \qquad (11)
\end{aligned}$$

This bound on the standard deviation can be computed for any value of t, given the physical and design parameters.

4.3 Digitization.

The ring oscillator is sampled repeatedly at moments $t_A, t_A + T_{02}, t_A + 2T_{02}, ..$ until a valid sample is detected. We denote the phase at these moments with $X_0, X_1, ...$

$$X_i = \xi(t_A + i \cdot T_{02})\,,\ i = 0, 1... \qquad (12)$$

Here we introduce an approximation that will be used for estimating probabilities:

■**Approximation 1:**

$$E(X_i) \approx E(X_0) + i \cdot \Delta\xi\,, \qquad (13)$$

where $\Delta\xi = T_{02}/T_{01}$. This assumption is justified by the fact that, for sufficiently high frequencies, the impact of the correlated noise is negligible compared to other randomness generating processes. This assumption, while not always explicitly stated, is used in state-of-the-art stochastic models of TRNG designs [FL14, HFBN15].

For the security analysis of the digitization, we define the normalized platform parameters: $d_{r,1},\ d_{f,1},\ d_{r,2},\ d_{f,2}$ denote the normalized delays of the tapped delay chain ($d_{r,l} = t_{r,l}/T_{01}$ and $d_{f,l} = t_{f,l}/T_{01}$ for $l \in \{1, 2\}$); D denotes the duty cycle of the entropy source.

In order to simplify the following analysis, we denote the standard deviation of the phase at time t_A as $\sigma_{t_A} = \sigma(X_0)$. We use $\sigma_{T_{02}} = \sigma(X_i - X_{i-1})$ to denote the standard deviation of the superimposed phase, which is the result of the white noise accumulated in T_{02}. We would like to note that the white noise accumulated during the time period between X_i and X_{i-1} for any i is independent of each other and also independent of the white noise accumulated during the accumulation time t_A.

The sampling circuit maps the phase value into the output bit. We introduce the mapping $s : \mathbb{R} \to \{0, 1, N\}$, where the symbol N denotes that a non-valid value is detected. Function $s(x)$ can be formally defined as:

$$s(x) = \begin{cases} 1, \text{ for } x \in S_1 \\ 0, \text{ for } x \in S_0 \\ N, \text{ for } x \in S_N\,, \end{cases} \qquad (14)$$

where S_1, S_0 and S_N are unions of mutually disjoint real intervals defined as:

$$S_1 = \bigcup_{k=-\infty}^{\infty} \Big\{ [k, k + d_{r,1}) \cup [k + D, k + D + d_{f,1}) \Big\}, \tag{15}$$

$$S_0 = \bigcup_{k=-\infty}^{\infty} \Big\{ [k + d_{r,1}, k + d_{r,1} + d_{r,2}) \cup [k + D + d_{f,1}, k + D + d_{f,1} + d_{f,2}) \Big\}, \tag{16}$$

$$S_N = \bigcup_{k=-\infty}^{\infty} \Big\{ [k + d_{r,1} + d_{r,2}, k + D) \cup [k + D + d_{f,1} + d_{f,2}, k + 1) \Big\}. \tag{17}$$

We stress that the sets S_1, S_0 and S_N are mutually disjoint and that:

$$S_1 \cup S_0 \cup S_N = \mathbb{R}. \tag{18}$$

For clarity reasons, we introduce the function $g : \mathbb{R} \to \{0, 1\}$.

$$g(x) = \begin{cases} 1, \text{ for } x \in S_N \\ 0, \text{ for } x \in S_1 \cup S_0 \,. \end{cases} \tag{19}$$

We note that $g(x)$ is a periodic function with the following property:
For any function $f : \mathbb{R} \to \mathbb{R}$ and any $A \subset \mathbb{R}$:

$$\int_{A \cap S_N} f(x)dx = \int_{A} f(x) \cdot g(x)dx \,. \tag{20}$$

A special case of this property is:

$$\int_{-\infty}^{\infty} f(x) \cdot g(x)dx = \int_{S_N} f(x)dx = \sum_{k=-\infty}^{\infty} \left[\int_{k+d_{r,1}+d_{r,2}}^{k+D} f(x)dx + \int_{k+D+d_{f,1}+d_{f,2}}^{k+1} f(x)dx \right]. \tag{21}$$

In line with these definitions and properties, we start the analysis of the digitization by examining the first sample taken at time t_A.

The phase at the first sampling moment is a normally distributed random variable X_0. The probability density function of this variable is:

$$\rho_{X_0}(x) = f_{E(X_0),\sigma_{t_A}}(x). \tag{22}$$

Let Y_i denote the sampled values, $Y_i = s(X_i)$. For convenience, we use the notation $\mu_i = E(X_i) - \lfloor E(X_i) \rfloor$. We can compute the probabilities of Y_0 as:

$$Pr(Y_0 = 1) = \int_{S_1} \rho_{X_0}(x)dx \,, \tag{23}$$

$$Pr(Y_0 = 0) = \int_{S_0} \rho_{X_0}(x)dx \,, \tag{24}$$

$$Pr(Y_0 = N) = \int_{S_N} \rho_{X_0}(x)dx = 1 - Pr(Y_0 = 1) - Pr(Y_0 = 0). \tag{25}$$

To simplify the notation, we let EV_i denote the event $Y_0 = N, \cdots, Y_i = N$. For example, the EV_0 denotes the event $Y_0 = N$. The second sample is only taken when EV_0 occurs. The i_{th} sample is taken when EV_{i-1} occurs.

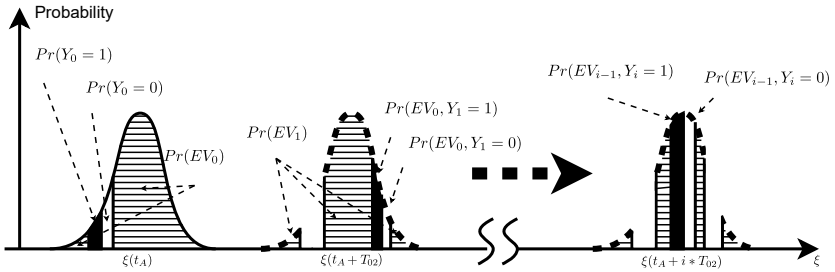


Figure 6: The intuitive interpretation of the repetitive sampling.

From Equations (21) and (25), it follows that:

$$Pr(EV_0) = Pr(Y_0 = N) = \int_{S_N} \rho_{X_0}(t)dt = \int_{-\infty}^{\infty} \rho_{X_0}(t) \cdot g(t)dt. \tag{26}$$

For events EV_i, the following property holds:

$$Pr(EV_{i-1}, Y_i = 0) + Pr(EV_{i-1}, Y_i = 1) + Pr(EV_{i-1}, Y_i = N) = Pr(EV_{i-1}). \tag{27}$$

An intuitive example of the repetitive sampling is shown in Figure 6. The probability density function $\rho_{X_0}(x)$ of the random variable $X_0 = \xi(t_A)$ is located at the left part of the figure. This distribution covers three types of regions under the curve $\rho_{X0}(x)$: the white regions, the black regions and the shaded regions, which correspond to segments of $\rho_{X_0}(x)$ over the sets S_0, S_1 and S_N respectively. The area of the white regions is equal to $Pr(Y_0 = 0)$, while the area of the black regions and the shaded regions are equal to $Pr(Y_0 = 1)$ and $Pr(Y_0 = N)$. It is described by Equations (23), (24) and (25). If the realization of X_0 is located at the shaded regions, the second sample is needed.

The middle distribution corresponds to the second sample. The middle distribution also covers three types of regions. The area of the white regions, the black regions and the shaded regions are equal to $Pr(EV_0, Y_1 = 0)$, $Pr(EV_0, Y_1 = 1)$ and $Pr(EV_1)$.

If the event EV_{i-1} occurs, the distribution of the i_{th} sample shown at the right part of the figure can be derived from the distribution of the $(i-1)_{th}$ sample.

Under the **Approximation 1**, Equation (12) can be rewritten as:

$$X_i = \xi(t_A) + i \cdot \frac{T_{02}}{T_{01}} + \sum_{k=1}^{i} Z_k = X_0 + i \cdot \frac{T_{02}}{T_{01}} + \sum_{k=1}^{i} Z_k, \tag{28}$$

where Z_i denotes a series of random variables defined as:

$$Z_i = \xi_{Gaussian}(t_A + i \cdot T_{02}) - \xi_{Gaussian}(t_A + (i-1) \cdot T_{02}). \tag{29}$$

Informally speaking, Z_i is the superimposed component of $\xi(t_A + i \cdot T_{02})$ caused by the white noise accumulated during the time interval $[t_A + (i-1) \cdot T_{02}, t_A + i \cdot T_{02})$. Z_i is a normally distributed random variable $\mathcal{N}(0, \sigma^2_{T_{02}})$.

The goal of the entropy estimation is to compute the binary probability of the raw bits. The computation of this binary probability requires $Pr(EV_{i-1}, Y_i = 1)$. Starting from the $Pr(Y_0 = 1)$ given by Equation (23), we first compute $\rho_{X_1|EV_0}(x)$ as:

$$\rho_{X_1|EV_0}(x) = \frac{1}{Pr(EV_0)} \int_{-\infty}^{\infty} f_{0,\sigma_{T_{02}}}(x - \mu) \cdot \rho_{X_0}\left(\mu - \frac{T_{02}}{T_{01}}\right) \cdot g\left(\mu - \frac{T_{02}}{T_{01}}\right) d\mu. \tag{30}$$

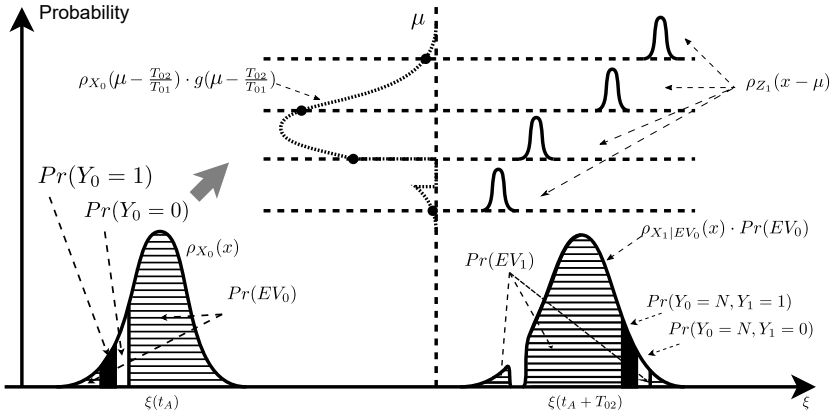


Figure 7: The intuitive interpretation of the first sample and the second sample.

The formal derivation of Equation (30) can be found in Appendix A.

An informal but intuitive interpretation of Equation (30) is shown in Figure 7. The left part of the figure is the probability density function of the $\rho_{X_0}(x)$. The black, white and shaded area under the curve correspond to $Pr(Y_0 = 1)$, $Pr(Y_0 = 0)$ and $Pr(EV_0)$ respectively. The second sample is taken, only when the event EV_0 occurs. The shaded region is shifted to the right by T_{02}/T_{01}, which corresponds to the dashed curve. The probability density function $\rho_{Z_1}(x)$ is shifted to positions with probabilities according the dashed curve. The weighted and shifted results are added together. The result is shown at the right bottom of this Figure 7.

Now the conditional probability function $\rho_{X_i|EV_{i-1}}(x)$, $(i > 1)$ can be derived iteratively as:

$$\rho_{X_i|EV_{i-1}}(x) = \frac{Pr(EV_{i-2})}{Pr(EV_{i-1})} \cdot \int_{-\infty}^{\infty} f_{0,\sigma_{T_{02}}}(x-\mu) \cdot \rho_{X_{i-1}|EV_{i-2}}\left(\mu - \frac{T_{02}}{T_{01}}\right) \cdot g\left(\mu - \frac{T_{02}}{T_{01}}\right) d\mu. \quad (31)$$

The formal derivation of Equation (31) can be found in Appendix A.

We can compute probabilities $Pr(EV_i)$ as:

$$\begin{aligned} Pr(EV_i) &= Pr(EV_{i-1}, Y_i = N) \\ &= Pr(EV_{i-1}) \cdot Pr(Y_i = N | EV_{i-1}) \\ &= Pr(EV_{i-1}) \cdot \int_{S_N} \rho_{X_i|EV_{i-1}}(x) dx. \end{aligned} \quad (32)$$

Starting with Equations (26) and (30), we can apply Equations (31) and (32) iteratively to compute probabilities $Pr(EV_i)$.

4.4 Binary probabilities

Now we can compute the following probabilities:

$$Pr(EV_{i-1}, Y_i = 1) = Pr(EV_{i-1}) \cdot \int_{S_1} \rho_{X_i|EV_{i-1}}(t) dt\,, \quad (33)$$

$$Pr(EV_{i-1}, Y_i = 0) = Pr(EV_{i-1}) \cdot \int_{S_0} \rho_{X_i|EV_{i-1}}(t) dt\,. \quad (34)$$

The $Pr(EV_{i-1}, Y_i = 0)$ can be computed using Equations (27), (33) and (34).

The value of raw bit b is determined by a sequence $(Y_0 Y_1 \cdots Y_j)$ where j is the smallest integer such that $Y_j \neq N$. Therefore, the binary probability of a raw bit can be computed as:

$$Pr(b = 1) = Pr(Y_0 = 1) + \sum_{k=1}^{\infty} Pr(EV_{k-1}, Y_k = 1)\,. \tag{35}$$

4.5 Entropy claim

Binary probabilities are computed starting from the platform parameters using Equations (11), (19), (22), (26), (30), (31), (33) and (35). The only unknown parameter is μ_0 which is equal to the phase of the oscillator at the moment t_A. The value of μ_0 depends on the global noise, low-frequency noises and the operating conditions. For this reason, it cannot be predicted with reasonable precision at design time. The conservative entropy claim is made by examining the effects of μ_0 on binary probabilities and using the value that results in the lowest entropy. This procedure is shown in the example in Section 6.

5 Experimental Validation of the Stochastic Model

We use an experimental approach to validate the proposed stochastic model, i.e. to check if the behavior of the physical TRNG matches the behavior predicted by the model. For this purpose, we implement the proposed design on a Xilinx Spartan-6 FPGA. In this experiment we monitor the number N_T of toggles of the *RO2* during the sampling phase before the raw bit is generated.

Before applying the stochastic model, we have to specify the design parameters and measure the platform parameters. Design parameters T_{01} and T_{02} are chosen indirectly by selecting the number of delay elements in *RO1* and *RO2*. We implement *RO1* and *RO2* using a single look-up table and three look-up tables respectively. The periods of *RO1* and *RO2* are then measured using two individual ripple counters, the measurement results are $T_{01} = 2171.8\ ps$ and $T_{02} = 2739.8\ ps$. In regard to other design parameters, the period of the system clock is chosen to be $T_{CLK} = 10\ ns$ and the jitter accumulates for 9 system clock cycles.

We follow the methodologies proposed in [YRG+17] and [RYDV15] to measure the platform parameters. The obtained propagation delays are $t_{r,1} = 22.25\ ps$, $t_{r,2} = 24.12\ ps$, $t_{f,1} = 35.93\ ps$ and $t_{f,2} = 40.90\ ps$. The duty cycle D measurement is 0.43 and the white-noise jitter strength is $\sigma_m^2/t_m = 0.0029\ ps$.

The number of toggles for a specific value of μ_0 is distributed as follows:

$$Pr(N_T = i) = Pr(EV_{i-1}, Y_i \neq N) = Pr(EV_{i-1}, Y_i = 1) + Pr(EV_{i-1}, Y_i = 0)\,, \tag{36}$$

which is easily computed by applying Equations (33) and (34).

Figure 8 shows the predicted toggle distributions for different values of parameter $\mu_0 \in (0, 1)$.

We note that due to a very low jitter strength, the variance of accumulated jitter at the sampling phase is much lower than the period of *RO1*, i.e. $\sigma_{T_{02}} \ll 1$. A consequence of this low jitter is that the probability $Pr(Y_i = N)$ is very low in cases when μ_i is close to the high-precision region ($\mu_i \approx 0$, $\mu_i \approx D$ and $\mu_i \approx 1$). Conversely, this probability is very high when μ_i is far away from the high-precision phase region (i.e. $\mu_i \approx D/2$ and $\mu_i \approx (1+D)/2$). We check how this affects the distribution of N_T for the selected oscillator

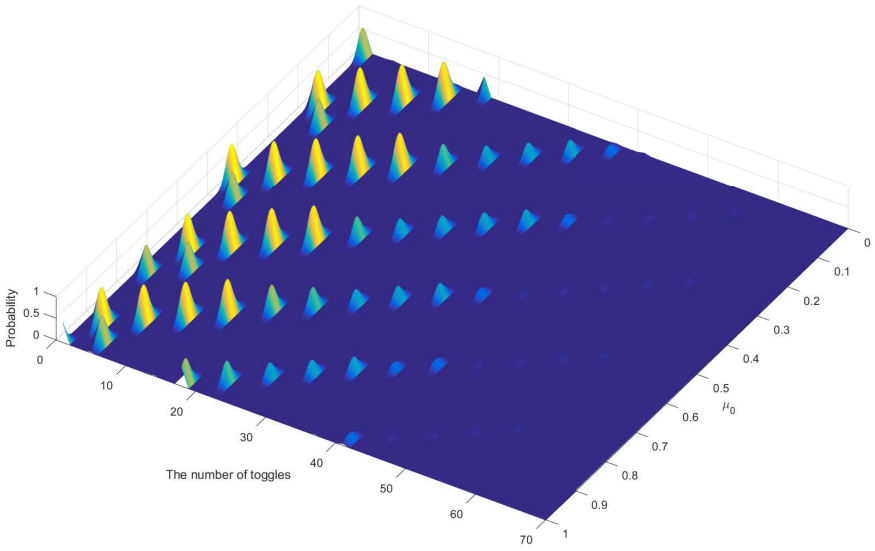


Figure 8: Probabilities of toggle numbers under different μ_0.

periods. Let's first observe that $T_{02}/T_{01} \approx 5/4$. Therefore:

$$\mu_{i+1} \approx \mu_i + 1/4 - \lfloor \mu_i + 1/4 \rfloor \,, \tag{37}$$

$$\mu_{i+2} \approx \mu_i + 1/2 - \lfloor \mu_i + 1/2 \rfloor \,, \tag{38}$$

$$\mu_{i+4} \approx \mu_i \,. \tag{39}$$

Since the high-precision regions are close to the beginning of the cycle and the middle of the cycle, we get:

$$Pr(Y_i = N) \approx Pr(Y_{i+2} = N) \,. \tag{40}$$

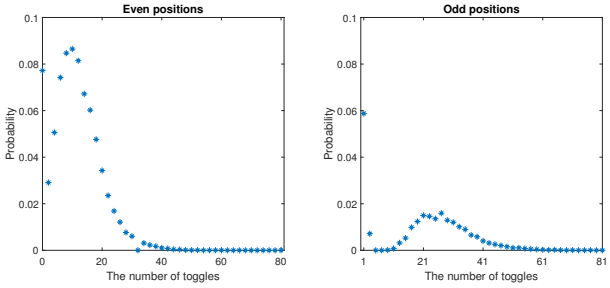
The result is that the N_T distribution for odd values is very different from the distribution for the even N_T values. This effect is observed in the model for any value of μ_0 as well as in the experimental data shown in Figure 9a. For clarity, this distribution is presented using two graphs, one for the odd N_T values and one for the even values.

In order to model the distribution of N_T we have to make assumptions about μ_0. In the used experimental setting, it is reasonable to assume that μ_0 follows a Gaussian distribution. We note that we don't make this assumption for computing the entropy, but rather use the worst case value. Therefore, the N_T distribution is computed as the compound distribution of $\mathcal{N}(\mu, \sigma^2)$ and the distribution given by Equation (36). For $\mathcal{N}(0.384, 0.09^2)$, the model reasonably approximates experimental data, as shown in Figure 9b. We note that $\mathcal{N}(0.384, 0.09^2)$ is unrelated to the platform parameters and is not used for entropy estimation. It is only used to validate the stochastic model.

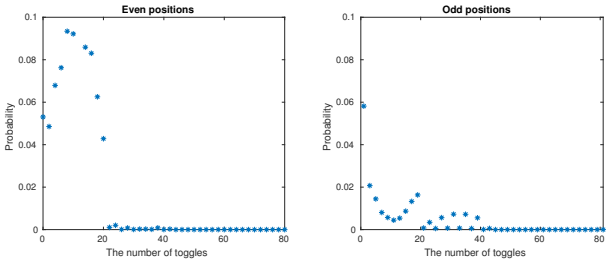
6 Implementation and Security Evaluation

6.1 Xilinx FPGA implementation

In this section, we present a Xilinx Spartan-6 FPGA implementation of ES-TRNG. In addition to look-up tables (LUTs) and sequential elements (flip-flops and latches), this



(a) Probability distribution of toggles computed from the collected experimental data.



(b) Probability distribution of toggles derived from the stochastic model.

Figure 9: Toggles probability distributions predicted by the model and obtained by experiments. Each distribution is shown using two graphs, one for the even values (left) and one for the odd values (right).

FPGA has high-speed carry chain primitives called CARRY4. These primitives can be configured to work as a tapped delay line.

Figure 10 shows the implementation of the core of the proposed digital noise source. RO1 is implemented as a high-speed oscillator using a single LUT, RO2 is implemented using 3 LUTs. A single CARRY4 element is used to implement the tapped delay chain.

The entire core of the TRNG shown in Figure 10 is implemented using only one CARRY4 element, 10 LUTs and 5 flip-flops. In addition to this core, the design contains a parity filter for post-processing and a control circuit for setting the enable and reset signals. A system clock of 100 MHz is used in the design.

6.2 Application of the model

In this section, we applied the stochastic model to derive optimal design parameters. The platform parameters measurement is described in Section 5. The stochastic model enables us to calculate the Shannon entropy and min-entropy for any value of μ_0 for a specific accumulation time. The conservative entropy estimation is made by using the global minimum. This procedure is illustrated in Figure 11. This figure shows the entropy estimations for three different values of accumulation time ($t_A = 5 \cdot T_{CLK}$, $10 \cdot T_{CLK}$ and $20 \cdot T_{CLK}$). The global minima are indicated by red dots.

To confirm that our entropy estimation is indeed conservative, we applied the entropy assessment procedure from [TBK+18] on the collected raw random numbers. The results are summarized in Table 2. The results reported by the NIST entropy assessment python package are always higher than the min-entropy estimation derived from the proposed stochastic model. This result is expected because the stochastic model estimation is based on the worst case scenario.

According to the standard [KS11], a minimal Shannon entropy level of 0.997 per bit is

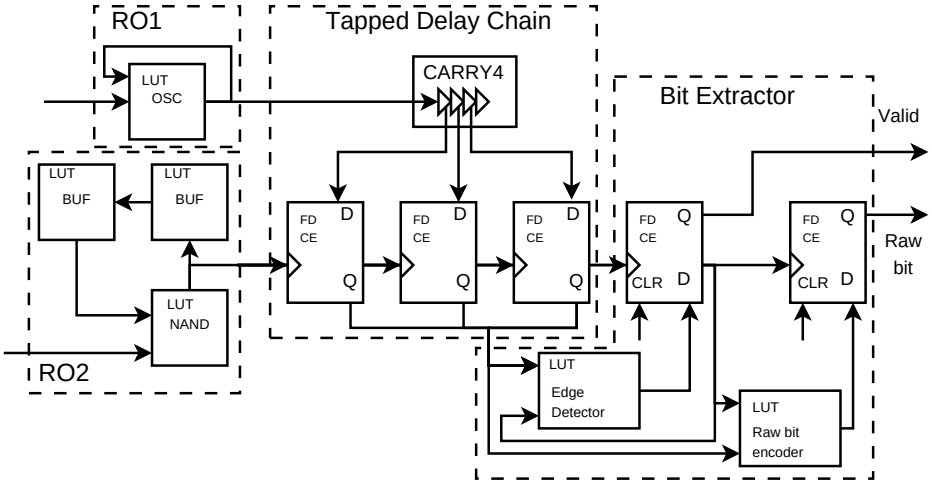


Figure 10: FPGA implementation of the digital noise source.

Table 2: Model verification

t_A (ns)	H_∞ (Stochastic model)	H_∞ (NIST SP800-90B)	H_1 (Stochastic model)
50	0.099	0.578	0.352
100	0.233	0.661	0.607
200	0.461	0.757	0.846
300	0.548	0.764	0.900

expected in the internal random numbers. This goal is achieved when the bias $\varepsilon_{internal}$ of the internal random numbers is upper-bounded by 3.2%. This Shannon entropy level cannot be achieved by accumulating the jitter less than 1 μs, which leads to a throughput smaller than 1 $Mbps$. However, a higher throughput can be obtained by using a shorter accumulation time followed by a simple algorithmic post-processing. Parity filter of order n_f, which combines n_f consecutive input bits into one output bit using a XOR function, reduces the bias $\varepsilon_{internal}$ to:

$$\varepsilon_{internal} = 2^{n_f - 1} \cdot \varepsilon_{raw}^{n_f}, \tag{41}$$

where ε_{raw} denotes the bias of raw random numbers.

To optimize the throughput of ES-TRNG, we computed the minimal n_f required to achieve the Shannon entropy level of 0.997. Figure 12 shows the lower bound estimation for H_1 and H_∞ for t_A ranging from 20 ns to 1 μs. For every t_A within this range, we calculate the expected throughput of the internal random numbers after the required n_f-stage parity filter. From the figure, we can see that the throughput increases monotonically when $t_A \leq 110$ ns. Within this region, the required n_f reduces rapidly because of the increasing of estimated Shannon entropy. There are several discontinuous segments on the throughput plot when $t_A > 110$ ns. Each segment corresponds to a different required n_f. When $t_A > 1000$ ns, the throughput is always lower than 1 $Mbps$. The global maximum of 1.15 $Mbps$ throughput is obtained at $t_A = 250$ ns where $n_f = 3$. At this point, the H_∞ of raw random numbers estimated by the stochastic model is 0.515 which is more conservative than 0.86 derived from the standard [TBK+18]. The post-processed data achieves a throughput higher than 1 $Mbps$ with a Shannon entropy level higher than 0.997. For verification, a 10 MB sequence of internal numbers was generated and tested using T0-T5 tests proposed in the AIS-31. The sequence passes all the tests.

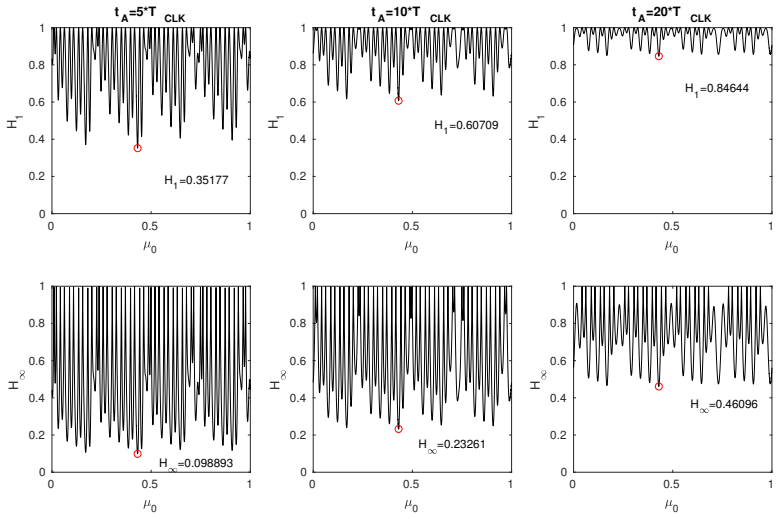


Figure 11: The estimated H_1 and H_∞ for different accumulation time.

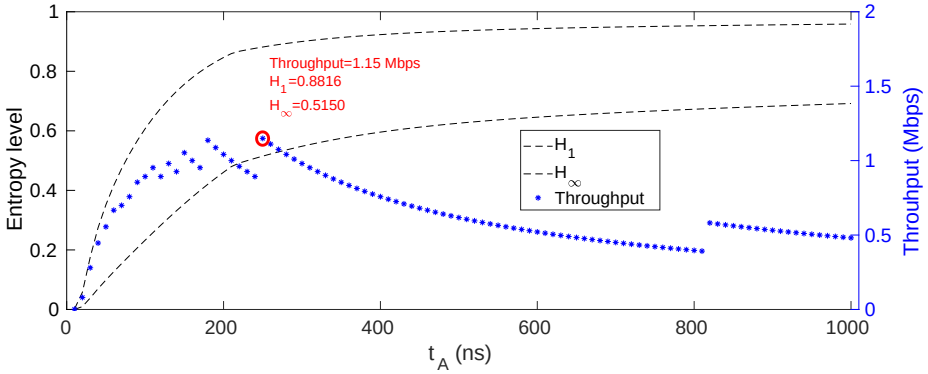


Figure 12: The lower bound estimation for H_1 and H_∞, the expected throughput after post-processing for different accumulation time.

6.3 Intel FPGA implementations

In order to show portability of our design to different FPGA platforms, we have also implemented ES-TRNG on Intel Cyclone V FPGA. The basic building blocks of this FPGA are adaptive logic modules (ALMs) that contain 8-input flexible LUTs, 2 fast adders and 2 flip-flops.

The tapped delay line is implemented using fast carry chains in dedicated adders. Since 1 ALM already contains two carry stages, we would need only 2 ALMs to implement a structure equivalent to CARRY4 in Xilinx FPGA. However, due to considerable differences of the carry stages' propagation delays, we opted to use output of every second carry stage. In this way, we obtained more balanced delays of the resulting delay chain stages at the price of the increased resource utilization - 4 ALMs instead of 2. To be able to accurately determine RO1 frequency, we implemented $RO1$ with 2 LUTs, while for $RO2$ we used 3 LUTs. The oscillation periods of $RO1$ and $RO2$ were determined with ripple counters and their values are $T_{01} = 1745.68\ ps$ and $T_{02} = 3020.068\ ps$ respectively. Bit extractor module is implemented by using a single ALM, thus achieving a compact implementation.

Table 3: The comparison with existing TRNG implementations

TRNG types	Stoch. models	Xilinx FPGAs				Intel FPGAs			
		Families	Area	Bit rate [Mbits/s]	Design effort	Families	Area	Bit rate [Mbits/s]	Design effort
Open-loop [BRGD13] [DGH07] [DGH09]	Y	Virtex5	n/a LUTs 64 Latches	20	MP MR	Stratix EP1S25	140 Stratix cells (∼14 LABs)	20	MP MR
FIGARO [Gol06] [SDH17]	N	Virtex2	n/a	n/a	n/a	CycloneV	n/a	n/a	MP noMR
BRAM [GP09]	N	Spartan3	n/a	100	MP noMR	-	-	-	-
CMAFC [MKD11]	N	Virtex5	128 LUTs	2	MP MR	-	-	-	-
DCTRNG [RYDV15] [GYRV18]	Y	Spartan6	67 slices	14.3	MP noMR	CycloneV	230 ALMs (23 LABs)	11.1	MP MR
			40 slices	1.53					
MERO [VHKK08] [SSR09]	Y	Virtex2	n/a	50	n/a	StratixII	9 ALUTs	2.5	n/a
ERO [PMB+16]	Y	Spartan6	46 LUTs 19 FFs	0.0042	MP MR	CycloneV	34 LUTs 20 FFs	0.0027	MP MR
COSO [PMB+16]	Y	Spartan6	18 LUTs 3 FFs	0.54	MP MR	CycloneV	13 LUTs 3 FFs	1.44	MP MR
MURO [PMB+16] [SPV06] [SMS07]	Y	Spartan6	521 LUTs 131 FFs	2.57	noMP noMR	CycloneV	525 LUTs 130 FFs	2.2	noMP noMR
PLL [PMB+16]	Y	Spartan6	34 LUTs 14 FFs	0.44	noMP noMR	CycloneV	24 LUTs 14 FFs	0.6	noMP noMR
TERO [PMB+16] [VD10]	Y	Spartan6	39 LUTs 12 FFs	0.625	MP MR	CycloneV	46 LUTs 12 FFs	1	MP MR
STR [PMB+16] [CFFA13]	Y	Spartan6	346 LUTs 256 FFs	154	MP MR	CycloneV	352 LUTs 256 FFs	245	MP MR
This work	**Y**	**Spartan6**	**10 LUTs+5FFs + (6 LUTs+6FFs) t_A counter**	**1.15**	**MP noMR**	**CycloneV**	**10 LUTs+6FFs + (6 LUTs+6FFs) t_A counter**	**1.067**	**MP noMR**

The measurement of Intel Cyclone V FPGA platform parameters is performed in the same way as for Xilinx Spartan-6 in Section 5. The propagation delays are calculated as follows: $t_{r,1} = 67.316\ ps$, $t_{r,2} = 68.316\ ps$, $t_{f,1} = 52.044\ ps$ and $t_{f,2} = 50.544\ ps$. The white-noise jitter strength is $\sigma_m^2/t_m = 0.020\ ps$, while the duty cycle D of $RO1$ is 0.58. After measuring the physical parameters, we determined the design parameters, as previously described. The jitter accumulates for $t_A = 230\ ns$, while the parity filter has to be of order $n_f = 3$ to obtain Shannon entropy level of 0.997. The obtained throughput of ES-TRNG on Intel Cyclone V FPGA is 1.067 $Mbps$. We observe that although the white-noise jitter strength on Intel Cyclone V is higher than on Xilinx Spartan 6, the propagation delays on Xilinx Spartan 6 are substantially lower, leading to similar throughputs on both FPGAs.

7 Results and Comparison

Table 3 shows the comparison to the state-of-the-art TRNG designs for both Xilinx and Intel FPGAs. The second column of the table indicates the availability of stochastic models for each TRNG design. The utilization of hardware resources are reported in the third column. Only the basic units in the FPGA are reported, such as LUTs, Flip-flops and Slices. Special dedicated primitives, like PLL building blocks for PLL-TRNG, are not included here. Among Xilinx TRNG designs listed in the table, ES-TRNG achieves the smallest hardware footprint. Our ES-TRNG generates more than 1 $Mbps$ internal random numbers with an estimated minimal Shannon entropy level of 0.997. Here we discussed two types of design effort for FPGA implementation: manual placement (MP) and manual routing (MR). MP is critical for some TRNG designs, because the quality of those TRNGs is sensitive to the relative spatial location of their building blocks. For TRNG designs based on identical delays or balanced routing, MR cannot be avoided.

As can be seen in Table 3, the ES-TRNG has one of the smallest area consumption compared to TRNGs implemented on Cyclone V FPGAs. The only TRNG designs with comparable implementation footprints are the COSO-TRNG (the coherent sampling ring

oscillator based TRNG) and the PLL-TRNG (the coherent sampling based TRNG using PLLs) [PMB+16]. It is worthwhile to note that although the COSO-TRNG has higher throughput than ES-TRNG, it requires laborious manual placement and routing, which has to be performed for every target device. On the other hand, the PLL-TRNG does not require any manual placement or routing on Cyclone V FPGAs, but it occupies dedicated PLL modules and provides a 40% lower throughput compared to ES-TRNG.

Here we would like to point out a problem with results comparison in this research domain that became relevant in recent years. Many TRNG designs were developed before the publication of evaluation standards [TBK+18] and [KS11]. Therefore, these old designs and some of the recent ones are not provided with stochastic models that are required today. Without the proper security analysis or by using debunked or simplified assumptions, it is possible to achieve a better hardware performance than if AIS-31 criteria are strictly followed. For example, a designer may base the entropy estimation solely on the results of the statistical tests. If this overestimation is used to guide the choice of design parameters, higher targets for throughput, area and energy are more easily achieved. This has the unfortunate effect that designs with a more rigorous security analysis appear worse in terms of hardware performance when compared to their predecessors.

In particular, in ring oscillator based designs, the jitter strength is often the critical parameter that significantly affects the performance of the final hardware implementation. Some recent works on jitter measurement on FPGAs [FL14, LB15, YRG+17] showed that the strength of the Gaussian noise is lower by an order of magnitude compared to the values that were used in older TRNG designs. For comparison, we carried out the ES-TRNG design procedure assuming that the jitter strength is five time higher than measured (this is similar to the jitter strength reported in [SPV06]). The obtained design parameters were $t_A = 30\ ns$ and $n_f = 2$ which results in the throughput of 6.25 $Mbps$.

8 Conclusion

In this work, we present a novel true random number generator called ES-TRNG. Two techniques, namely variable-precision phase encoding and repetitive sampling, are used to increase the throughput and the entropy of the proposed generator and to reduce the hardware footprint. The digital noise source is implemented using only 10 look-up tables (LUTs) and 5 flip-flops (FFs) of a Xilinx Spartan-6 FPGA, and achieves a throughput of 1.15 Mb/s with 0.997 bits of Shannon entropy. On Intel Cyclone V FPGAs, this implementation uses 10 LUTs and 6 FFs, and achieves a throughput of 1.07 $Mbps$. The proposed generator is backed up with a security analysis based on the stochastic model of the entropy source.

Acknowledgments

We would like to thank the editors and the anonymous reviewers of TCHES for their constructive comments. This work was supported in part by the Research Council KU Leuven: C16/15/058. In addition, this work is supported in part by the Hercules Foundation AKUL/11/19, and through the Horizon 2020 research and innovation programme under grant agreement No 644052 HECTOR and Cathedral ERC Advanced Grant 695305. Bohan Yang is supported in part by the Scholarship from China Scholarship Council (No.201206210295).

References

[BCC+13] Daniel J. Bernstein, Yun-An Chang, Chen-Mou Cheng, Li-Ping Chou, Nadia Heninger, Tanja Lange, and Nicko van Someren. Factoring RSA keys from certified smart cards: Coppersmith in the wild. In *Advances in Cryptology - ASIACRYPT*, pages 341–360, 2013.

[BL05] Marco Bucci and Raimondo Luzzi. Design of Testable Random Bit Generators. In Josyula R. Rao and Berk Sunar, editors, *CHES 2005*, pages 147–156, 2005.

[BPPT06] Ralf Brederlow, Ramesh Prakash, Christian Paulus, and Roland Thewes. A Low-power True Random Number Generator using Random Telegraph Noise of Single Oxide-Traps. In *IEEE International Solid State Circuits Conference - Digest of Technical Papers*, pages 1666–1675, Feb 2006.

[BRGD13] Molka Ben-Romdhane, Tarik Graba, and Jean-Luc Danger. Stochastic model of a metastability-based true random number generator. In *TRUST*, pages 92–105. Springer Berlin Heidelberg, 2013.

[CFFA13] Abdelkarim Cherkaoui, Viktor Fischer, Laurent Fesquet, and Alain Aubert. A Very High Speed True Random Number Generator with Entropy Assessment. In *CHES*, pages 179–196, 2013.

[DGH07] Jean-Luc Danger, Sylvain Guilley, and Philippe Hoogvorst. Fast true random generator in fpgas. In *2007 IEEE Northeast Workshop on Circuits and Systems*, pages 506–509, Aug 2007.

[DGH09] Jean-Luc Danger, Sylvain Guilley, and Philippe Hoogvorst. High speed true random number generator based on open loop structures in fpgas. *Microelectronics Journal*, 40(11):1650–1656, 2009.

[FIP94] PUB FIPS. 140-1. *Security Requirements for Cryptographic Modules*, 11, 1994.

[FL14] Viktor Fischer and David Lubicz. Embedded Evaluation of Randomness in Oscillator Based Elementary TRNG. In *CHES*, pages 527–543, 2014.

[Gol06] Jovan Dj. Golić. New Methods for Digital Generation and Postprocessing of Random Data. *IEEE Trans. Computers*, 55(10):1217–1229, 2006.

[GP09] Tim Güneysu and Christof Paar. Transforming Write Collisions in Block RAMs into Security Applications. In *Proceedings of the 2009 International Conference on Field-Programmable Technology, FPT 2009, Sydney, Australia, December 9-11, 2009*, pages 128–134, 2009.

[GYRV18] Miloš Grujić, Bohan Yang, Vladimir Rožić, and Ingrid Verbauwhede. Towards Inter-Vendor Compatibility of True Random Number Generators for FPGAs. In *DATE*, pages 1–4, 2018.

[HFBN15] Patrick Haddad, Viktor Fischer, Florent Bernard, and Jean Nicolai. A Physical Approach for Stochastic Modeling of TERO-Based TRNG. In *CHES*, pages 357–372, 2015.

[KS11] Wolfgang Killmann and Werner Schindler. A Proposal for: Functionality classes for random number generators . BSI, Bonn, 2011.

[LB15] David Lubicz and Nathalie Bochard. Towards an oscillator based TRNG with a certified entropy rate. *IEEE Trans. Computers*, 64(4):1191–1200, 2015.

[Mar98] Georges Marsaglia. DIEHARD Test suite. *Online: http://www. stat. fsu. edu/pub/diehard/. Last visited*, 8(01):2014, 1998.

[Mar03] Dichtl Markus. How to Predict the Output of a Hardware Random Number Generator. In *CHES*, pages 181–188, 2003.

[MJS+16] Sanu K. Mathew, David Johnston, Sudhir Satpathy, Vikram Suresh, Paul Newman, Mark A. Anders, Himanshu Kaul, Amit Agarwal, Steven Hsu, Gregory K. Chen, and Ram K. Krishnamurthy. μrng: A 300-950 mv, 323 gbps/w all-digital full-entropy true random number generator in 14 nm finfet CMOS. *J. Solid-State Circuits*, 51(7):1695–1704, 2016.

[MKD11] Mehrdad Majzoobi, Farinaz Koushanfar, and Srinivas Devadas. FPGA-Based True Random Number Generation Using Circuit Metastability with Adaptive Feedback Control. In *CHES*, pages 17–32, 2011.

[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 *FPL*, pages 1–10, Aug 2016.

[PSR06] Fabio Pareschi, Gianluca Setti, and Riccardo Rovatti. A Fast Chaos-based True Random Number Generator for Cryptographic Applications. In *Proceedings of the 32nd European Solid-State Circuits Conference*, pages 130–133, Sept 2006.

[RSN+10] Andrew Rukhin, Juan Soto, James Nechvatal, Miles Smid, Elaine Barker, Stefan Leigh, Mark Levenson, Mark Vangel, David Banks, Alan Heckert, James Dray, and San Vo. Nist special publication 800-22 revision 1a. National Institute of Standards and Technology, 2010.

[RYDV15] Vladimir Rožić, Bohan Yang, Wim Dehaene, and Ingrid Verbauwhede. Highly efficient entropy extraction for true random number generators on FPGAs. In *Proceedings of the 52nd Annual Design Automation Conference, San Francisco, CA, USA, June 7-11, 2015*, pages 116:1–116:6, 2015.

[SDH17] Martin Schramm, Reiner Dojen, and Michael Heigl. Experimental assessment of firo- and garo-based noise sources for digital trng designs on fpgas. In *2017 International Conference on Applied Electronics (AE)*, pages 1–6, Sept 2017.

[SMS07] Berk Sunar, William J. Martin, and Douglas R. Stinson. A Provably Secure True Random Number Generator with Built-In Tolerance to Active Attacks. *IEEE Transactions on Computers*, pages 109–119, Jan 2007.

[SPV06] Dries Schellekens, Bart Preneel, and Ingrid Verbauwhede. FPGA Vendor Agnostic True Random Number Generator. In *FPL*, pages 1–6, 2006.

[SSR09] Renaud Santoro, Olivier Sentieys, and Sébastien Roy. On-the-fly evaluation of fpga-based true random number generator. In *2009 IEEE Computer Society Annual Symposium on VLSI*, pages 55–60, May 2009.

[TBK+18] Meltem Sönmez Turan, Elaine Barker, John Kelsey, Kerry McKay, Mary Baish, and Michael Boyle. Recommendation for the Entropy Sources Used for Random BitGeneration. NIST Special Publication 800-90B, 2018.

[VD10] Michal Varchola and Miloš Drutarovský. New high entropy element for fpga based true random number generators. In *CHES*, volume 6225 of *LNCS*, pages 351–365. 2010.

[VHKK08] Ihor Vasyltsov, Eduard Hambardzumyan, Young-Sik Kim, and Bohdan Karpinskyy. Fast Digital TRNG Based on Metastable Ring Oscillator. In *Cryptographic Hardware and Embedded Systems - CHES 2008, 10th International Workshop, Washington, D.C., USA, August 10-13, 2008. Proceedings*, pages 164–180, 2008.

[WT08] Knut Wold and Chik How Tan. Analysis and Enhancement of Random Number Generator in FPGA Based on Oscillator Rings. In *ReConFig'08: 2008 International Conference on Reconfigurable Computing and FPGAs, 3-5 December 2008, Cancun, Mexico, Proceedings*, pages 385–390, 2008.

[YRG+17] Bohan Yang, Vladimir Rožić, Miloš Grujić, Nele Mentens, and Ingrid Verbauwhede. On-chip Jitter Measurement for True Random Number Generators. In *AsianHOST*, pages 1–6, October 2017.

[YRM+16] Bohan Yang, Vladimir Rožić, Nele Mentens, Wim Dehaene, and Ingrid Verbauwhede. TOTAL: TRNG on-the-fly testing for attack detection using Lightweight hardware. In *DATE*, pages 127–132, 2016.

A The proofs in the stochastic model

A.1 The proof of the claim in Equation (30)

From Equation (28), we have:

$$X_1 = X_0 + \frac{T_{02}}{T_{01}} + Z_1, \tag{42}$$

where T_{02}/T_{01} is a constant. The random variables X_0 and Z_1 are mutually independent.

As the first step, from the definition of $g(x)$, we can derive the conditional probability of X_0 given EV_0:

$$\begin{aligned}
\rho_{X_0|EV_0}(x) &= \frac{d}{d\alpha}[Pr(X_0 \le \alpha | EV_0)] \\
&= \frac{1}{Pr(EV_0)} \cdot \frac{d}{d\alpha}[Pr(X_0 \le \alpha, EV_0)] \\
&= \frac{1}{Pr(EV_0)} \cdot \frac{d}{d\alpha}[Pr(X_0 \le \alpha, X_0 \in S_N)] \\
&= \frac{1}{Pr(EV_0)} \cdot \frac{d}{d\alpha}\left[\int_{(-\infty,\alpha]\cap S_N} \rho_{X_0}(x)dx\right] \\
\text{(by Equation (20))} &= \frac{1}{Pr(EV_0)} \cdot \frac{d}{d\alpha}\left[\int_{-\infty}^{\alpha} \rho_{X_0}(x) \cdot g(x)dx\right] \\
&= \frac{1}{Pr(EV_0)} \cdot \rho_{X_0}(x) \cdot g(x)\,.
\end{aligned} \tag{43}$$

The Z_1 is independent of X_0, thus we have:

$$\rho_{Z_1|EV_0}(x) = \rho_{Z_1}(x). \tag{44}$$

We derive the probability density of $X_0 + Z_1$, given the condition EV_0:

$$\begin{aligned}
\rho_{(X_0+Z_1)|EV_0}(x) &= \rho_{Z_1|EV_0} * \rho_{X_0|EV_0})(x) \\
\text{(by Equation (44))} &= (\rho_{Z_1} * \rho_{X_0|EV_0})(x) \\
\text{(by convolution)} &= \int_{-\infty}^{\infty} \rho_{Z_1}(x-\mu) \cdot \rho_{X_0|EV_0}(\mu)d\mu \\
\text{(by Equation (43))} &= \frac{1}{Pr(EV_0)} \cdot \int_{-\infty}^{\infty} \rho_{Z_1}(x-\mu) \cdot \rho_{X_0}(\mu) \cdot g(\mu)d\mu \\
\text{(by Equation (29))} &= \frac{1}{Pr(EV_0)} \cdot \int_{-\infty}^{\infty} f_{0,\sigma_{T_{02}}}(x-\mu) \cdot \rho_{X_0}(\mu) \cdot g(\mu)d\mu.
\end{aligned} \tag{45}$$

As the final step, we derive the Equation (30) as:

$$\begin{aligned}
\rho_{X_1|EV_0}(x) &= \rho_{(X_0+Z_1+\frac{T_{02}}{T_{01}})|EV_0}(x) = \rho_{(X_0+Z_1)|EV_0}\left(x - \frac{T_{02}}{T_{01}}\right) \\
&= \frac{1}{Pr(EV_0)} \cdot \int_{-\infty}^{\infty} f_{0,\sigma_{T_{02}}}\left(x - \frac{T_{02}}{T_{01}} - \mu\right) \cdot \rho_{X_0}(\mu) \cdot g(\mu) d\mu \\
\text{(by Equation (45))} &= \frac{1}{Pr(EV_0)} \cdot \int_{-\infty}^{\infty} \rho_{Z_1}(x - (\mu + \frac{T_{02}}{T_{01}})) \cdot \rho_{X_0}(\mu) \cdot g(\mu) d\mu \\
&= \frac{1}{Pr(EV_0)} \cdot \int_{-\infty}^{\infty} f_{0,\sigma_{T_{02}}}(x - \mu) \cdot \rho_{X_0}\left(\mu - \frac{T_{02}}{T_{01}}\right) \cdot g\left(\mu - \frac{T_{02}}{T_{01}}\right) d\mu.
\end{aligned} \tag{46}$$

A.2 The proof of the claim in Equation (31)

The derivation of Equation (31) is similar to Section A.1.

From Equation (28), we have:

$$X_i = X_{i-1} + \frac{T_{02}}{T_{01}} + Z_i, \tag{47}$$

where T_{02}/T_{01} is a constant. The random variables X_{i-1} and Z_i are mutually independent.

As the first step, we derive the $\rho_{X_{i-1}|EV_{i-1}}(x)$ as:

$$\begin{aligned}
\rho_{X_{i-1}|EV_{i-1}}(x) &= \frac{d}{d\alpha}[Pr(X_{i-1} \leq \alpha | EV_{i-1})] \\
&= \frac{1}{Pr(EV_{i-1})} \cdot \frac{d}{d\alpha}[Pr(X_{i-1} \leq \alpha, EV_{i-1})] \\
&= \frac{1}{Pr(EV_{i-1})} \cdot \frac{d}{d\alpha}[Pr(X_{i-1} \leq \alpha, Y_{i-1} = N, EV_{i-2})] \\
&= \frac{Pr(EV_{i-2})}{Pr(EV_{i-1})} \cdot \frac{d}{d\alpha}[Pr(X_{i-1} \leq \alpha, Y_{i-1} = N | EV_{i-2})] \\
&= \frac{Pr(EV_{i-2})}{Pr(EV_{i-1})} \cdot \frac{d}{d\alpha}[Pr(X_{i-1} \leq \alpha, X_{i-1} \in S_N | EV_{i-2})] \\
&= \frac{Pr(EV_{i-2})}{Pr(EV_{i-1})} \cdot \frac{d}{d\alpha}\left[\int_{(-\infty,\alpha] \cap S_N} \rho_{X_{i-1}|EV_{i-2}}(x) dx\right] \\
\text{(by Equation (20))} &= \frac{Pr(EV_{i-2})}{Pr(EV_{i-1})} \cdot \frac{d}{d\alpha}\left[\int_{-\infty}^{\alpha} \rho_{X_{i-1}|EV_{i-2}}(x) \cdot g(x) dx\right] \\
&= \frac{Pr(EV_{i-2})}{Pr(EV_{i-1})} \cdot \rho_{X_{i-1}|EV_{i-2}}(x) \cdot g(x).
\end{aligned} \tag{48}$$

The Z_i is independent of EV_{i-1}, thus we have:

$$\rho_{Z_i|EV_{i-1}}(x) = \rho_{Z_i}(x). \tag{49}$$

We derive the probability density of $X_{i-1} + Z_i$, given the condition EV_{i-1}:

$$\begin{aligned}
\rho_{(X_{i-1}+Z_i)|EV_{i-1}}(x) &= (\rho_{Z_i|EV_{i-1}} * \rho_{X_{i-1}|EV_{i-1}})(x) \\
\text{(by Equation (49))} &= (\rho_{Z_i} * \rho_{X_{i-1}|EV_{i-1}})(x) \\
\text{(by convolution)} &= \int_{-\infty}^{\infty} \rho_{Z_i}(x-\mu) \cdot \rho_{X_{i-1}|EV_{i-1}}(\mu) d\mu \\
\text{(by Equation (46))} &= \frac{Pr(EV_{i-2})}{Pr(EV_{i-1})} \cdot \int_{-\infty}^{\infty} \rho_{Z_i}(x-\mu) \cdot \rho_{X_{i-1}|EV_{i-2}}(\mu) \cdot g(\mu) d\mu \\
\text{(by Equation (29))} &= \frac{Pr(EV_{i-2})}{Pr(EV_{i-1})} \cdot \int_{-\infty}^{\infty} f_{0,\sigma_{T_{02}}}(x-\mu) \cdot \rho_{X_{i-1}|EV_{i-2}}(\mu) \cdot g(\mu) d\mu. \qquad (50)
\end{aligned}$$

As the final step, we derive the Equation (31) as:

$$\begin{aligned}
\rho_{X_i|EV_{i-1}}(x) &= \rho_{(X_{i-1}+Z_i+\frac{T_{02}}{T_{01}})|EV_{i-1}}(x) = \rho_{(X_{i-1}+Z_i)|EV_{i-1}}\left(x - \frac{T_{02}}{T_{01}}\right) \\
\text{(by Equation (50))} &= \frac{Pr(EV_{i-2})}{Pr(EV_{i-1})} \cdot \int_{-\infty}^{\infty} f_{0,\sigma_{T_{02}}}\left(x - \frac{T_{02}}{T_{01}} - \mu\right) \cdot \rho_{X_{i-1}|EV_{i-2}}(\mu) \cdot g(\mu) d\mu \\
&= \frac{Pr(EV_{i-2})}{Pr(EV_{i-1})} \cdot \int_{-\infty}^{\infty} f_{0,\sigma_{T_{02}}}(x-\mu) \cdot \rho_{X_{i-1}|EV_{i-2}}\left(\mu - \frac{T_{02}}{T_{01}}\right) \cdot g\left(\mu - \frac{T_{02}}{T_{01}}\right) d\mu. \qquad (51)
\end{aligned}$$