Stochastic Modelling of Counting Ring Oscillator TRNGs

Thomas Pouwels

Stochastic Modelling of Counting Ring Oscillator TRNGs

by

Thomas Pouwels

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on July 16, 2024, at 2:00 PM.

Student number:	4705971		
Project duration:	November, 2023	–	July, 2024
Thesis committee:	Dr. M. Taouil,		TU Delft, supervisor
	Dr. S. Vollebregt,		TU Delft
	Ir. E. Hagenaars,		Ministry of the Interior and Kingdom Relations

TU Delft

Abstract

True Random Number Generators (TRNG) are fundamental cryptographic primitives. They are needed, for example, to generate cryptographic keys and numbers. Poor random number generation can lead to weaknesses in the cryptographic system. A good example can be found in the work of Heninger et al., who managed to retrieve large amounts of private RSA keys due to entropy problems in random number generators used for key generation.

Statistical test suites such as Dieharder and NIST SP 800-22 can be used to determine whether random number generators provide the required entropy. To improve such an evaluation, which is necessary for high-assurance applications, a stochastic model of the noise source must be developed as described, for example, in AIS 20/31.

The research presented in this thesis focuses on designing and validating a noise source based on counting ring oscillator periods. The work presents an analysis of this ring oscillator-based design, consisting of stochastic models, entropy evaluations, high-level and SPICE simulations, and hardware validation using the Intel Arria 10 FPGA platform.

Using the developed models, this work (i) quantifies the improvement in randomness due to sampling the noise source with a second ring oscillator, (ii) establishes a lower bound for the min-entropy of the noise source, and (iii) shows that the design benefits from using ring oscillators with shorter periods and, thus, less area. These findings are verified in high-level and SPICE simulations.

In the end, despite the appealing theoretical benefits of the proposed TRNG noise source, the hardware validation revealed unmodeled noise components, leading to inconclusive results. Consequently, no TRNG implementation based on the designed noise source was made. This highlights the challenges of translating theoretical advantages into successful hardware implementations.

Contents

Acronyms

AES	Advanced Encryption Standard
ASIC	application-specific integrated circuit
BSI	Bundesamt für Sicherheit in der Informationstechnik
DRNG	deterministic random number generator
EMFI	electromagnetic fault injection
EMI	electromagnetic Interference
FFT	fast Fourier transform
FPGA	field-programmable gate array
LSB	least significant bit
NIST	National Institute of Standards and Technology
PCIe	peripheral component interconnect express
PLL	phase-locked loop
PTRNG	physical true random number generator
RNG	random number generator
RO	ring oscillator
STR	self-timed ring
TERO	transient effect ring oscillator
TRNG	true random number generator

1 Introduction

This section provides a brief overview of the subjects covered in this thesis, emphasising their significance and highlighting the main contributions. Section 1.1 presents the motivation for this study and the relevance of proper random number generation. Section 1.2 gives a quick overview of the subject in literature, including a review of the current state-of-the-art. Section 1.3 presents the academic contributions of this thesis. In Section 1.4, the thesis outline is introduced.

1.1. Motivation

Kerckhoff's principle of cryptography states that a cryptosystem should be secure even if everything about it, except the key, is public knowledge [1]. An example of a cryptosystem that did not follow this principle is CRYPTO1, used in the Mifare Classic by NXP [2]. Although kept secret for a long time, the protocol was eventually reverse-engineered, which showed weaknesses in the pseudo-random number generator and protocol that allowed adversaries '*to find any key in a matter of seconds*' [2]. Note that at the time, the Mifare Classic was the most widely used contactless card in the market [3].

Conversely, a well-known example of a cryptosystem that does follow Kerckhoff's principle is Advanced Encryption Standard (AES) [4]. Disregarding implementation-based attacks such as side channels or fault injection, the security of AES depends entirely on the security of the used encryption key. It is why bodies such as the National Institute of Standards and Technology (NIST) recommend keys are based directly or indirectly on the output of an approved random number generator (RNG) [5].

In like manner, random numbers also appear in other parts of cryptographic systems, such as message randomisation, nonces, identification, and more [6]. Message randomisation, for example, is applied to make the encryption process non-deterministic. This is necessary because deterministic encryption gives an adversary valuable information. If the adversary sees the same ciphertext twice, they know the same message was transmitted twice.

A nonce (short for 'number used once') is another way of introducing randomness into cryptographic systems. They are, for example, used in the CTR mode of operation of AES [7]. A ciphertext is obtained by encrypting the nonce with a key and then XORing this cypher with the plaintext. When more cyphers are needed, the nonce is incremented and then encrypted again. This way, encrypting the same message with the same key again will still give a different ciphertext.

All of the techniques discussed so far assume that the randomness used is, in fact, genuinely random. If this were not the case, weaknesses would be introduced into the basis of the cryptographic primitives. This is not just a theoretical problem. In [8], Heninger et al. performed a large-scale network survey and were able to retrieve 172,000 RSA private keys due to entropy problems in the used random number generators. The paper shows that the necessary computations are trivially implemented and executed in several hours on a simple CPU.

Implementing proper random number generators is thus of paramount importance for the correct functioning of cryptographic systems. It is a task not to be taken lightly.

1.2. State of the Art

The generation of random numbers is a problem that has been studied thoroughly, also before the need for cryptographic random numbers arose. Already in 1890, Nature published an article by statistician Francis Galton where he wrote that ”as an instrument for selecting at random, I have found nothing superior to dice” [9].

Nonetheless, when the need for large amounts of random numbers arose, a more efficient solution than rolling dice was required. This is why, in 1955, the RAND Corporation published ”A Million Random Digits with 100,000 Normal Deviates” [10]. As the title suggests, it contained a million random numbers generated by an electronic roulette wheel. It was used extensively, for example, in Monte Carlo methods [10].

Major improvements in the availability of random numbers were made when, in 1999, Intel released the Intel 810 chipset [11], with an integrated random number generator. It ’supplies applications and security middleware products with true non-deterministic random numbers’ [11] by exploiting thermal noise across a resistor. Finally, proper random numbers were available locally to the machine.

With the emergence of the Internet of Things, more and more devices are becoming reliant on secure communication and, thus, random number generators. This across a wide range of technology nodes, such as CMOS and FinFET, and device types, such as CPUs, microcontrollers, field-programmable gate arrays (FPGAs) and more. Not all of these technologies allow for the same kinds of random number generators, and random number generators that can be implemented on different technologies might still give differing results.

To compare these different random number generators, common statistical test suites have been developed. The test suites most used in literature include the ones provided by NIST [12] and the Bundesamt für Sicherheit in der Informationstechnik (BSI) [13], and Dieharder [14]. These test suites allow developers to check for irregularities or patterns that would indicate their design might not be random (enough).

Nevertheless, passing such tests does not mean a design is truly random. A finite number of statistical tests can, by definition, only test a finite number of patterns [13].

Recently, BSI published a (draft) revision of their mathematical-technical reference for the evaluation of random number generators [13], often referred to as AIS 20/31. Regarding the evaluation of physical true random number generators, AIS 20/31 requires the developer to establish assurance by means of a stochastic model of the random numbers [13] for all functional classes. This stochastic model should aim to describe the ‘randomness’ of the output by relating it to the physical randomness on which the true random number generator is based.

In [15] and [16], it is shown that over-estimation of the randomness is an easy mistake to make as external perturbations influence the noise source. It is thus extremely important to first develop and validate the stochastic model before statistically testing the output bits of the noise source. Only then will passing the test suite provide the necessary assurance that the output randomness indeed originates from the intended source of randomness.

As of today, numerous designs for hardware random number generators exist, and even more variations of these designs have been developed for use in different applications and devices. However, many of these designs have the fundamental shortcoming that there is no clear stochastic model relating the internal, physical randomness to the output; instead, they rely only on statistical tests to prove the randomness of the design.

For these random number generator designs, there thus still exists a clear gap between the current state of the art and the required theoretical rationale that provides the desired assurance. This work aims to help close this gap by providing a stochastic model for a noise source that, to the best of our knowledge, currently has none.

1.3. Contributions

In this thesis, a noise source for a true random number generator is stochastically modelled, simulated and tested. The design of the noise source is inspired by the work of Noumon Allini et al. [17]. Their work, however, only included empirical validation of the entropy of the noise source. As discussed in the state of the art, it is imperative that a stochastic model of the source is developed and tested to fully understand the randomness in a noise source. This is where the contributions of this thesis come in. They are as follows:

- **This thesis presents a novel approach to modelling a noise source** for a true random number generator that obtains its randomness from counting the rising edges of a ring oscillator. Two stochastic models are presented, one under the assumption of an ideal sampling time and one where this assumption is relaxed to a normally distributed sampling time. The improvement in randomness due to a noisy sample clock is then quantified by comparing the models.
- **This thesis establishes a lower bound for the min-entropy of the noise source** and relates it to physical device parameters and sampling time. It is shown that not only the least significant bit but also other bits in the counter contain some entropy when sampled, improving the efficiency of the noise source by 60%.
- **This thesis shows that using ring oscillators with smaller periods allows for shorter sampling times**. The period of a ring oscillator is loosely proportional to the number of inverters it consists of. This means that the noise source has the huge practical benefit that using less area gives better performance.

1.4. Thesis Outline

This report is organised in several chapters. First, relevant background information on random number generation is discussed in Chapter 2. Then, Chapter 3 gives an overview of the noise source, presents the stochastic model, and provides entropy estimations. This is followed by Chapter 4, where software simulations are conducted to verify the stochastic model. Chapter 5 tests hardware implementations of the noise source to validate the model and provides a discussion on the results obtained in the tests, relating them to the stochastic model and simulation results. Finally, Chapter 6 provides a summary of the results and potential future research subjects.

2 Random Numbers

In this chapter, an introduction to hardware random number generation is presented. Section 2.1 starts with an overview of the topic and relevant standards. Section 2.2 then introduces different ways of generating random numbers in hardware. An overview of the security aspects of random number generators based on ring oscillators is presented in Section 2.3.

2.1. Random Number Generation

The generation of good random numbers is a crucial aspect of the implementation of cryptographic systems. They are used in encryption schemes, key-generation processes, authentication protocols, and other security-relevant functions. The quality of the generated random numbers directly affects the security of the cryptographic system.

In general, two main classes of RNGs can be specified. Deterministic random number generators (DRNGs), also known as pseudorandom number generators, take random inputs (seeds) and extend these using deterministic algorithms to potentially very long output sequences. True random number generators (TRNGs), on the other hand, produce random numbers based on some nondeterministic noise source. These can be further divided into two subclasses: physical true random number generators (PTRNGs), which use a physical source, for example a noisy diode [18], and Non-Physical TRNGs, such as `/dev/random` in some versions of the Linux kernel [19], which gain their entropy from non-physical sources.

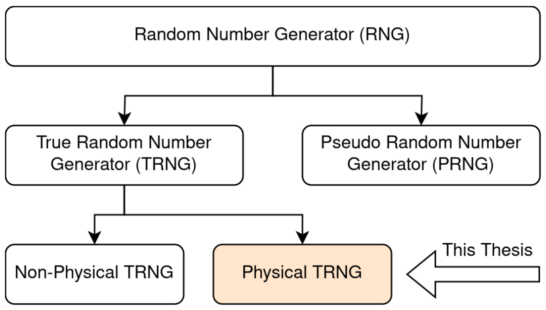
Random Number Generator (RNG)
True Random Number Generator (TRNG)
Pseudo Random Number Generator (PRNG)
Non-Physical TRNG
Physical TRNG
This Thesis

Figure 2.1: Classification of RNGs.

Figure 2.1 gives a graphical overview of the RNG classifications. The term TRNG is often used to indicate a physical true random number generator, a convention that will also be used in this thesis.

To evaluate a TRNG, the minimal entropy of the noise source must be determined. One way of doing this is using statistical test suites such as the ones described in NIST SP 800-22 [12] and AIS 20/31 [13]. However, this is no guarantee the entropy claim is correct, as a finite number of tests can only check for a finite number of output patterns [12, 13].

To improve on the evaluation, a stochastic model of the noise source can be used [13, 20]. The goal is to mathematically describe the physical phenomena contributing to the noise with the help of random variables. This way, a relation is created between the unpredictable analogue phenomena inside the device and the digitised output of the random number generator. After carefully estimating the model's physical parameters, a lower bound for the entropy can be established. Then, and only then, can the test suites be used to verify the entropy claims.

2.2. Noise Sources

In [21], Cicek et al. split the TRNG into three main components: the entropy source, the entropy harvester and the post-processor. The entropy source is a (physical) effect which has some amount of entropy in itself. The harvester then generates random bits based on this entropy. The post-processor is often included to remove potential bias, but can be skipped altogether.

Figure 2.2 shows the different categories of entropy sources, their generation method and possible technologies to implement the source.

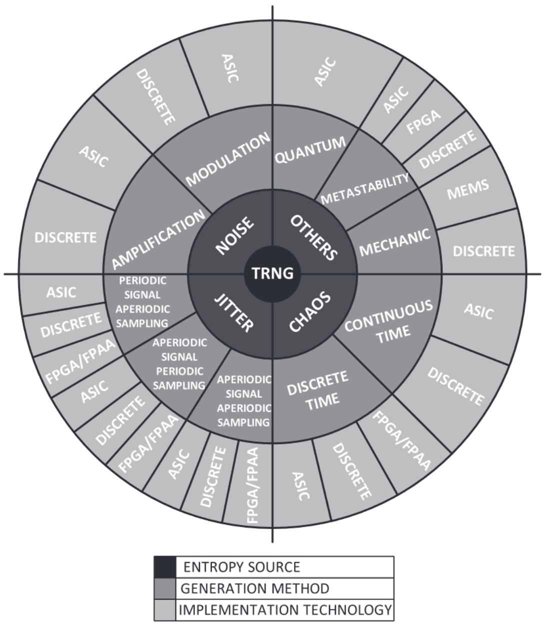

Figure 2.2: Cicek's classification of entropy sources [21]

As an example, in [22], Wang et al. generate random bits from thermal noise, the entropy source, by amplifying it with a single CMOS inverter, the entropy harvester. Afterwards, a 4-bit Von Neumann [23] circuit, the post-processor, is used to remove bias from the output.

2.2.1. Entropy Sources in FPGAs

The TRNG proposed in this thesis will be implemented and tested on a FPGA. FPGAs are hardware devices that can be reconfigured. Advantages are reduced development time, increased cost-effectiveness in low-volume production compared to application-specific integrated circuits (ASICs), and the flexibility to update already fielded devices.

As per Cicek's classification, Figure 2.2, entropy sources in FPGAs are generally based on metastability, chaos or jitter. TRNGs exploiting metastability can, for example, be constructed using Set-Reset latches, as discussed in [24] and [25]. In [26], a TRNG is presented based on the continuous unified chaotic system.

A common way of exploiting jitter is by using ring oscillators (ROs) (such as in [16, 17, 27] for example). Figure 2.3 shows the architecture of a RO. It consists of an odd number of inverters chained together in a ring. Due to the propagation delay of the inverters, an oscillating signal is produced. Often, one of the inverters is replaced with a NAND gate so that the inverting property can be turned on or off.

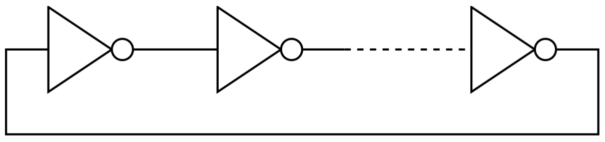

Figure 2.3: Architecture of a ring oscillator (RO)

These oscillations do not have a perfectly stable period. Due to random noise, the inverters may switch slightly before or after the expected switching time. This effect is illustrated in Figure 2.4 and referred to as *jitter* in the time domain. As more and more periods of the RO go by, this jitter accumulates and the total uncertainty on the switching time increases.

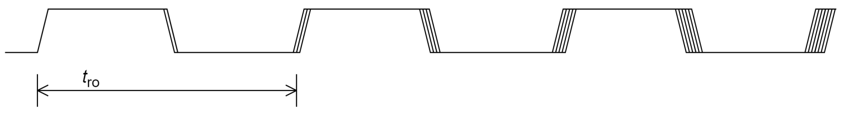

Figure 2.4: Illustration of the increasing uncertainty in switching time due to jitter.

2.2.2. Entropy Harvesting for Ring Oscillators

Harvesting jitter from ROs can be done in various ways. In [27], for example, Sunar et al. use several parallel ROs, sampled using an XOR tree followed by a flip-flop. However, the design by [27] is not random without post-processing, as shown in [28]. The design is improved by individually sampling the ring oscillators using flip-flops before the XOR tree [28], but no updated stochastic model is provided.

Baudet et al. [16] sample a single RO using a flip-flop, clocked with some sampling frequency. Using a phase-oriented approach, they also present a more precise study on the stochastic model of ROs.

In [17], Noumon Allini et al. propose a different method to generate random numbers from the single RO by using a counter instead of the flip-flop. From this counter, the least significant bit (LSB) is taken as a random number. They show that this method gives random numbers of significantly better quality compared to simply sampling the RO signal. However, they only present an empirical evaluation of the entropy.

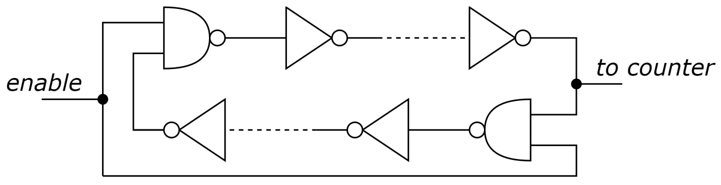

Figure 2.5: Architecture of a transient effect ring oscillator (TERO)

Varchola et al. in [29] take an entirely different direction, where the design of the RO itself is adjusted. The TERO, see Figure 2.5, still exploits jitter as a source of entropy, but the harvesting is different. The oscillator is constructed such that after the enable signal, it settles in a stable state within some random number of oscillations, which are counted using a simple ripple counter.

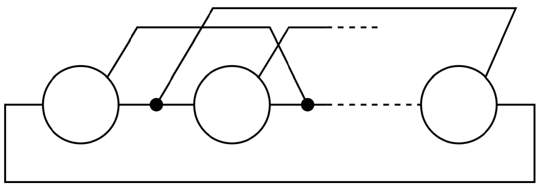

Figure 2.6: Architecture of a self-timed ring (STR)

Another design variation by Cherkaoui et al. [30] uses STRs to generate random numbers. In a STR, the inverters used in the RO are replaced with a Muller gate followed by an inverter. This new element takes input from the previous element in the ring and feedback from the element after it. An overview can be found in Figure 2.6. The output of each of the elements is sampled and XORed to generate the random output bits, which are still based on timing jitter.

2.3. Security of Ring Oscillators

The quality of the output of the TRNG has consequences for the security of the applications that depend on it. For example, a faulty TRNG that only outputs zeroes would make for a very poor key generator. Even in less extreme cases, security can be compromised by the overestimation of output entropy. In the following sections, some vulnerabilities in ROs are given, followed by an overview of different types of attacks against RO-based TRNGs.

2.3.1. Vulnerabilities

In [16], Baudet et al. show that the randomness of a TRNG can be overestimated under the influence of global deterministic perturbations (or 'global jitter'), a claim that is supported by [15]. This global jitter can be introduced by supply voltage or temperature changes, among others, as Valtchanov et al. described in [31].

Next to changes to the amount of jitter, the period of a RO can also change under external influences. This effect is exploited in [32, 33, 34], where ROs are used to measure die temperature changes and supply voltage fluctuations on FPGAs. The study by De Micco et al. [34] shows sensitivities of $0.02MHz/°C$ and $0.01MHz/mV$.

Fischer et al. propose a simple countermeasure to such changes in period and jitter in [35]. Instead of sampling using the system clock, another RO is used. This way, both clock signals are influenced by the same global effects, compensating for each other. It requires the ROs to be placed close to each other in the hardware and have similar oscillation frequencies. Similar results were obtained by [17].

The usage of a second RO for the sample clock will also introduce more entropy into the noise source, as will be shown in Chapter 3.

A last possible vulnerability in ROs is their electromagnetic emanation. By closely examining the frequency spectrum of the EM emanations at different working conditions, significant information on the ROs, such as location and frequency, can be obtained [36]. This information can then be used to simplify various attacks.

2.3.2. Attacks

In [37], Ngo et al. use power traces and a deep-learning model to attack the RO-based TRNG in an STM32 MCU. The attack uses side channels and is non-invasive, but it does require physical access to measure the power usage of the device precisely.

An invasive attack is carried out by Markettos & Moore in [38]. A $900mV$ sine wave is injected on the $5V$ power rail of a TRNG based on two ROs, and it is shown that it is possible to lock the ROs to the frequency of the sine wave. When this happens, the entropy of the noise source is drastically reduced. The proposed attack lowers the ability of the microcontroller to produce 4 billion (2^{32}) random numbers to just 255 ($< 2^8$) [38].

In [39], Bayon et al. improved on this attack with the use of electromagnetic fault injection (EMFI). This way, the attacker does not need access to the power rail of the TRNG. Next to that, the attacked TRNG uses 50 ROs. This is considerably more than the two used by [38] and also more common in real cryptographic devices. Using a probe located closely to the location of the ROs in the FPGA, leaving the FPGA packaging intact, the frequency of the ROs could be locked to that of the EM field induced by the probe. This drastically increased the bias of the noise source, making it fail all statistical tests.

Mitigating the effects of EMFI is therefore essential for ensuring the security and reliability of RO-based TRNGs. Many papers have been published with strategies to mitigate the effect of electromagnetic interference [40, 41, 42].

Martin et al. [43] show that exposure to ionising radiation can also impact the performance of the RO. The quality of the randomness of the TRNG was determined for different accumulated doses of radiation using NIST test suites. It was shown that the quality of the TRNG deteriorates before the first failure in the deterministic blocks, meaning security is compromised before the device fails.

In [44], instead of the entropy source (a STR), the entropy harvester (an XOR tree) is attacked. By introducing various power and clock glitches, the critical path delay of the XOR tree is violated, introducing faults in the output. Thus, although the randomness of the entropy source is not compromised, the randomness at the output of the RNG is decreased.

2.4. Conclusion

This chapter presented an introduction to random number generation in hardware, with a focus on FPGAs. First, the different types of RNGs were discussed. Then, the typical hardware-based TRNG components were explained using Cicek's classification. An overview of the entropy sources in FPGAs was given, focusing on the RO based designs, as they are the focus of this thesis. Lastly, Section 2.3 presented a brief overview of the security aspects of RO based TRNGs.

3 Stochastic Model

This chapter first describes the design of the TRNG noise source in Section 3.1. Then, in Section 3.2, a stochastic model for the output of this design is presented. In Section 3.3 this model is extended under the assumption of an ideal sampling time. The entropy of the noise source is estimated in Section 3.4. The stochastic model is then expanded to account for non-ideal sampling (Section 3.5), and the implications on the entropy are discussed. Some notes on the extension of the models to a more common TRNG design are discussed in Section 3.6. Section 3.7 concludes the chapter.

3.1. Design

In this thesis, a TRNG design based on counting RO periods is stochastically modelled and tested. The design is inspired by the work of Noumon Allini et al. [17].

The entropy source of this design is the timing jitter in the RO. The entropy harvester is the counter, counting the number of oscillations in a given sampling time. Together, these will often be referred to as the noise source. The post-processing step is left out of scope and simply the identity mapping is used.

Thus, the RO increments an *m*-bit counter, which is sampled with a sampling time T. At this sampling time, an amount of oscillations occurred, which will be called n_1. This n_1 is a realization of the random variable N_1, for which a distribution needs to be found. From this distribution, probabilities for the counter values can be computed. The value observed at the counter's output is simply $c = \lfloor n_1 \rfloor$, as it only increments every whole period.

Generally, the reference clock will have a frequency higher than the desired sample frequency, so a clock divider that counts n_2 clock cycles is used. An overview of the annotated design can be seen in Figure 3.1.

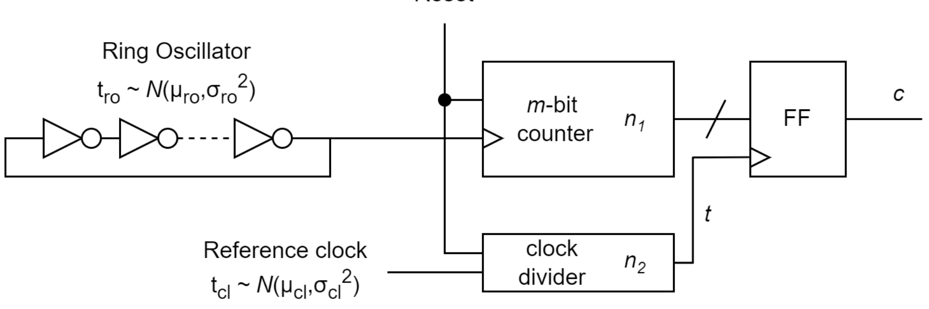

Figure 3.1: Design of the noise source.

3.2. Distribution of Oscillations

The period of a ring oscillator can be modelled as a combination of an ideal period μ_{ro} that depends on the number of inverters, or the *leg length*, and the routing delay between them, local jitter $\sim \mathcal{N}(0, \sigma_{ro}^2)$ and a global jitter that will not be taken into the model [31]. Together, the period of a ring oscillator is then distributed normally with period μ_{ro} and jitter σ_{ro}^2, and the time elapsed after n_1 ring oscillator periods is distributed $\sim \mathcal{N}(n_1\mu_{ro}, n_1\sigma_{ro}^2)$.

The same distribution holds for the sample time T, where the period is denoted with μ_{cl} and the jitter with σ_{cl}: $T \sim \mathcal{N}(n_2\mu_{cl}, n_2\sigma_{cl}^2)$. Figure 3.2 shows an illustration of these periods.

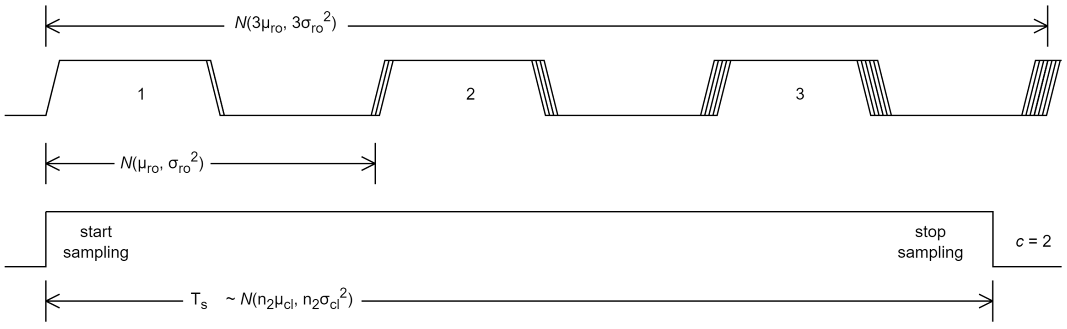

Figure 3.2: Illustration of Ring Oscillator periods with sampling time.

Both distributions model a time T given a number of oscillations. In the noise source design, however, the number of oscillations that happened in a set sampling time is measured. Thus, instead of a probability for T given n_1, a model is needed that, given a sampling time T, presents a distribution for the number of oscillations n_1 measured at the output of the m-bit counter, or $P(N_1 = n_1 \mid T = t)$.

Using Bayesian statistics and the fact that $n_1, t \geq 0$:

$$P(N_1 = n_1 \mid T = t) = \frac{P(T = t \mid N_1 = n_1)P(N_1)}{P(T)} = \frac{P(T = t \mid N_1 = n_1)P(N_1)}{\int\limits_0^\infty P(T = t \mid N_1 = n_1)P(N_1 = n_1)dn_1} \tag{3.1}$$

This requires us to find a prior distribution for $P(N_1)$. The only thing that is known about the value of N_1 when looking at the oscillator's output is that it is non-negative. However, at the output of the $m-$bit counter, N_1 is uniformly distributed over the interval $[0, 2^m)$. Assuming, without loss of generality as will be shown later, that m is large enough such that it is of no influence on the distribution $P(N_1 = n_1 \mid T = t)$:

$$P(N_1 \mid T) = \frac{P(T \mid N_1)P(N_1)}{\int\limits_0^\infty P(T \mid N_1)P(N_1)dn_1} = \frac{P(T \mid N_1)P(N_1)}{P(N_1)\int\limits_0^\infty P(T \mid N_1)dn_1} = \frac{P(T \mid N_1)}{\int\limits_0^\infty P(T \mid N_1)dn_1} \tag{3.2}$$

As $P(N_1)$ does not depend on n_1, but only on the size of the counter m. All that is left is to solve the integral:

$$\int\limits_0^\infty P(T = t \mid N_1 = n_1)dn_1 = \int\limits_0^\infty \frac{1}{\sigma_{ro}\sqrt{2n_1\pi}} e^{-\frac{(t-n_1\mu_{ro})^2}{2n_1\sigma_{ro}^2}} dn_1 = \frac{1}{\mu_{ro}} \tag{3.3}$$

The distribution of the number of oscillations N_1 in a sample period T is now given by:

$$P(N_1 \mid T) = f_{N_1|T}(n_1|t) = \begin{cases} \dfrac{\mu_{ro}}{\sigma\sqrt{2n_1\pi}} e^{-\frac{(t-n_1\mu_{ro})^2}{2n_1\sigma^2}} & \text{if } n_1 > 0, t \geq 0 \\ 0 & \text{otherwise} \end{cases} \tag{3.4}$$

3.3. Ideal Sample Time

In the coming sections, the reference clock will be assumed not to have any jitter, as it makes the calculations for a lower bound on the min-entropy less complex. This means that T is modelled as an ideal clock T_i with period $n_2\mu_{cl}$. In the following calculations, n_2 and μ_{cl} will even be ignored, and a period $t = n_2\mu_{cl}$ is taken instead as implementation details do not matter yet.

3.3.1. Expectation and Variance

The expectation of a continuous random variable is defined as $E[X] = \int\limits_{-\infty}^{\infty} xf(x)dx$. The expected amount of samples after a sampling period T_i is thus:

$$\begin{aligned} E[N_1 \mid T_i] &= \int\limits_{-\infty}^{\infty} n_1 f_{N_1|T_i}(n_1|t)dn_1 = \frac{\mu_{ro}}{\sigma_{ro}\sqrt{2\pi}} \int\limits_{0}^{\infty} \sqrt{n_1} e^{-\frac{(t-n\mu_{ro})^2}{2n\sigma_{ro}^2}} dn_1 \\ &= \frac{t\mu_{ro} + \sigma_{ro}^2}{\mu_{ro}^2} \end{aligned} \tag{3.5}$$

The expectation for the number of oscillations after a sampling period T_i is the sampling period divided by the period of one oscillation, as would be expected, plus a small constant based on the physical parameters of the noise source.

The variance of a variable X is defined as $Var[X] = E[X^2] - E[X]^2$:

$$\begin{aligned} E[N_1^2 \mid T_i] &= \int\limits_{-\infty}^{\infty} n_1^2 f_{N_1|T_i}(n_1|t)dn_1 = \frac{\mu_{ro}}{\sigma_{ro}\sqrt{2\pi}} \int\limits_{0}^{\infty} n_1\sqrt{n_1} e^{-\frac{(t-n\mu_{ro})^2}{2n_1\sigma_{ro}^2}} dn_1 \\ &= \frac{t^2\mu^2 + 3t\mu\sigma^2 + 3\sigma^4}{\mu^4} \end{aligned} \tag{3.6}$$

$$\begin{aligned} Var[N_1 \mid T_i] &= E[N_1^2 \mid T_i] - E[N_1 \mid T_i]^2 = \frac{t^2\mu_{ro}^2 + 3t\mu_{ro}\sigma_{ro}^2 + 3\sigma_{ro}^4}{\mu_{ro}^4} - \left(\frac{t\mu_{ro} + \sigma_{ro}^2}{\mu_{ro}^2}\right)^2 \\ &= \frac{t\mu_{ro}\sigma_{ro}^2 + 2\sigma_{ro}^4}{\mu_{ro}^4} \end{aligned} \tag{3.7}$$

3.3.2. Skewness and Kurtosis

The skewness and kurtosis of a random variable tell us something about the shape of the distribution. The value for the skewness indicates whether the distribution is symmetric (zero skew) or skewed positively or negatively. Kurtosis, on the other hand, is a measure of the 'tailedness' of a distribution. Higher kurtosis corresponds to more and more extreme outliers compared to low kurtosis. As the normal distribution has kurtosis 3, the *excess* kurtosis is often defined as the kurtosis minus 3.

As the integrals in this section are rather complex, they are computed using SymPy [45], a Python library for symbolic mathematics.

Skewness

The skewness is defined as $Skew[X] = E\left[\left(\frac{X-\mu}{\sigma}\right)^3 \mid T_i\right]$

$$\begin{aligned} Skew[N_1 \mid T_i] &= \frac{E[N_1^3 \mid T_i] - 4 \times E[N_1 \mid T_i] \times Var[N_1 \mid T_i] - E[N_1 \mid T_i]^3}{Var[N_1 \mid T_i]^{3/2}} \\ &= \frac{3t\sigma_{ro}\mu_{ro} + 8\sigma_{ro}^3}{(t\mu_{ro} + 2\sigma_{ro}^2)^{\frac{3}{2}}} \end{aligned} \tag{3.8}$$

Where the third moment of $(N_1 \mid T_i)$ is computed as:

$$\begin{aligned} E[N_1^3 \mid T_i] &= \int n_1^3 f_{N_1|T_i}(n_1|t)dn_1 = \frac{\mu_{ro}}{\sigma_{ro}\sqrt{2\pi}} \int_0^\infty n_1^2 \sqrt{n_1} e^{-\frac{(t-n_1\mu_{ro})^2}{2n_1\sigma_{ro}^2}} dn_1 \\ &= \frac{t^3\mu_{ro}^3 + 6t^2\mu_{ro}^2\sigma_{ro}^2 + 15t\mu_{ro}\sigma_{ro}^4 + 15\sigma_{ro}^6}{\mu_{ro}^6} \end{aligned} \tag{3.9}$$

As all parameters for the skewness are positive, the distribution of N_1 is positively (or right) skewed. When the sampling period is increased, the skewness decreases, and the distribution becomes symmetric for $t \to \infty$ (Equation 3.10).

$$\lim_{t\to\infty} Skew[N_1 \mid T_i] = \lim_{t\to\infty} \frac{3t\sigma_{ro}\mu_{ro} + 8\sigma_{ro}^3}{(t\mu_{ro} + 2\sigma_{ro}^2)^{\frac{3}{2}}} = 0 \tag{3.10}$$

Kurtosis

The kurtosis is defined as $Kurt[X] = E\left[\left(\frac{X-\mu}{\sigma}\right)^4\right]$

$$\begin{aligned} Kurt[N_1 \mid T_i] &= \frac{E[N_1^4 \mid T_i] - 4E[N_1 \mid T_i] \times E[N_1^3 \mid T_i] + 6E[N_1 \mid T_i]^2 \times E[N_1^2 \mid T_i] + 3E[N_1 \mid T_i]^4}{Var[N_1 \mid T_i]^2} \\ &= \frac{3\left(t^2\mu^2 + 9t\mu\sigma_{ro}^2 + 20\sigma_{ro}^4\right)}{t^2\mu^2 + 4t\mu\sigma_{ro}^2 + 4\sigma_{ro}^4} \end{aligned}$$

Where the fourth moment of $(N_1 \mid T_i)$ is computed as:

$$\begin{aligned} E[N_1^4 \mid T_i] &= \int n^4 f_{N_1|T_i}(n)dn = \frac{\mu}{\sigma\sqrt{2\pi}} \int_0^\infty n^3 \sqrt{n} e^{-\frac{(t-n\mu)^2}{2n\sigma^2}} dn \\ &= \frac{t^4\mu^4 + 10t^3\mu^3\sigma_{ro}^2 + 45t^2\mu^2\sigma_{ro}^4 + 105t\mu\sigma_{ro}^6 + 105\sigma_{ro}^8}{\mu^8} \end{aligned} \tag{3.11}$$

Computing the kurtosis for $T_i \to \infty$ (Equation 3.12) shows that the kurtosis of the distribution of N tends to 3 or that the excess kurtosis tends to zero. As mentioned before, this is the same kurtosis as that of the normal distribution.

$$\lim_{T_i\to\infty} Kurt[N] = \lim_{T_i\to\infty} \frac{3\left(T_i^2\mu^2 + T_i\mu\sigma_{ro}^2 + 20\sigma_{ro}^4\right)}{T_i^2\mu^2 + 4T_i\mu\sigma_{ro}^2 + 4\sigma_{ro}^4} = 3 \tag{3.12}$$

3.3.3. Normal Approximation

The previous section showed that the skewness and excess kurtosis tend to zero as the sampling time increases. This, in combination with the shape of the distribution, implies that for large sampling times, the distribution in Equation 3.4 can be approximated with a normal distribution with parameters $E[N_1|T_i]$ and $Var[N_1|T_i]$.

$$N_1 \mid T_i \sim \mathcal{N}\left(E[N_1 \mid T_i], Var[N_1 \mid T_i]\right) = \mathcal{N}\left(\frac{t\mu_{ro} + \sigma_{ro}^2}{\mu_{ro}^2}, \frac{t\mu_{ro}\sigma_{ro}^2 + 2\sigma_{ro}^4}{\mu_{ro}^4}\right),\ t \gg 0 \tag{3.13}$$

This can be even further simplified (Equation 3.14) as, in practice, the oscillator period is large in comparison to the timing jitter, $\mu_{ro} \gg \sigma_{ro}^2$. Since for the normal approximation to hold t is also large, $t\mu_{ro} \gg \sigma_{ro}^2$, further supporting the simplification.

$$N_1 \mid T_i \sim \mathcal{N}\left(\frac{t}{\mu_{ro}}, \frac{t\sigma_{ro}^2}{\mu_{ro}^3}\right),\ t \gg 0 \tag{3.14}$$

3.4. Entropy Estimations

Now that it is shown that the distribution of the number of oscillations at the output of the counter can be approximated using a normal distribution (under the right conditions), the next step is to estimate the randomness of the output bits. Section 3.4.1 discusses how this randomness is calculated. Then, a translation is made from the continuous distribution to discrete counter values. The randomness for a counter bit can be computed using these counter values. Section 3.4.3 discusses the worst- and best-case scenarios for the output entropy. With all of this information, a minimum sampling time for good randomness is calculated.

3.4.1. Min-Entropy

The entropy used to estimate the randomness of the output bits is the min-entropy, defined as [13, p. 86]:

$$H_{min}(X) = -\log_2\left(\max_{1\leq i\leq k}\{P(X=\omega_i)\}\right) \tag{3.15}$$

Where X is a random variable that assumes values in the finite set $\Omega = \{\omega_1, \omega_2, ..., \omega_k\}$. For the entropy estimation, the individual output bits of the counter will be used, meaning that for every bit, the set $\Omega = \{0, 1\}$, and the min-entropy is loosely defined as the negative log of the probability of the most likely output bit.

The reason to use min-entropy is that it defines the greatest lower bound on the entropy [46], meaning that each observation of the random variable has at least H_{min} information. This is a very nice property when using the TRNG for cryptographic applications. AIS 20/31 specifies that the min-entropy for internal random numbers should be at least 0.98 per bit [13].

3.4.2. Discrete Counter Values

At the moment, the distribution of the amount of oscillations is continuous. At the start of this chapter, it was already briefly mentioned that the observed value at the output of the counter $c = \lfloor n_1 \rfloor$. The probability for a particular counter value c is thus equal to the integral of the probability density function of $N_1|T_i$ from c to $c+1$. More formally:

$$P(C=c) = P(N_1 < c+1 \mid T_i) - P(N_1 < c \mid T_i),\ c \in \{0, 1, 2, ..., 2^m - 1\} \tag{3.16}$$

It is still assumed that m is large enough not to influence the distribution.

This is illustrated in Figure 3.3, where every region bounded by two adjacent dotted lines, the x-axis and the curve of $P(N_1|T_i)$ corresponds to a $P(C=c)$.

The probability of a certain bit having a value of 0 (or 1) is now also well-defined. It is simply the sum of all the $P(C=c)$ in which that bit of the counter value c equals 0 (or 1). For example, even counter values correspond to LSB = 0, while odd counter values correspond to LSB = 1, and therefore:

$$\begin{aligned} P(LSB=0) &= \sum_{c=0,2,...}^{2^{m-1}} P(C=c) \\ P(LSB=1) &= \sum_{c=1,3,...}^{2^m-1} P(C=c) \end{aligned} \tag{3.17}$$

3.4.3. Worst Case Estimation

Calculating the min-entropy for the LSB based on Figure 3.3 and Equation 3.17 gives $H_{min}(X) = 1$. Actually, it is not even necessary to calculate anything, as the normal distribution is symmetric, and it follows that the areas of even and uneven counter values are symmetric as well.

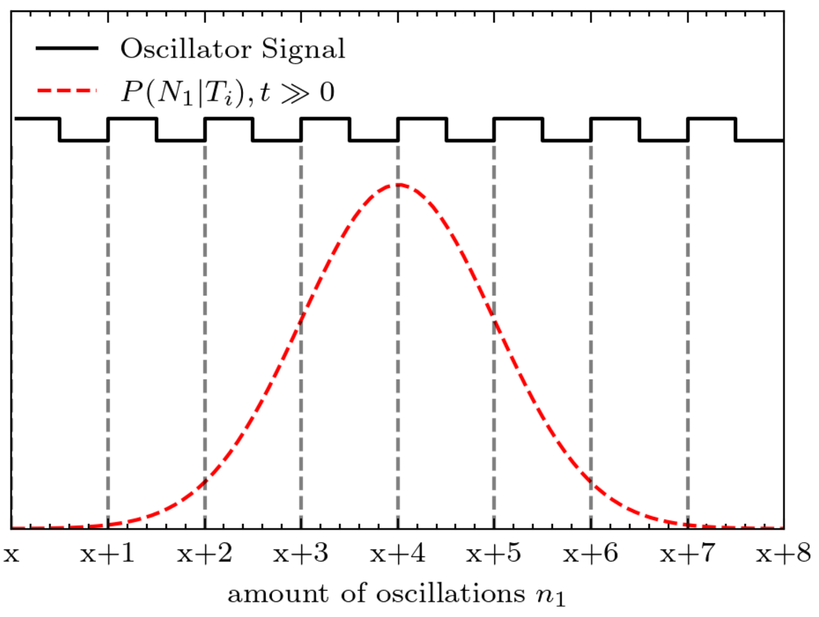

Figure 3.3: Illustration of the ring oscillator signal and the probability density function $P(N_1|T_i), t \gg 0$.

The simplest would then be to ensure that the RO is always sampled at this exact point. In hardware, however, this is non-trivial. The sampling time t can most likely only be chosen in discrete steps, and the period of the RO can change over time.

If, or when, this happens, the areas will no longer be symmetric as illustrated in Figure 3.4. The top figure shows the perfectly aligned case, where the min-entropy of the LSB is 1, no matter the variance of the normal distribution. The bottom figure shows the worst-case scenario when the distribution is shifted exactly a half oscillation. For a robust design, the worst-case min-entropy of the LSB should be such that the device still meets the min-entropy requirements.

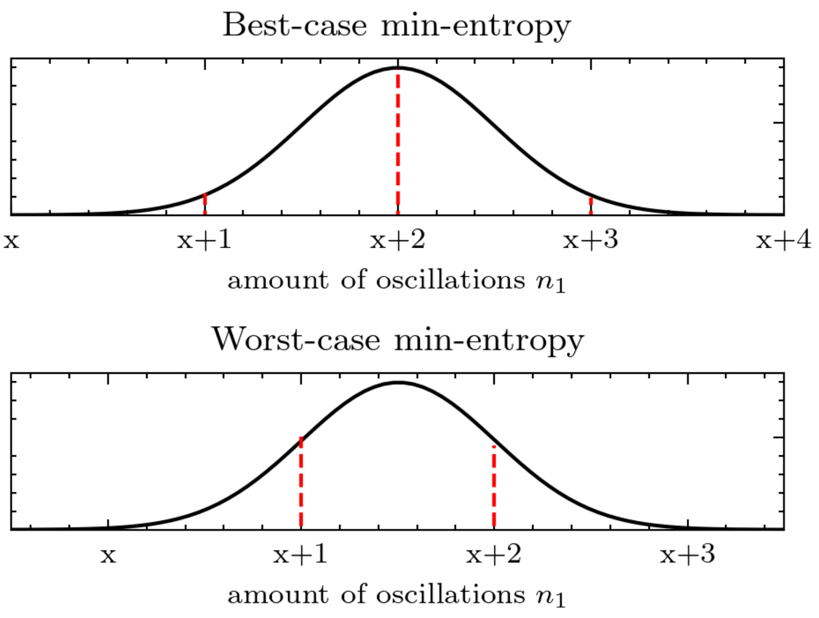

Figure 3.4: Illustration of the distribution $P(N_1|T_i)$ for best- and worst-case min-entropy.

3.4.4. Sampling Time

To calculate the min-entropy for the worst case, first note that for a normal distribution 99.9999% of the values lie within five standard deviations from the mean. This interval is denoted as the width of the distribution, $w = 10 \times \sqrt{\frac{t\sigma_{ro}^2}{\mu_{ro}^3}}$, and it is assumed that values outside the interval occur with probability 0.

When increasing the width, for example, by increasing the sampling time as per Equation 3.14, more counter values will have a non-zero probability of occurring. Meaning the min-entropy will increase.

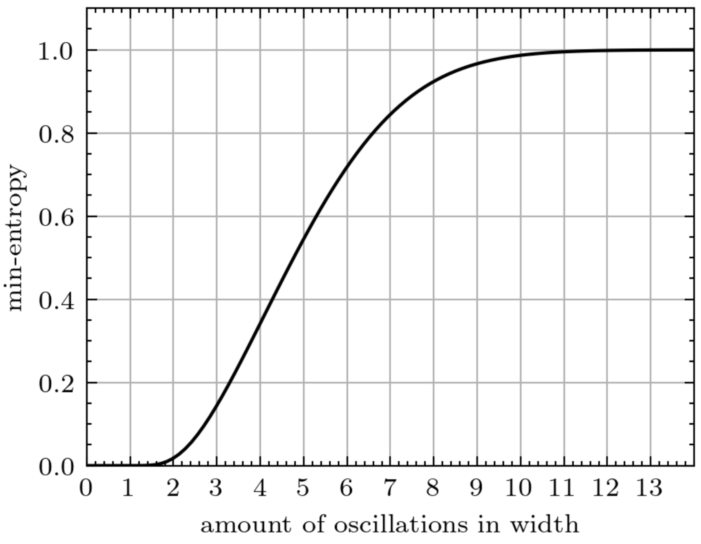

Figure 3.5: Worst-case min-entropy of the LSB as a function of the width of the normal distribution.

Figure 3.5 plots the worst-case min-entropy as a function of the distribution width w. A min-entropy of at least 0.98 is achieved when $w \geq 10$. From this, the desired sampling time t can be calculated as:

$$\begin{aligned} w &\geq 10 \\ w &= 10 \times \sqrt{\frac{t\sigma_{ro}^2}{\mu_{ro}^3}} \\ \sqrt{\frac{t\sigma_{ro}^2}{\mu_{ro}^3}} &\geq 1 \\ \frac{t\sigma_{ro}^2}{\mu_{ro}^3} &\geq 1^2 \\ t &\geq \frac{\mu_{ro}^3}{\sigma_{ro}^2} \end{aligned} \tag{3.18}$$

Putting this in the normal approximation in Equation 3.14, this means that the min-entropy is ≥ 0.98 when the variance is at least equal to 1. Next to that, the requirement that $t\mu_{ro} \gg \sigma_{ro}$ is also satisfied, as $\mu_{ro}^4 \gg \sigma_{ro}^3$.

To achieve a min-entropy $H_{min}(LSB) \geq 0.98$, the sample time t can thus be calculated given the physical parameters of the ring oscillator. Practically speaking, this means that the counting RO noise source can function as a TRNG without post-processing when taking the LSB as output and sufficiently long sample time as per Equation 3.18.

Equation 3.18 also tells us that the minimum sampling time for a min-entropy of 0.98 increases with the RO period cubed. As the period of a RO grows with the number of inverters it consists of, this means that ROs with fewer inverters allow for shorter sampling times. Using fewer inverters also means using less area, making a RO with a small period both less costly and more efficient.

The other bits in the counter also have some entropy. To estimate their worst-case min-entropy, note that the second bit changes value every two oscillations, the third bit every four oscillations, etc. The perceived width for every next bit is thus half that of the bit before it. Table 3.1 shows the H_{min} for bits 1, 2 and 3 when the sampling time t is chosen such that the H_{min} of the 0^{th} bit, the LSB, is $\geq$0.98. It makes sense to also use bits 1 and 2 in a post-processing step, as it improves the efficiency of the TRNG by over 60%.

Table 3.1: Worst-case min-entropy per bit with t such that $H_{min}(LSB) = 0.98$

Bit	H_{min}
0 (LSB)	0.98
1	0.55
2	0.07
3	0.00

3.4.5. Sizing the Counter

Until now, the counter was assumed to be large enough that it is possible to count the number of oscillations precisely. Table 3.1 shows that in a practical application, it would be preferred to use only 3 bits for the counter, as there is no entropy in the other bits. As long as the counter simply overflows, this also does not affect the calculations made thus far. The probabilities would merely add up, which is precisely the desired effect as per Equation 3.17. This is illustrated in Figure 3.6.

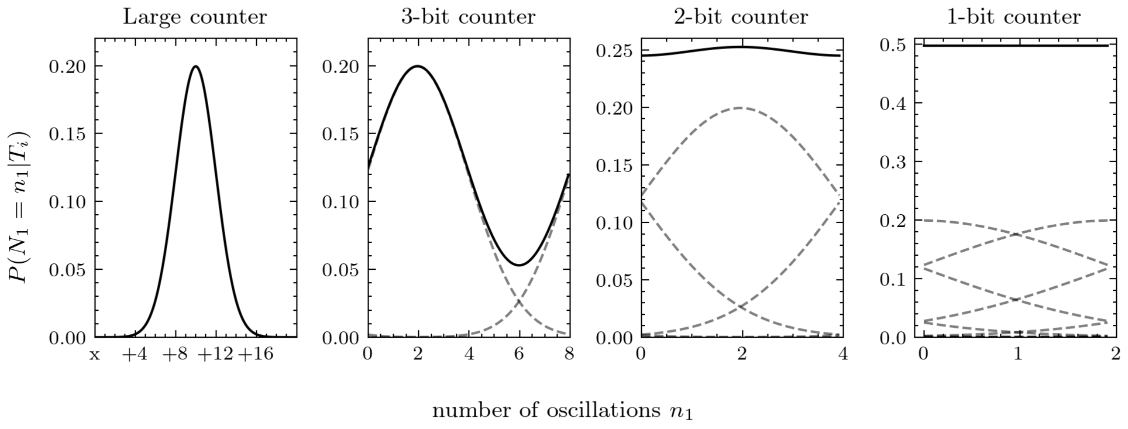

Figure 3.6: Illustration of the distribution in counters of different sizes.

3.5. Non-Ideal Sample Time

In this section, the assumption of an ideal sample time is dropped. Instead, the sample time is modelled as a normal distribution T_n (Equation 3.19), as described in Section 3.2.

$$P(T_n = t|N_2 = n_2) = f_{T_n|N_2}(t|n_2) = \frac{1}{\sqrt{2n_2\pi}\sigma_{cl}} e^{-\frac{(T_n - n_2\mu_{cl})^2}{2n_2\sigma_{cl}^2}} \tag{3.19}$$

By using this distribution for the sampling time, an attempt is made to calculate a more realistic distribution for N_1. First, the distribution is computed, after which it is compared against the distribution given an ideal sampling time calculated before. Then, the expectation and variance of the distribution are computed. Ideally, the expectation does not change while the variance increases.

By the law of total probability:

$$\begin{aligned} P(N_1) &= \int_0^\infty f_{N_1|N_2}(n_1|n_2) f_{N_2}(n_2) dn_2 \\ P(N_1) &= \int_0^\infty f_{N_1|T_n}(n_1|t) f_{T_n}(t) dt \\ P(T_n) &= \int_0^\infty f_{T_n|N_2}(t|n_2) f_{N_2}(n_2) dn_2 \end{aligned} \tag{3.20}$$

Where the integral is taken from zero instead of $-\infty$ since the probability of negative n_1, n_2 and t is zero in a practical application.

Equating $P(N_1)$ and filling in for $f_{T_n}(t) = P(T_n)$:

$$\int_0^\infty f_{N_1|N_2}(n_1|n_2) f_{N_2}(n_2) dn_2 = \int_0^\infty f_{N_1|T_n}(n_1|t) \left(\int_0^\infty f_{T_n|N_2}(t|n_2) f_{N_2}(n_2) dn_2 \right) dt \tag{3.21}$$

Since $f_{N_1|T_n}(n_1|t)$ does not depend on n_2, it can be taken into the integral:

$$\int_0^\infty f_{N_1|N_2}(n_1|n_2) f_{N_2}(n_2) dn_2 = \int_0^\infty \int_0^\infty f_{N_1|T_n}(n_1|t) f_{T_n|N_2}(t|n_2) f_{N_2}(n_2) dn_2 \, dt \tag{3.22}$$

As $f_{N_1|T_n}(n_1|t) f_{T_n|N_2}(t|n_2) f_{N_2}(n_2)$ is non-negative and measurable, per definition of a probability density function, and $[0, \infty)$ is σ-finite, by Tonelli's theorem the order of integration can be swapped:

$$\int_0^\infty f_{N_1|N_2}(n_1|n_2) f_{N_2}(n_2) dn_2 = \int_0^\infty \int_0^\infty f_{N_1|T_n}(n_1|t) f_{T_n|N_2}(t|n_2) f_{N_2}(n_2) dt \, dn_2 \tag{3.23}$$

Now both sides can be differentiated with respect to n_2, and $f_{N_2}(n_2)$ cancels out:

$$\begin{aligned} f_{N_1|N_2}(n_1|n_2) f_{N_2}(n_2) &= \int_0^\infty f_{N_1|T_n}(n_1|t) f_{T_n|N_2}(t|n_2) f_{N_2}(n_2) dt \\ f_{N_1|N_2}(n_1|n_2) &= \int_0^\infty f_{N_1|T_n}(n_1|t) f_{T_n|N_2}(t|n_2) dt \end{aligned} \tag{3.24}$$

The distribution of the output of the noise source N_1, with a non-ideal sample clock divided by some n_2, is given by:

$$P(N_1 = n_1|N_2 = n_2) = f_{N_1|N_2}(n_1|n_2) = \int_0^\infty f_{N_1|T_n}(n_1|t) f_{T_n|N_2}(t|n_2) dt \tag{3.25}$$

3.5.1. Consistency with Ideal Model

The sample time T_n is distributed normally with jitter σ_{cl}. When this jitter σ_{cl} goes to zero, the sample time becomes ideal, and thus the non-ideal model should coincide with the ideal model for $\lim \sigma_{cl} \to 0$:

$$\begin{aligned} f_{T_n|N_2}(t|n_2) &= \frac{1}{\sigma_{cl}\sqrt{2n_2\pi}} e^{-\frac{(t-n_2\mu_{cl})^2}{2n_2\sigma_{cl}^2}} \\ \lim_{\sigma_{cl}\to 0} \frac{1}{\sigma_{cl}\sqrt{2n_2\pi}} e^{-\frac{(t-n_2\mu_{cl})^2}{2n_2\sigma_{cl}^2}} &= \delta(t - n_2\mu_{cl}) \end{aligned} \tag{3.26}$$

Where δ denotes the Dirac distribution. Filling in:

$$\begin{aligned} P(N_1 \mid N_2) &= \int_0^\infty f_{N_1|T_n}(n_1|t)\delta(t - n_2\mu_{cl}) dt \\ P(N_1 \mid N_2) &= P(N_1 \mid T_n = n_2\mu_{cl}) \end{aligned} \tag{3.27}$$

Which is the same as the probability given ideal sampling time $T_i = n_2\mu_{cl}$ as described in Section 3.3, meaning the models are consistent.

3.5.2. Expectation and Variance

Ideally, the expectation for the distribution of N_1 stays the same when jitter is introduced in the sampling time T, while the variance should preferably increase.

Expectation

The expectation of $P(N_1 \mid N_2)$ is computed as:

$$\begin{aligned} E[N_1|N_2] &= \int_0^\infty n_1 f_{N_1|N_2}(n_1|n_2) dn_1 \\ &= \int_0^\infty n_1 \int_0^\infty f_{N_1|T_n}(n_1|t) f_{T_n|N_2}(t|n_2) dt\ dn_1 \\ &= \int_0^\infty \int_0^\infty n_1 f_{N_1|T_n}(n_1|t) f_{T_n|N_2}(t|n_2) dt\ dn_1 \end{aligned} \tag{3.28}$$

According to Tonelli's theorem, the order of integration can be swapped again, as $[0, \infty)$ is σ-finite and $n_1 f_{N_1|T_n}(n_1|t) f_{T_n|N_2}(t|n_2)$ is non-negative and measurable on this domain. Then, $f_{T_n|N_2}(t|n_2)$ can be taken out of the integral as it does not depend on n_1:

$$E[N_1|N_2] = \int_0^\infty \left(\int_0^\infty n_1 f_{N_1|T_n}(n_1|t) dn_1 \right) f_{T_n|N_2}(t|n_2) dt \tag{3.29}$$

This inner integral has been computed before (Equation 3.5):

$$\int_0^\infty n_1 f_{N_1|T_n}(n_1|t) dn_1 = E[N_1|T_n] = E[N_1|T_i] = \frac{t\mu_{ro} + \sigma_{ro}^2}{\mu_{ro}^2} \tag{3.30}$$

Filling in gives:

$$\begin{aligned} E[N_1|N_2] &= \int_0^\infty \left(\frac{t\mu_{ro} + \sigma_{ro}^2}{\mu_{ro}^2}\right) f_{T_n|N_2}(t|n_2)dt \\ &= \frac{\sigma_{ro}^2}{\mu_{ro}} \int_0^\infty t f_{T_n|N_2}(t|n_2)dt + \frac{\sigma_{ro}^2}{\mu_{ro}^2} \int_0^\infty f_{T_n|N_2}(t|n_2)dt \end{aligned} \tag{3.31}$$

Here, the integrals are almost equal to the normal distribution's expectation and cumulative density function, which is taken from $-\infty$ to ∞. However, it is known that the distribution is shifted along the positive axis. Since its standard deviation is very small compared to the mean, it is safe to assume the part from $-\infty$ to 0 is negligible. The first integral then simply evaluates to the expected value $n_2\mu_{cl}$, and the second integral to 1:

$$\begin{aligned} E[N_1|N_2] &= \frac{\sigma_{ro}^2}{\mu_{ro}}(n_2\mu_{cl}) + \frac{\sigma_{ro}^2}{\mu_{ro}^2} \times 1 \\ &= \frac{n_2\mu_{cl}\mu_{ro} + \sigma_{ro}^2}{\mu_{ro}^2} \end{aligned} \tag{3.32}$$

Comparing Equation 3.32 with Equation 3.5, they are equal when the ideal sampling time is chosen as $n_2\mu_{cl}$.

Variance

Intuitively, using a non-ideal clock will increase the variance compared to the ideal case. To test this hypothesis, note that the expectation is the same in both scenarios and thus:

$$\begin{aligned} Var[N_1|N_2] &= E[N_1^2|N_2] - E[N_1|N_2]^2 \\ Var[N_1|T_i] &= E[N_1^2|T_i] - E[N_1|T_i]^2 \\ E[N_1|N_2] &= E[N_1|T_i] \\ Var[N_1|N_2] \geq Var[N_1|T_i] \ &\textit{iff} \ E[N_1^2|N_2] \geq E[N_1^2|T_i] \end{aligned} \tag{3.33}$$

To compute the second moment of $N_1|N_2$, the same steps as for the expectation are taken:

$$\begin{aligned} E[N_1^2|N_2] &= \int_0^\infty n_1^2 f_{N_1|N_2}(n_1|n_2)dn_1 \\ &= \int_0^\infty \int_0^\infty n_1^2 f_{N_1|T_n}(n_1|t) f_{T_n|N_2}(t|n_2)dt\ dn_1 \\ &= \int_0^\infty \left(\int_0^\infty n_1^2 f_{N_1|T_n}(n_1|t)dn_1\right) f_{T_n|N_2}(t|n_2)dt \end{aligned} \tag{3.34}$$

Again, this inner integral has been computed before (Equation 3.6):

$$\int_0^\infty n_1^2 f_{N_1|T_n}(n_1|t)dn_1 = E[N_1^2|T_n] = E[N_1^2|T_i] = \frac{t^2\mu_{ro}^2 + 3t\mu_{ro}\sigma_{ro}^2 + 3\sigma_{ro}^4}{\mu_{ro}^4} \tag{3.35}$$

Filling in gives:

$$
\begin{aligned}
E[N_1^2|N_2] &= \int_{-\infty}^{\infty} \left[\frac{t^2\mu_{ro}^2 + 3t\mu_{ro}\sigma_{ro}^2 + 3\sigma_{ro}^4}{\mu_{ro}^4}\right] f_{T_n|N_2}(t|n_2)dt \\
&= \frac{1}{\mu_{ro}^2}\int_0^{\infty} t^2 f_{T_n|N_2}(t|n_2)dt + \frac{3\sigma_{ro}^2}{\mu_{ro}^3}\int_0^{\infty} t f_{T_n|N_2}(t|n_2)dt + \frac{3\sigma_{ro}^4}{\mu_{ro}^4}\int_0^{\infty} f_{T_n|N_2}(t|n_2)dt \\
&= \frac{1}{\mu_{ro}^2}(n_2^2\mu_{cl}^2 + n_2\sigma_{cl}^2) + \frac{3\sigma_{ro}^2}{\mu_{ro}^3}(n_2\mu_{cl}) + \frac{3\sigma_{ro}^4}{\mu_{ro}^4} \\
&= \frac{n_2^2\mu_{cl}^2\mu_{ro}^2 + 3n_2\mu_{cl}\sigma_{ro}^2\mu_{ro} + 3\sigma_{ro}^4}{\mu_{ro}^4} + \frac{n_2\sigma_{cl}^2}{\mu_{ro}^2}
\end{aligned}
\tag{3.36}
$$

Where the same approximation for the integrals of the normal is used, and the fact that:

$$
\int_0^{\infty} t^2 f_{T_n|N_2}(t|n_2)dt = E[T_n^2|N_2] = (E[T_n|N_2])^2 + Var[T_n|N_2] = n_2^2\mu_{cl}^2 + n_2\sigma_{cl}^2 \tag{3.37}
$$

Choosing $t = n_2\mu_{cl}$ for the sample time in Equation 3.6:

$$
\begin{aligned}
E[N_1^2|T_i] &= \frac{n_2^2\mu_{cl}^2\mu_{ro}^2 + 3\mu_{cl}n_2\mu_{ro}\sigma_{ro}^2 + 3\sigma_{ro}^4}{\mu_{ro}^4} \\
E[N_1^2|N_2] &= \frac{n_2^2\mu_{cl}^2\mu_{ro}^2 + 3n_2\mu_{cl}\sigma_{ro}^2\mu_{ro} + 3\sigma_{ro}^4}{\mu_{ro}^4} + \frac{n_2\sigma_{cl}^2}{\mu_{ro}^2} \\
E[N_1^2|N_2] - E[N_1^2|T_i] &= \frac{n_2\sigma_{cl}^2}{\mu_{ro}^2} \geq 0 \ \text{(q.e.d.)}
\end{aligned}
\tag{3.38}
$$

This is the variance increase due to the sample clock having non-zero jitter. This increase depends on n_2, which is the amount of sample clock cycles, σ_{cl}, the jitter on every sample clock cycle, and μ_{ro}, the period of the sampled RO. Adding this increase in variance to the variance found in Equation 3.7, the total variance under a non-ideal clock is equal to:

$$
\begin{aligned}
Var[N_1 \mid N_2] &= \frac{n_2\sigma_{cl}^2}{\mu_{ro}^2} + \frac{n_2\mu_{cl}\mu_{ro}\sigma_{ro}^2 + 2\sigma_{ro}^4}{\mu_{ro}^4} \\
&= \frac{n_2\left(\mu_{ro}\sigma_{cl}^2 + \mu_{cl}\sigma_{ro}^2\right)}{\mu_{ro}^3} + \frac{2\sigma_{ro}^4}{\mu_{ro}^4}
\end{aligned}
\tag{3.39}
$$

When implementing the reference clock as a second ring oscillator with approximately the same physical properties as the first ring oscillator, this becomes:

$$
Var[N_1 \mid N_2] = 2\frac{n_2\sigma_{ro}^2}{\mu_{ro}^2} + \frac{2\sigma_{ro}^4}{\mu_{ro}^4} \approx 2 \times Var[N_1 \mid T_i] \tag{3.40}
$$

This means that using a second ring oscillator to sample the first one approximately doubles the variance at the output of the noise source. As the variance is linear with time, practically speaking this means that the sampling time calculated with Equation 3.18 can be halved to achieve the same worst-case min-entropy.

3.6. Extension to Flip-Flop Design

This improvement in variance when sampling with a second RO is clearly a very useful property of the proposed noise source. There are, however, also different methods of generating random bits from ROs, of which the most commonly used one is to sample the RO using a flip-flop. This design can be seen in Figure 3.7. The output from several of these elements is then XORed to form the output bits. It is, for example, used in [28].

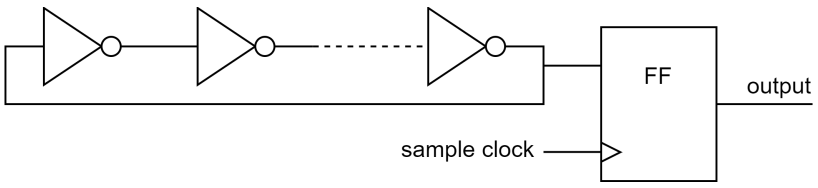

Figure 3.7: Ring oscillator sampled using a flip-flop.

The output of this flip-flop can be modelled as the LSB of a counter that is increased not only on the rising edge of the oscillating signal but also on the falling edge. As this does not violate any of the assumptions made so far, it is safe to say that the variance of such a design will also increase when using a RO for the sample clock.

Furthermore, the counter value c now equals $\lfloor 2n_1 \rfloor \mod 2$, assuming the oscillator signal has a duty cycle of 50%. This implies that the relative width of the distribution also doubles, and the sample time t can be halved. Note that it is *very important* that the assumption on the duty cycle holds. Otherwise, there will be a bias in the output bits, and the min-entropy will be lower than expected. The noise source proposed in this thesis does not have this problem, as only the rising edges are counted, and a bias in the duty cycle is of no influence.

3.7. Conclusion

This chapter presented a design for a TRNG noise source. This design was then first modelled under the assumption of an ideal sample clock, and it was shown that this model could be approximated as a normal distribution with expectation and variance according to Equation 3.14 when the sampling time was taken appropriately large.

Using this model, it was possible to calculate the minimum sampling time to achieve the required min-entropy of at least 0.98 for the LSB. This sampling time can be reduced for ROs with fewer inverters, allowing for small and efficient designs.

The model was then extended for a non-ideal sample clock. It was shown that this model is consistent with the ideal clock model and that the variance increases as the jitter of the sample clock increases. When the sample clock is chosen as a RO with the same period and jitter as the sampled RO, the variance approximately doubles compared to the ideal clock, which means only half the sample time is needed for the same min-entropy.

4 Verification

In this chapter, simulations are performed to verify that they correspond to the mathematical models and conclusion from Chapter 3. A first set of simulations is carried out using Rust. These simulations are quick and easy to do, but their significance is limited as the simulation only uses a very simple noise model. The methods used and the results of these simulations can be found in Section 4.1. To improve on the Rust simulations, SPICE simulations are carried out using Cadence Spectre. These simulations provide a better understanding of how the noise source will perform in hardware. Section 4.2 describes the used models and parameters, and presents the results of the SPICE simulations. A conclusion to this chapter is presented in Section 4.3. Appendix A contains all the important code snippets and plots of the simulation results.

4.1. Rust Simulations

The first set of simulations to verify the models is carried out using Rust. In Section 4.1.1, simulations under the assumption of an ideal clock are carried out. Section 4.1.2 contains the simulations for a non-ideal model, as well as some discussion on the difference between the two.

In Rust, the `rand` and `rand_distr` crate [47] are used for the simulations. The code can be found in Appendix A.1.1. The `rand` crate provides the `ThreadRng` struct, which generates random numbers using the ChaCha12 stream cypher [48] as DRNG, seeded by the operating system every 64 KiB of random. Using these crates, the period of a RO is simulated as a normal distribution with mean and standard deviation as per Section 3.2. The simulation does not take global noise into account.

4.1.1. Ideal Clock

For the ideal clock simulations, the noise source is simulated in software according to Algorithm 1. The algorithm describes the simulation for a single measurement. This simulation is then repeated 100,000 times to generate the samples for a single experiment. These experiments are repeated for different ideal sampling times T_i and RO periods μ_{ro}. For the jitter of the RO, a relative value of 2% of the period is taken, which seemed reasonable based on preliminary tests on the FPGAs. The simulated periods are 3, 4 and 5ns, which should correspond well to the periods on the FPGA.

Algorithm 1 Simulation of a single measurement for the ideal model.

```
$t \leftarrow \mathcal{N}(\mu_{ro}, \sigma_{ro})$
$c \leftarrow 0$
while $t \leq T_i$ do
    $t \leftarrow t + \mathcal{N}(\mu_{ro}, \sigma_{ro})$
    $c \leftarrow c + 1$
end while
return $c$
```

Figure 4.1 shows the simulation results for a simulation with $\mu_{ro} = 4ns$ and varying sampling times. The sampling time is chosen such that the distribution follows the best-case scenario for the min-entropy, as can be seen by the even distribution of samples. The distribution of the simulated samples corresponds well to the theoretical distribution. The distribution of samples clearly gets wider as the sampling time increases, as predicted by the model. Plots for the worst-case scenario and different RO periods ($\mu_{ro} = 3ns$ and $\mu_{ro} = 5ns$) can be found in Appendix A.1.2.

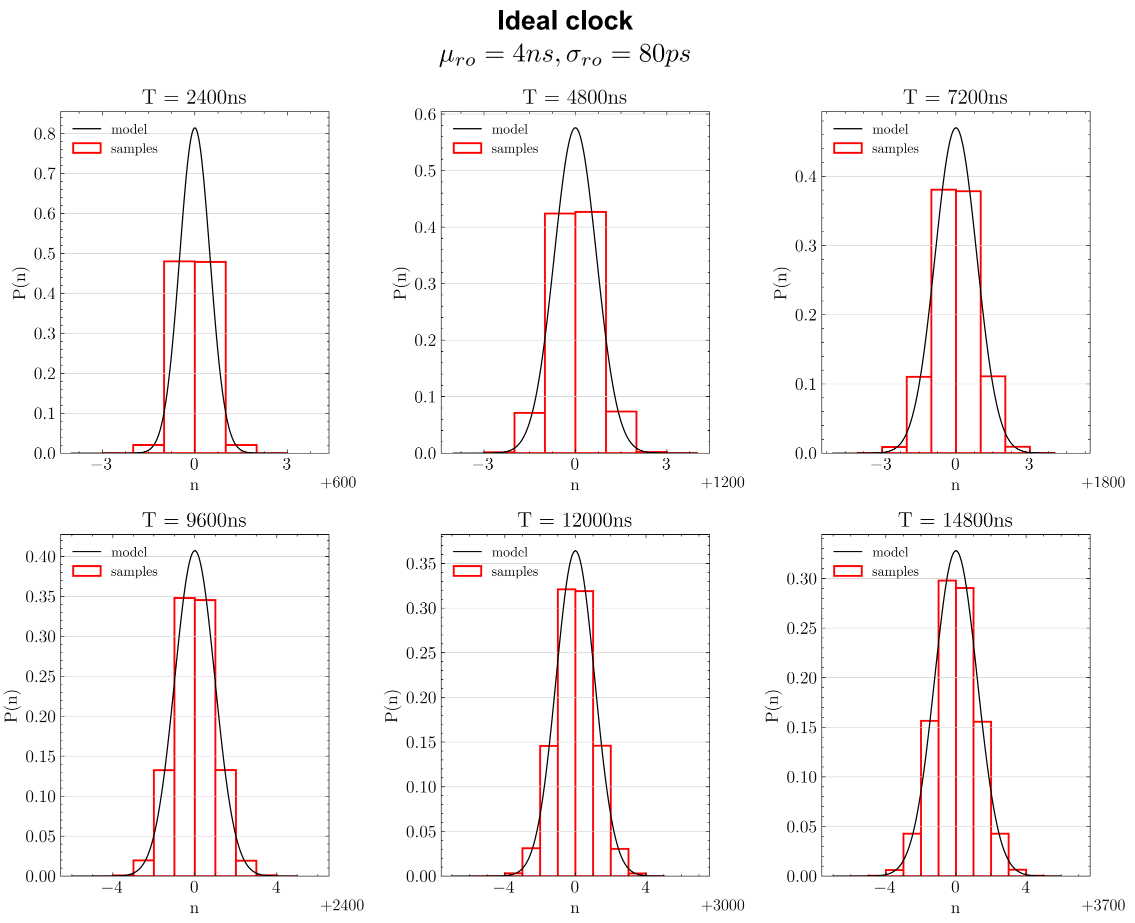

Figure 4.1: Impact of sampling time T on distribution of samples

From these distributions, the sample variance can now be calculated. Remember that the mathematical model predicted a linear increase in variance as the sampling time increased.

Figure 4.2 shows the variance of the simulated samples for the different sampling times, as well as the theoretical variance per Equation 3.14. The predicted linear increase in variance as the sample time increases is clearly visible. The small discrepancy between the simulated and computed variances can be attributed to the precision loss in the integer samples.

Figure 4.3 and 4.4 show the min-entropy depending on the sampling time of simulated samples in the best-case and worst-case scenario for different RO periods. As a reminder (see also Section 3.4), the best-case scenario happens when the sampling time T_i is such that the noise source is sampled on the rising edge of the RO, while the worst-case scenario happens exactly half a RO period later.

The best-case scenario should always yield a min-entropy close to 1, which is confirmed by Figure 4.3. For the worst-case min-entropy, there should be a dependence on both the sampling time and the period and jitter, as per Equation 3.18.

Figure 4.4 indeed shows an increase in min-entropy as the sampling time increases. The plot also shows that for longer RO periods, a longer sampling time is needed to achieve the same min-entropy, while the relative jitter is the same.

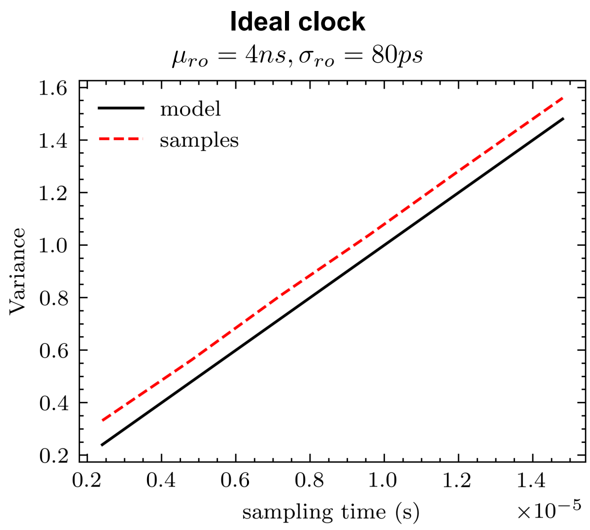

Figure 4.2: Change in variance due to increasing sampling time, theoretical model and simulated samples.

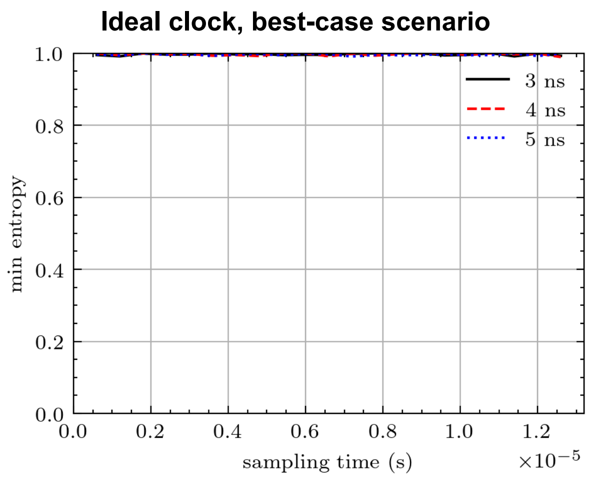

Figure 4.3: Impact of sampling time on min-entropy of the LSB.

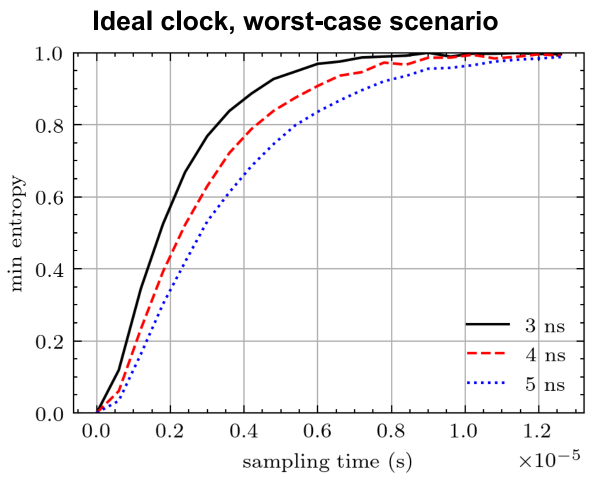

Figure 4.4: Impact of sampling time on min-entropy of the LSB.

For the worst-case scenario, Equation 4.1 allows us to compute the minimal sampling such that the min-entropy is at least the required 0.98, as shown in Section 3.4.4.

$$t = \frac{\mu_{ro}^3}{\sigma_{ro}^2} \tag{4.1}$$

Filling in for $\mu_{ro} = 3ns$ and $\sigma_{ro} = 2\% * \mu_{ro}$ gives $t = 0.75 \times 10^{-5}$. The same can be done for the other periods, giving $t = 1 \times 10^{-5}$ and $t = 1.25 \times 10^{-5}$, for $\mu_{ro} = 4ns$ and $\mu_{ro} = 5ns$ respectively. These calculated sampling times correspond well with the simulation results in Figure 4.4.

Another way to obtain these sampling times lies in the other conclusion that was drawn in Section 3.4.4, namely that the variance of the distribution is 1 when the sampling time is chosen such that the min-entropy is 0.98. Figure 4.2 showed a plot of the variance for $\mu_{ro} = 4ns$. The figure shows that when the variance is 1, the sampling time $t \approx 1 \times 10^{-5}$, which is the same time as calculated before.

Also observe that, assuming jitter is relative to the period, a longer RO period has a negative impact on the required sampling time. This follows directly from Equation 4.1 and is confirmed by the simulations in Figure 4.4.

4.1.2. Non-Ideal Clock

With the results in the previous section in mind, we now take a look at what happens when the RO is sampled with a non-ideal clock. The noise source is simulated in the software according to Algorithm 2, where the difference is that the sampling time T_s is now also a simulated RO, as per Section 3.5. The experiment is repeated for different μ_{ro} and n_2, where σ_{ro} is again assumed to be 2% of μ_{ro}. For the clock RO, the same parameters are used as for the sampled RO.

Algorithm 2 Simulation of a single measurement for the non-ideal model.

```
$t \leftarrow \mathcal{N}(\mu_{ro}, \sigma_{ro})$
$c \leftarrow 0$
$T_s \leftarrow \mathcal{N}(n_2\mu_{cl}, \sqrt{n_2}\sigma_{cl})$
while $t \leq T_s$ do
    $t \leftarrow t + \mathcal{N}(\mu_{ro}, \sigma_{ro})$
    $c \leftarrow c + 1$
end while
return $c$
```

Figure 4.5 shows the simulation results for $\mu_{ro} = \mu_{cl} = 4ns$. The sampling time is once again chosen such that the distribution follows the best-case scenario for the min-entropy. For easier comparison with the ideal clock results, the expected value of the sampling time ($n_2 \times \mu_{cl}$) is used to indicate the plots.

The distributions again correspond well to the theoretical distribution, and the distribution of samples gets wider as the sampling time increases. Compared to the distributions using an ideal clock in Figure 4.1, the non-ideal clock simulations show a wider distribution for all sampling times.

Plots for the worst case scenario and different RO periods ($\mu_{ro} = 3ns$ and $\mu_{ro} = 5ns$) can be found in Appendix A.1.3.

Again, the sample variance can be computed from these simulated samples. The models predict that variance should still increase linearly and approximately double compared to the ideal clock simulations.

Figure 4.6 shows the variance of the simulated samples, as well as the theoretical variance according to Equation 3.40, for increasing sampling times. The variance still increases linearly, and its value has indeed doubled compared to the variance of the ideal clock simulations seen in Figure 4.2. Again, there is a small discrepancy between the theoretical and sample variance, which can be attributed to the precision loss due to integer sample values.

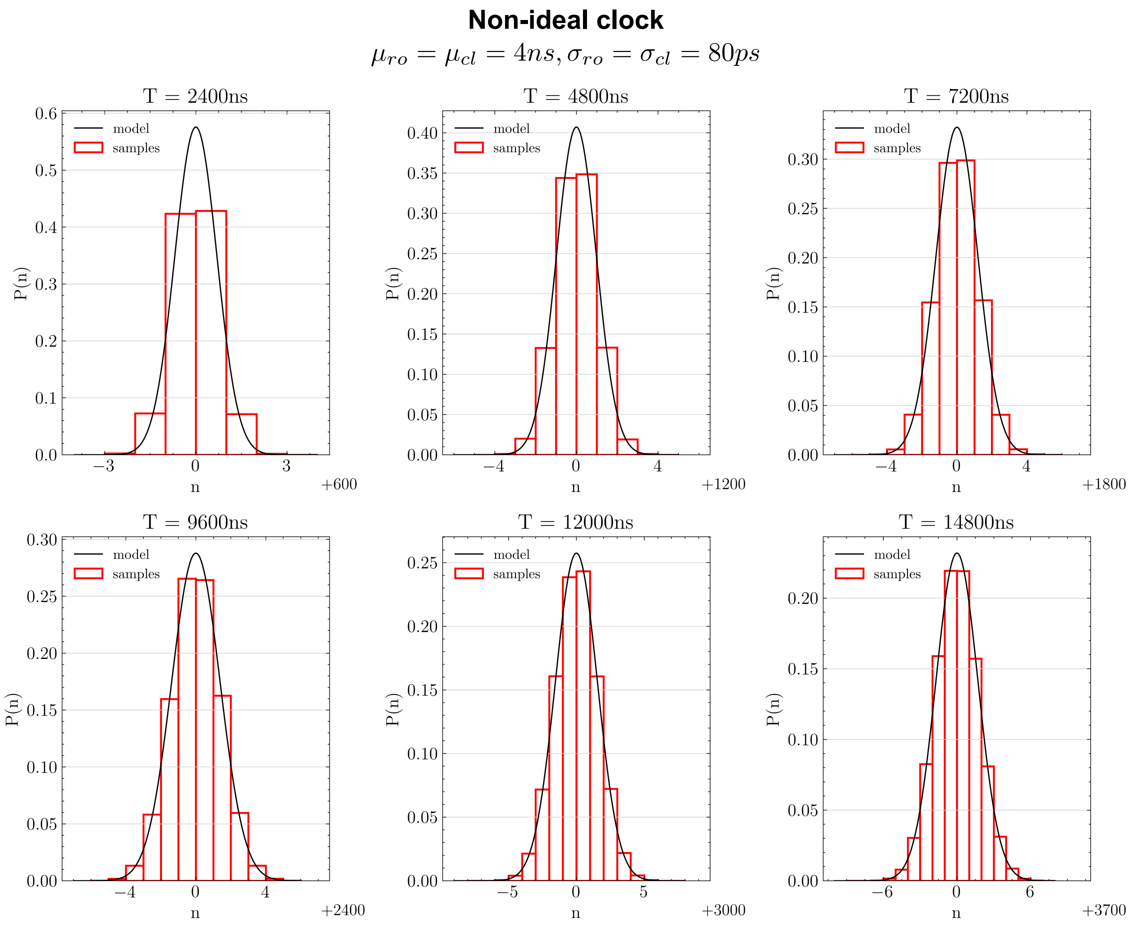

Figure 4.5: Impact of sampling time T on distribution of samples

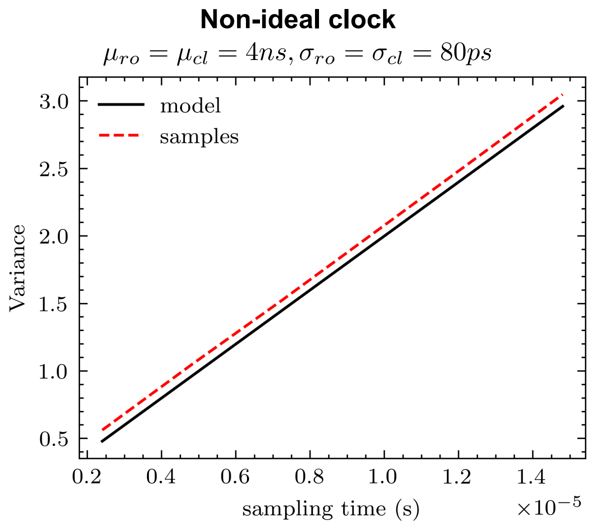

Figure 4.6: Change in variance due to increasing sampling time, theoretical model and simulated samples.

Figure 4.7 and 4.8 show plots of the min-entropy as the sampling time increases for simulated samples in the best-case and worst-case scenario, for different RO periods. Figure 4.7 shows that the best-case scenario still gives a min-entropy close to 1 independent of the sampling time.

For the ideal clock, it was discussed that the minimum sampling time for a min-entropy of at least the required 0.98 could be obtained from the variance plot. Simply take the sampling time when the variance is equal to 1. As the shape of the distribution does not change, the same thing can be applied to the non-ideal clock simulations. Now, as the variance has doubled, this would mean that only half the sampling time of the ideal clock simulations is needed to achieve the same min-entropy.

This is confirmed by comparing the worst-case min-entropies in Figure 4.4 with the ones in Figure 4.8. Note also that the relative distance between the slopes for different periods has not changed. A RO with a shorter period thus still allows for shorter sampling times.

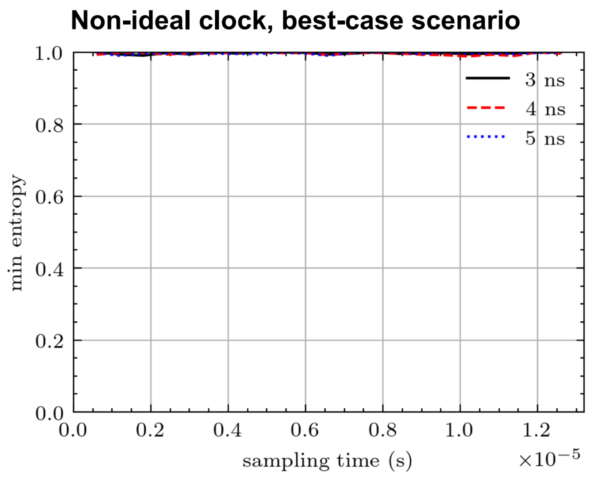

Figure 4.7: Impact of sampling time on min-entropy of the LSB.

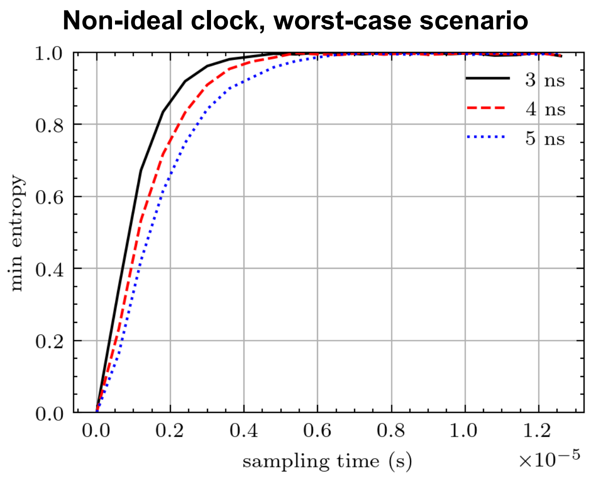

Figure 4.8: Impact of sampling time on min-entropy of the LSB.

4.2. SPICE simulations

The second set of simulations to verify the models is carried out using Cadence Spectre. In Section 4.2.1, simulations are carried out for the ideal clock model. Section 4.2.2 presents simulations for the non-ideal model, comparing them with the ideal clock results.

Spectre is a SPICE-class circuit simulator, allowing for more realistic simulations of the RO. Instead of simulating according to one of the algorithms described before, a circuit netlist is provided to the simulator. An example of such a netlist can be seen in Listing 4.1.

This netlist describes a RO with a leg length of 3 inverters. The simulations are also performed for ROs with leg lengths 5 and 7, to compare the influence of the RO period on the variance. Note that the period of these ROs will be much smaller than those in the Rust simulations and FPGA tests. On top of that, the amount of noise can not be explicitly set to 2% of the period. Instead, a noise bandwidth has to be supplied to the simulator.

The simulations are performed using BSIM models [49] for the transistors. The zero-bias threshold voltage is adjusted slightly for every transistor to simulate process variation that would lead to different RO periods. To ensure that the simulated noise is different for every run, the noise seed parameter of the transient analysis is changed in every experiment. The simulations are repeated for different temperatures (-15, 25 and 80 $^{\circ}C$), different noise bandwidths (5, 50 and 500GHz), and different technology nodes (45nm CMOS, 20nm FinFET).

Listing 4.1: SPICE Netlist

```
.INCLUDE 'p045/p045_cmos_models_tt.inc'
.GLOBAL VDD GND
.PARAM Lmin = 45n
.PARAM Wmin = 45n
.PARAM Wp=90n
Vsupply VDD 0 1
Vground GND 0 0
.MODEL n1 nmos LEVEL=54 vth0=0.40145
.MODEL p1 pmos LEVEL=54 vth0=-0.50567
.MODEL n2 nmos LEVEL=54 vth0=0.38652
.MODEL p2 pmos LEVEL=54 vth0=-0.40200
.MODEL n3 nmos LEVEL=54 vth0=0.50165
.MODEL p3 pmos LEVEL=54 vth0=-0.49321
Mn1 a1 a3 GND GND n1 L=Lmin W=Wmin
Mp1 a1 a3 VDD VDD p1 L=Lmin W=Wp
Mn2 a2 a1 GND GND n2 L=Lmin W=Wmin
Mp2 a2 a1 VDD VDD p2 L=Lmin W=Wp
Mn3 a3 a2 GND GND n3 L=Lmin W=Wmin
Mp3 a3 a2 VDD VDD p3 L=Lmin W=Wp
.IC V(a1)=1
.END
```

4.2.1. Ideal Clock

For the ideal clock simulations, a RO is simply simulated for the desired sampling time. The simulated counter value is then obtained by counting the number of rising edges that occurred before the sampling time. This simulation is then repeated 1000 times to generate the samples for a single experiment. These experiments are then repeated for the different parameters discussed before.

Figure 4.9 shows the simulation results for a RO with leg length 3, using the 45nm CMOS models, for varying sampling times. The shape of the histograms resembles the expected distribution, and the width increases as the sampling time increases.

The sample variance can be computed again from these simulated samples. As with the Rust simulations, the variance is predicted to increase linearly with sample time.

Figure 4.10 shows a plot of the variance of the simulated samples for different temperatures, with a noise bandwidth of 500GHz. The variance clearly increases linearly with the sample time for all temperatures. There is, however, a large difference in slope for different temperatures.

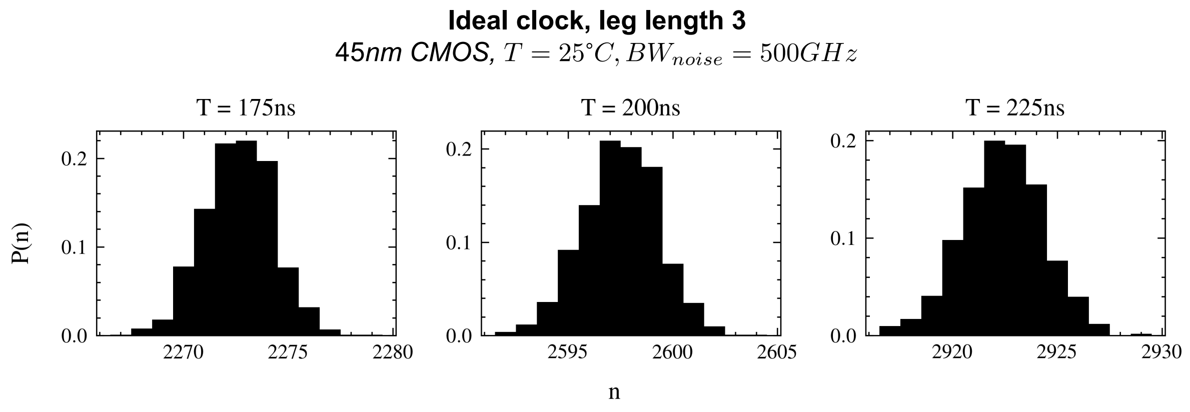

Figure 4.9: Distribution of simulated samples for various sampling times.

Figure 4.11 shows that when changing the noise bandwidth, the variance still increases linearly in time. Again, different bandwidths give different slopes.

The relatively large change in variance due to temperature and noise bandwidth warrants caution when designing TRNGs based on this noise source. The sampling time should be chosen such that the variance is at least 1 for the highest operating temperature, and a proper value for the noise bandwidth should be obtained from physical measurements.

No plots for the min-entropy of the SPICE simulations are provided, as there is no clear best- or worst-case situation. Combined with the relatively small sample size, they would provide no information.

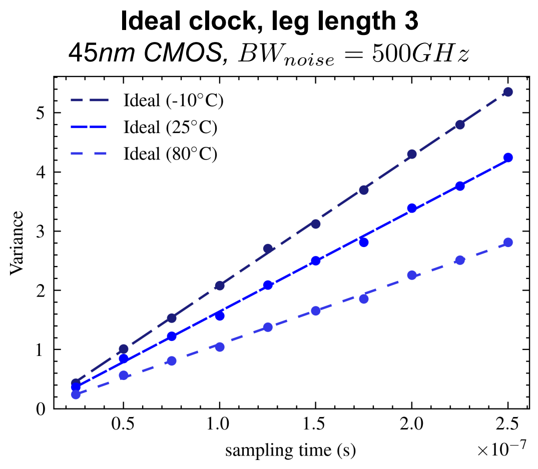

Figure 4.10: Impact of sampling time and temperature on variance.

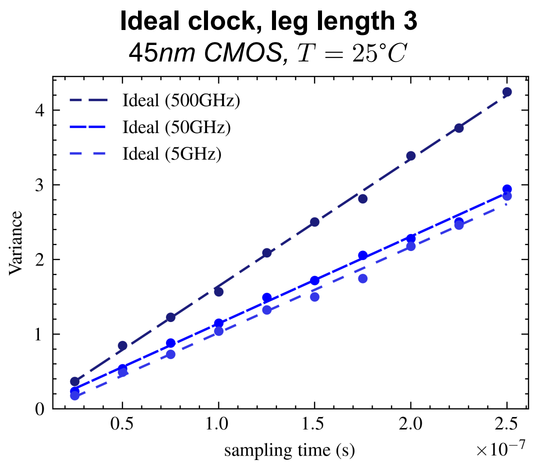

Figure 4.11: Impact of sampling time and noise bandwidth on variance.

Figures 4.12 and 4.13 show plots of the variance for different temperatures for ROs with leg lengths of 5 and 7 inverters respectively. Comparing the variances to those in Figure 4.10 shows that a longer period due to a longer leg leads to less variance for equal sampling time. This is the same result as obtained with the Rust simulations.

The simulations thus far have been based on transistor models of 45nm CMOS technology. The FPGA that will be used for the hardware validation, however, is based on 20nm FinFET technology. Simulations are also carried out using a FinFET transistor model to see the impact of using this different technology node.

Figure 4.14 shows plots of the variance for different temperatures for a RO with leg length 3, using the 20nm FinFET model. Compared to the plots of the variance for the CMOS simulation, there is a clear reduction in variance, which is to be expected from a newer technology node. Temperature dependence still exists and seems to have increased, especially for low temperatures.

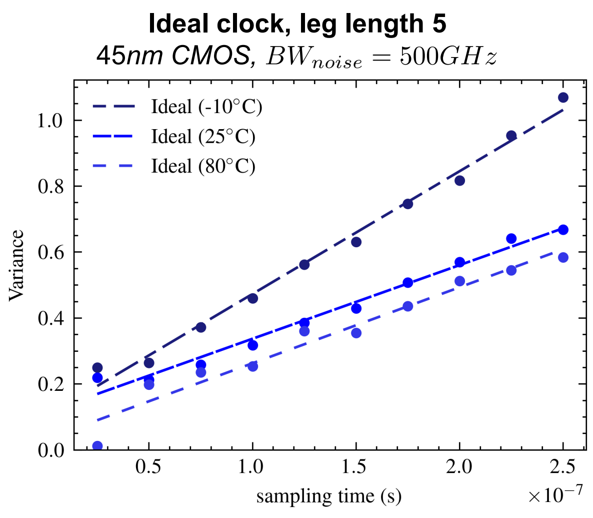

Figure 4.12: Impact of sampling time and temperature on variance.

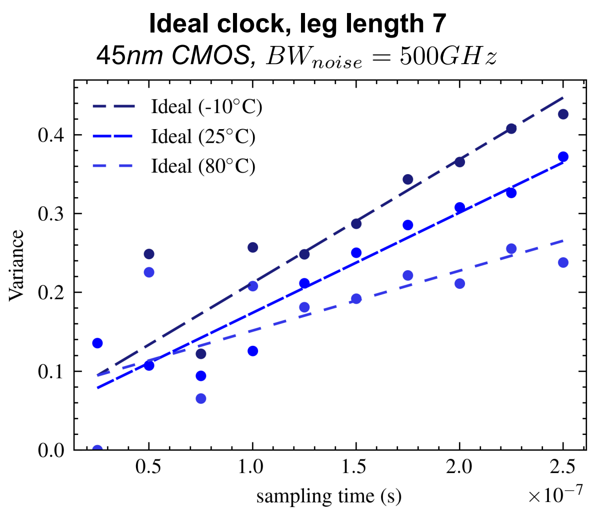

Figure 4.13: Impact of sampling time and temperature on variance.

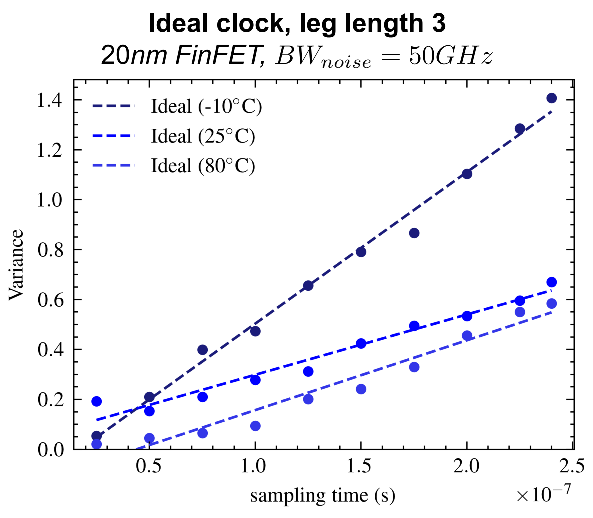

Figure 4.14: Impact of sampling time and temperature on variance.

4.2.2. Non-Ideal Clock

To simulate sampling the counter with a non-ideal clock, such as a second RO, a second set of 1000 simulations is done to simulate the sampling times. For this set, the average period of the ROs is determined. Using this average period, the 'clock divider' value n_2 is calculated. The sampling times are then extracted from the simulations as the time when rising edge n_2 occurred.

After simulating this set of sampling times, the counter values can be obtained in almost the same way as with the ideal clock simulations. Instead of using the same (ideal) sampling time, every ideal clock simulation is paired with a sampling time simulation. Then, the rising edges in this sampling time are counted.

Figure 4.15 shows the simulation results for the RO with leg length 3 from Figure 4.9, now sampled with another RO. Again, the expected value of the sampling times indicates the plots, which facilitates comparison with the ideal clock simulations.

The shape of the histograms resembles the expected distribution. Compared to Figure 4.9, the width of the distribution has increased. This is the expected behaviour, which was also seen in the Rust simulations.

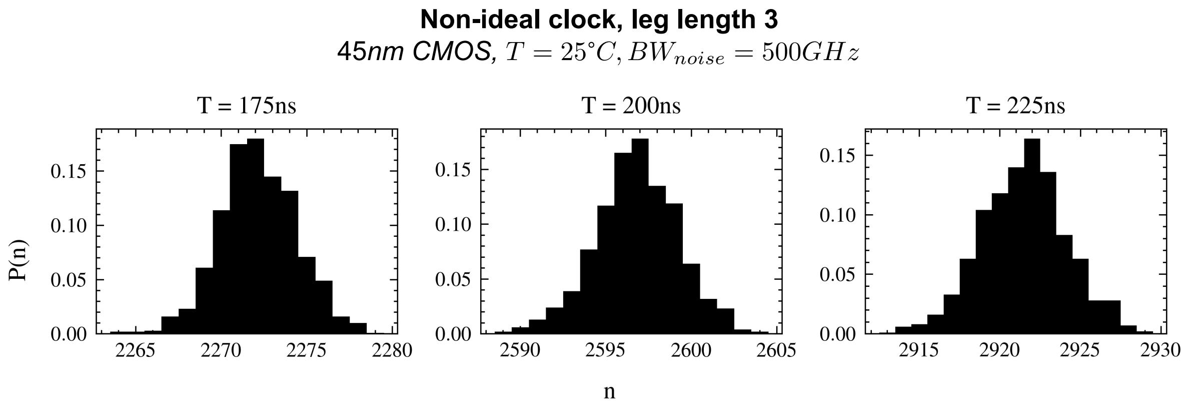

Figure 4.15: Distribution of simulated samples for various sampling times.

Figures 4.16 and 4.17 show the variance for the non-ideal clock simulations, for a RO with leg length 3 using the 45nm CMOS models. Figure 4.16 shows the temperature dependence for a given noise bandwidth ($500GHz$), while Figure 4.17 shows the noise bandwidth dependence for a given temperature ($25°C$) For reference, the ideal clock simulation results are also included.

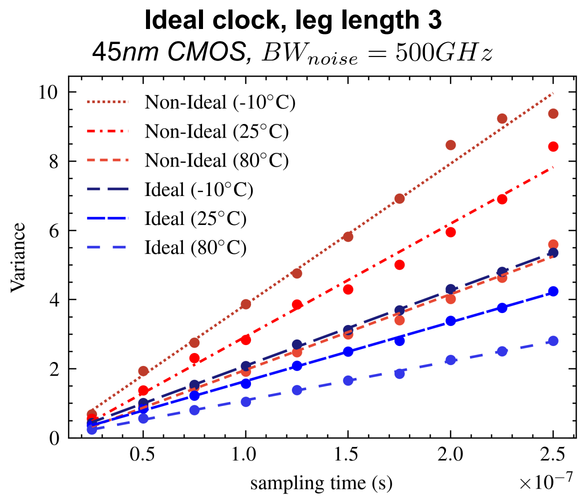

Figure 4.16: Impact of sampling time and temperature on variance.

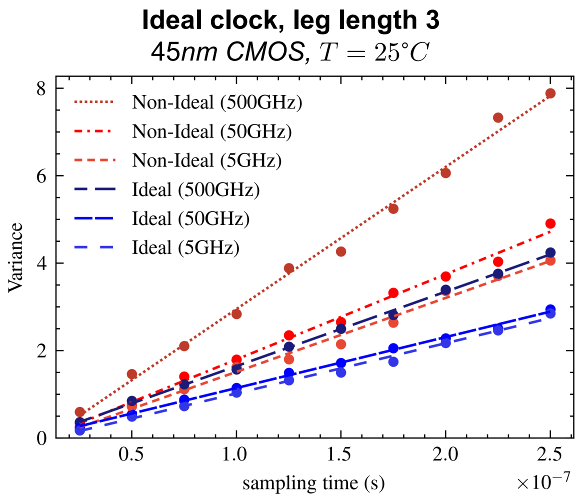

Figure 4.17: Impact of sampling time and noise bandwidth on variance.

As predicted by the mathematical model from the previous chapter and previously confirmed by the Rust simulation, the variance approximately doubles when sampling the RO with another RO. This increase exists for all simulated temperatures (Figure 4.16) and noise bandwidths (Figure 4.17).

When the period of the ROs in the ideal simulation was increased by adding more inverters, the variance decreased. Figures 4.18 and 4.19 show plots of the variance for different temperatures for ROs with leg lengths 5 and 7, sampled with a second RO. Compared to Figure 4.16, the variance has decreased, but compared to the ideal clock simulations, the variance has again doubled.

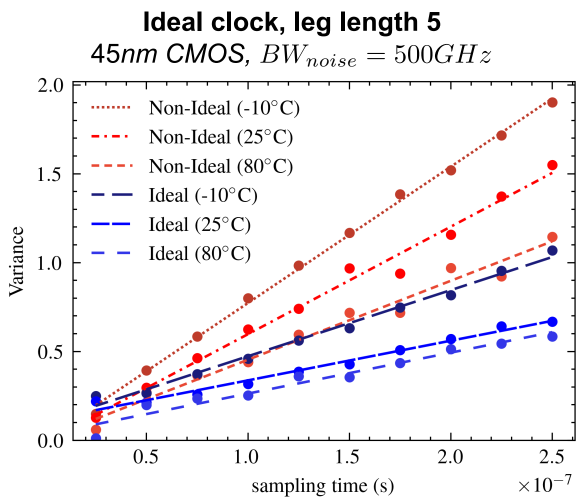

Figure 4.18: Impact of sampling time and temperature on variance.

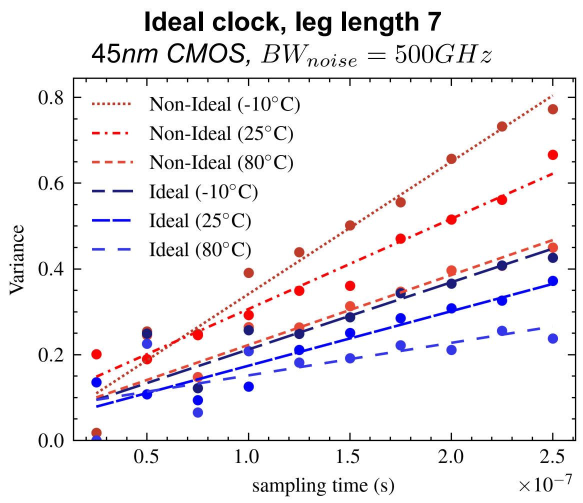

Figure 4.19: Impact of sampling time and temperature on variance.

Lastly, the non-ideal clock simulations are performed for the 20nm FinFET technology node, as this is the technology used in the FPGA for hardware validation. When analysing the ideal clock simulations, it was already discussed that the variance was smaller compared to the CMOS simulations.

Figure 4.20 shows plots of the variance for different temperatures for a RO with leg length 3, with the 20nm FinFET model. Here as well, using a second RO to generate the sampling clock improves the variance across all temperatures and sampling times.

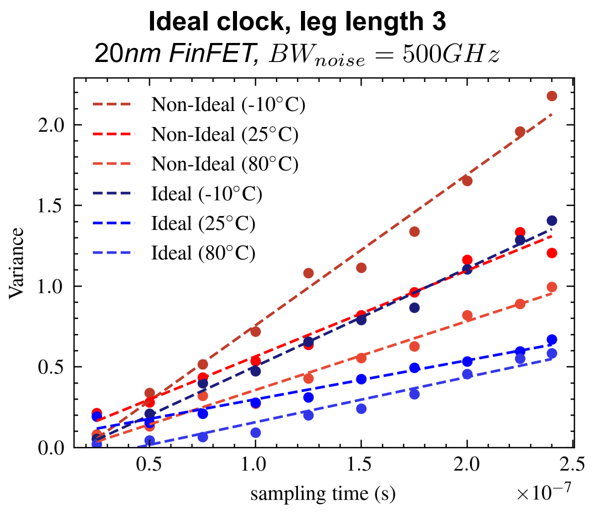

Figure 4.20: Variance of simulated samples as a function of the sample time for different temperatures, leg length 3, $BW_{noise} = 50GHz$, 20nm FinFET

4.3. Conclusion

In this chapter, two sets of software simulations were used to verify the mathematical findings and conclusions from the previous chapter. First, a simple Rust program was used that simulated the period and jitter of a RO as a normal distribution. The simulation results for both the ideal and non-ideal sample clock corresponded well to the stochastic models.

The calculated minimum sampling time for a worst-case min-entropy of 0.98, required by AIS 20/31 [13], was verified. It was shown that this sampling time corresponds to a variance in the counter of 1. The non-ideal sample clock simulations confirmed that using a second RO to sample the first doubles the variance.

Then, SPICE simulations were performed using Cadence Spectre. The simulations again showed results that were predicted by the stochastic models from Chapter 3 and previously verified in the Rust simulations. Using a noisy sample clock approximately doubled the variance of the noise source for all the tested parameters.

In both simulations, reducing the period of the RO increased the variance for a given sampling time, as predicted by the model. This means that using fewer inverters and, thus, less area improves the performance of the noise source.

5 Validation

In this chapter, the mathematical models and conclusions are validated in hardware using the Intel Arria 10 FPGA. Section 5.1 starts with an overview of the tests that will be conducted, including the test setup in Section 5.1.1. The results of these tests are presented in Section 5.2 and then analysed in Section 5.3. A discussion on the test results and their relation with the stochastic model and the simulation results is presented in Section 5.4. Section 5.5 concludes this chapter.

5.1. Test Overview

This section first gives an overview of the test setup in Section 5.1.1. It introduces the used hardware and software components and provides an overview of the hardware implementation of the noise sources. Then, Section 5.1.2 gives an overview of the different experiments that will be conducted to validate the mathematical model in hardware.

5.1.1. Test Setup

The setup consists of an Intel Arria 10 FPGA, on which the noise sources are implemented, a PC that handles the collection of samples and communication with the FPGA, and a function generator to generate a very low jitter clock. Figure 5.1 shows their connections.

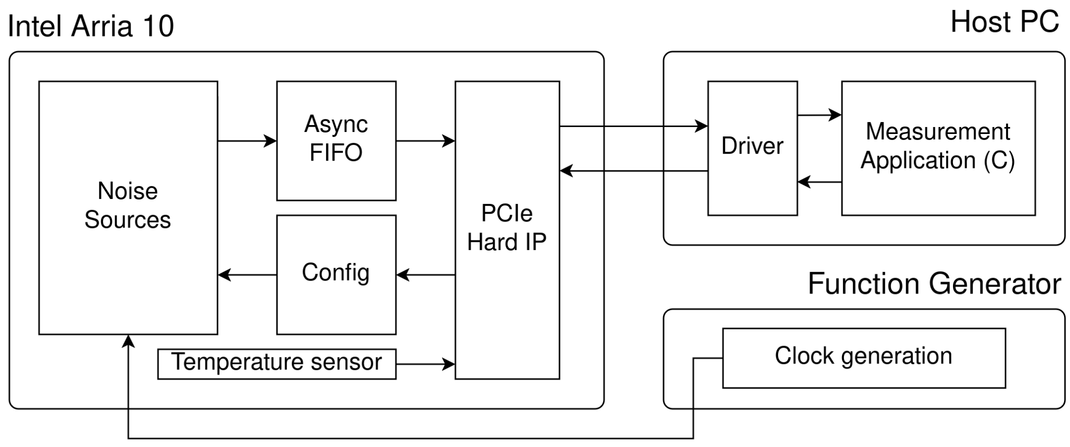

Figure 5.1: Top-level overview of the test setup.

The Arria 10 FPGA contains the noise sources, including sampling circuitry, which outputs random numbers to an asynchronous FIFO to buffer them. A peripheral component interconnect express (PCIe) IP block handles communication with the host PC. A configuration block is included to select different parameters, such as the sampling clock, from the host PC without needing to reprogram the device.

The function generator is used to validate the stochastic models, as an ‘ideal’ design and a non-ideal design are needed. The latter is easy to realise, while a sampling period with zero jitter is necessary for the ideal clock model. This is impossible in practice, but a jitter of $\sim 10ps$ can be achieved by testing with a function generator. This is much smaller than that of the ROs and will therefore suffice. It also allows for comparing a sample clock with known parameters and a sample clock with unknown parameters, such as a second RO.

The internal clock of the FPGA, a phase-locked loop (PLL), will also be tested and compared to the function generator. If the noise source is to be implemented in a TRNG design, it should be able to function without an external clock generator.

Figure 5.2 shows the implementation of the noise sources block from Figure 5.1. It consists of 6 different sample clocks, of which one is selected to generate the sample signal. This signal samples six different noise sources, of which one is fed to the output. The *sample clock select*, *noise source select* and N_2 signals are all configurable from the host PC.

Next to the already mentioned advantage of not needing to reprogram the device, this also ensures the noise source’s physical parameters stay the same when testing different sample clocks.

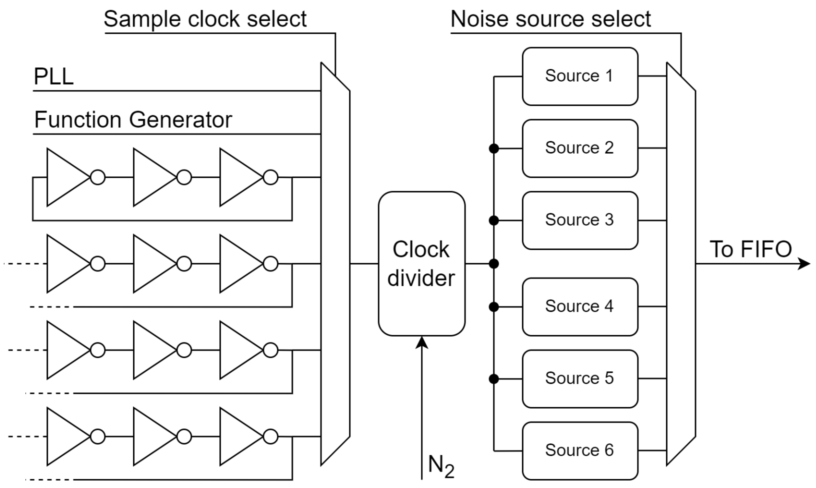

Figure 5.2: Overview of the noise sources with sample clocks.

The noise source design is slightly more complicated than the one presented in Chapter 3. It can be seen in Figure 5.3 and consists of a noisy clock (a RO with three inverters in this example), two ripple counters, and a flip-flop to sample the counters. The second counter is added so that the counters can be alternated. This ensures that the ripple counter has settled before sampling, removing noise due to metastability, which would make proper analysis of the jitter noise more complicated.

Both ripple counters are 32-bit. In Section 3.4, it was discussed that only the three least significant bits have entropy and that using a wider counter, therefore, does not improve the TRNG. When validating the model, however, the calculations are more straightforward when the counters do not overflow.

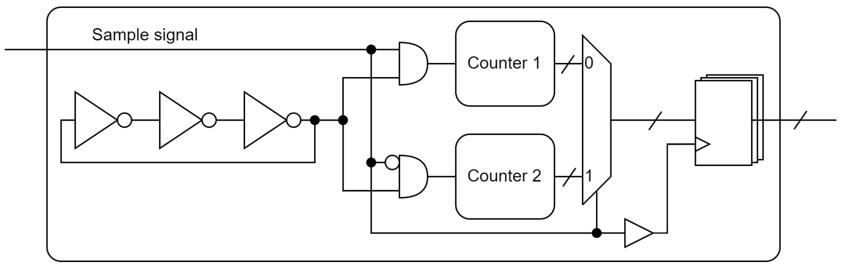

Figure 5.3: Design of one noise source.

5.1.2. Test Plan

Here, the proposed tests are described. First, some general parameters are discussed, such as the length of the ROs. Then, three sets of experiments are described.

For the sample clocks, the PLL, function generator, and four different ROs will be tested. For the noise clocks, also four (different from those used for the sample clocks) ROs are used. The other two noise sources use the PLL and function generator as noise clocks. This allows for, among others, sampling of the PLL with a RO, which should give almost the same entropy as the other way around, according to the mathematical model.

In Chapters 3 and 4 it was shown that using ROs with shorter periods gives better results. For this reason, the length of the ROs is also chosen very short in hardware. However, preliminary tests showed that ROs with a leg length of only 1 or 3 inverters oscillated too quickly for the logic inside the FPGA. This led to stability issues in the counter and sample circuit, which in turn led to a lot of artefacts in the measurement results. When implementing the noise source in a TRNG, this is not necessarily a problem, as it simply makes the output less predictable. For this thesis, however, those artefacts make proper analysis of the results impossible, and they are therefore not used.

Thus, ROs with leg lengths of 5, 7, 9 and 11 inverters are used for both the noise RO and the sample RO.

In some experiments, the sampling time T will be chosen to be much larger than what is needed for a min-entropy of at least 0.98. This would reduce the efficiency of the TRNG. The counter, however, has only integer precision, making calculations more precise for higher counter values as this reduces the relative error.

To check that results are reproducible, the test design discussed in Section 5.1.1 is synthesised twice using different seed values. Then, both designs are tested on two different Intel Arria 10 FPGAs.

Ideal Clock

For the first set of measurements, the function generator is used to generate a sampling signal close to what would be considered ideal. With this sampling signal, the four ROs with leg lengths 5, 7, 9 and 11 are sampled for different sampling times.

The sampling times are taken from $5\mu s$ to $50\mu s$, with $5\mu s$ intervals. For each of these sampling times, 500,000 samples are taken.

With these results, it should be possible to confirm the variance in the counter is linear with sampling time and that the variance decreases when using ROs with longer periods.

PLL

For the second set of measurements, the same tests as above are carried out. The only difference is that instead of the function generator, the PLL is used to generate the sampling signal. This test is purely carried out to check the performance of the PLL. It should behave as an ideal clock, but very little is known about its actual performance.

Non-Ideal Clock

For the third and final set of measurements, ROs are used to sample the noise source. Every noise RO is sampled using a sample RO of the same leg length.

Again, the sampling times are taken from $5\mu s$ to $50\mu s$, with $5\mu s$ intervals. For each of these sampling times, 500,000 samples are taken.

The goal is to confirm the increase in variance due to sampling with a non-ideal clock by comparing the results from this set of measurements to those obtained in the first set of ideal clock measurements.

5.2. Results

In this section, some of the more interesting results of the tests described in the test plan are presented. All results are included in Appendix B. An analysis of the results is presented in Section 5.3.

5.2.1. Ideal Clock

Figure 5.4 shows histograms of some of the experiments with a RO with leg length 7, sampled with the function generator. These histograms correspond well to the expected distribution of samples discussed so far. It is clear that the variance of the data increases as the sample time increases.

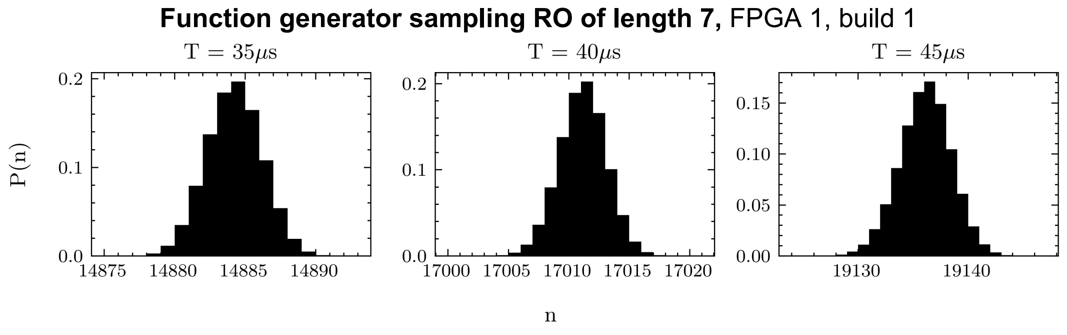

Figure 5.4: Distribution of measured samples for various sampling times.

Figure 5.5 shows histograms of a RO with leg length 11, showing that a longer leg length, implying a longer period, negatively impacts the variance.

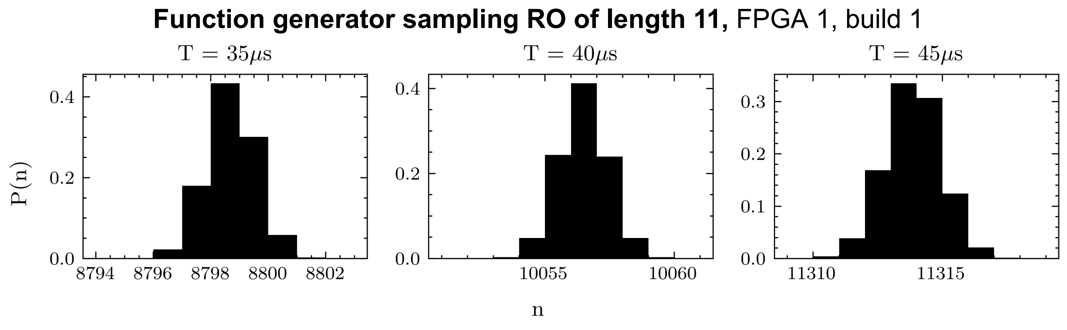

Figure 5.5: Distribution of measured samples for various sampling times.

Figure 5.6 shows histograms that do not correspond to the expected distribution, although the individual peaks resemble the predicted distribution. This behaviour seems to be inherent to a specific build, as it is replicated on a different FPGA (Figure B.12), while different builds do show different behaviour (Figure B.3). Some possible reasons for this behaviour were investigated, yet no satisfactory explanation was found.

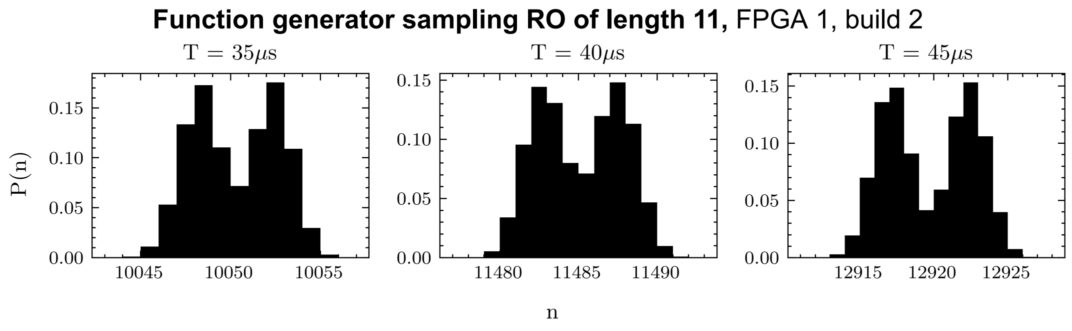

Figure 5.6: Distribution of measured samples for various sampling times.

5.2.2. PLL

Figure 5.7 shows results for measurements with the PLL. The double peaks visible with some function generator measurements are now present in nearly all measurements (see also Appendix B.2).

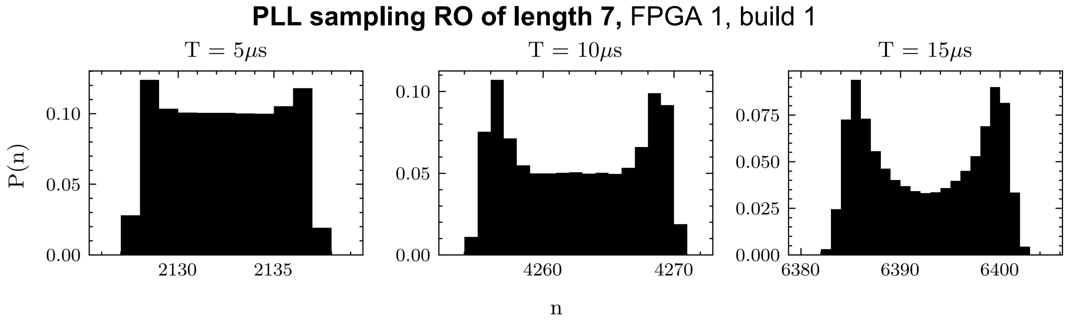

Figure 5.7: Distribution of measured samples for various sampling times.

Using the PLL as noise source and sampling with the function generator, Figure 5.8 and Figure B.16, shows that this time however it seems to be a problem in the PLL. Further analysis is presented in Section 5.3.2.

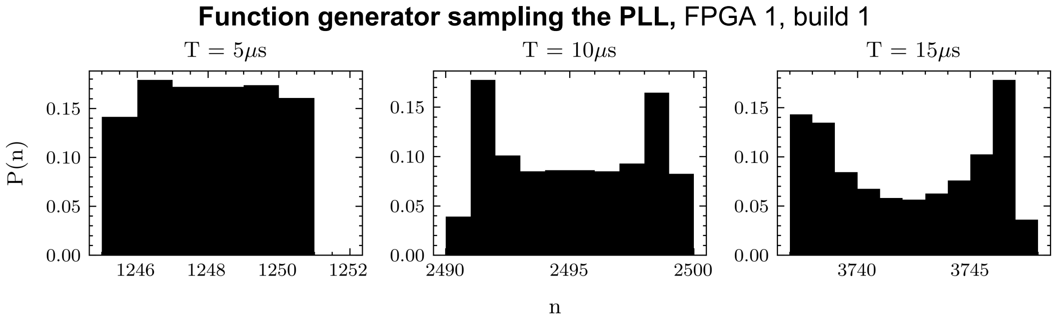

Figure 5.8: Distribution of measured samples for various sampling times.

5.2.3. Non-Ideal Clock

Figure 5.9 shows histograms of some of the experiments with a RO with leg length 7, sampled with another RO of the same length. Again, these histograms seem to correspond well to the expected distribution of samples, and the variance of the data increases as the sample time increases.

Compared to the measurement results with the function generator in Figure 5.4, there definitely seems to be some increase in the width of the distribution. Further analysis is presented in Section 5.3.3.

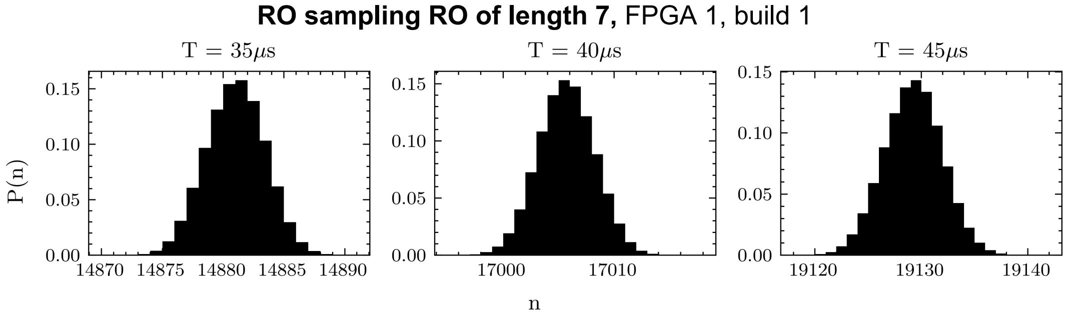

Figure 5.9: Distribution of measured samples for various sampling times.

5.3. Analysis

In this section, the obtained results are analysed. First, the different sets of measurements, as presented in the test plan, are analysed separately in Section 5.3.1 through Section 5.3.3. Then, in Section 5.3.4, a comparison is made between the results of the ideal and non-ideal clock measurements.

Note that although measurements with a RO with five inverters were carried out, its period, in the end, turned out to also be too short to clock the asynchronous design reliably. This led to artefacts in the measurement results, which is why they are not included in the analysis presented here.

5.3.1. Ideal Clock

The first set of measurements tested the different noise RO using the function generator as the sample clock. The goal was to confirm a linear relation between the sampling time and variance, as well as a reduction in variance for longer RO periods. Again, only the ROs with leg lengths 7, 9 and 11 are included in the analysis.

The first step in analyzing the results is to calculate the average periods of the ROs from the measurements. Table 5.1 lists the periods for both builds on FPGA 1. The periods for the same builds on the other FPGA are almost equal to these, again indicating the results are influenced more by different builds than by using a different FPGA.

The ROs with leg lengths 9 and 11 have periods corresponding to those utilised in the Rust simulations, facilitating a comparison between the measurement and simulation outcomes.

Table 5.1: Estimated periods of the sampled ROs on FPGA 1

Leg length	Build 1	Build 2
7	$2.35ns$	$2.71ns$
9	$3.02ns$	$2.98ns$
11	$3.98ns$	$3.48ns$

In Figure 5.4, measurement results from sampling a RO of length 7 with the function generator were shown for some sampling times. More distributions can be seen in Appendix B.1. Visually, it is clear that the variance increases for longer sampling times and decreases for ROs with longer periods.

Figure 5.10 shows the impact of sampling time on the variance for the three different ROs. Where the model predicted, and the simulations showed, a linear increase in variance, the increase in this plot is clearly superlinear.

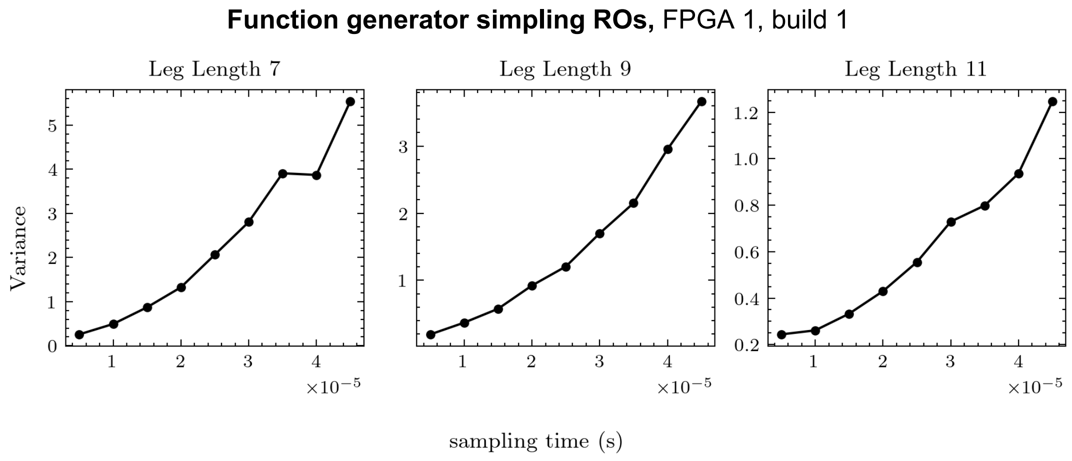

Figure 5.10: Impact of sampling time on variance for different leg lengths.

One explanation for this is that this variance is not instantaneous, sample-to-sample variance but the variance over the duration of the whole experiment.

The largest sampling time taken for the measurements was $50\mu s$. This means the total time to conduct an experiment of 500,000 samples can take up to 25 seconds. As discussed in Section 2.3.1, the period of a RO is sensitive to changes in temperature and voltage. Although the FPGA is cooled, the logic blocks that implement the RO are bound to have some temperature change over this period, meaning the period of the RO changes as well. As the voltage will never be completely stable over time either, this time dependence must be considered when calculating the samples' variance.

To see the effect of the time dependency, the data is split into segments of 10 milliseconds. Figure 5.11 shows the histograms of these segments.

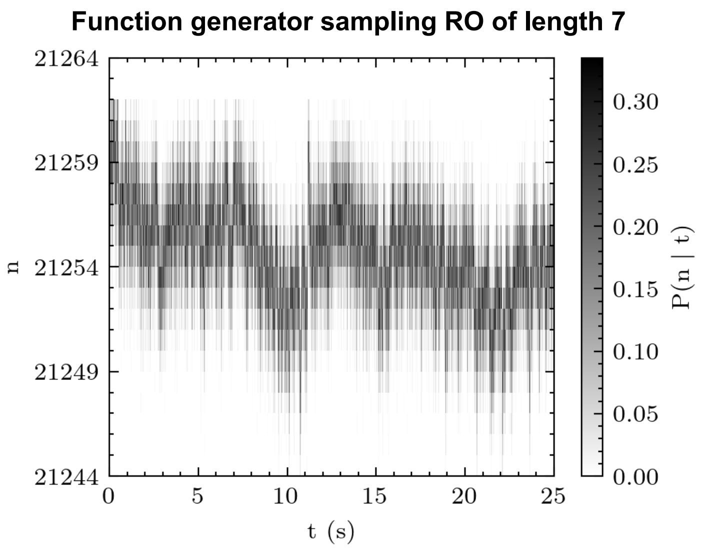

Figure 5.11: Impact of total experiment time on distribution of samples.

The average value of the samples changes over time. This means that a portion of the variance of the data is not due to jitter but due to fluctuations in the period of the RO. Also, note that this change is minimal: the average counter value varies between 21259 and 21250, indicating a change in the period of only 0.04%.

Fully correcting the variance in Figure 5.10 for this time dependence is non-trivial. The segment sizes should not be too small to make a proper variance estimate, but they can also not be as large as the period would not be stable again. Figure Figure 5.12 shows the raw and corrected variance, where segments, or slices, of 200 samples are used. Although still slightly superlinear, this variance more closely resembles that of the model.

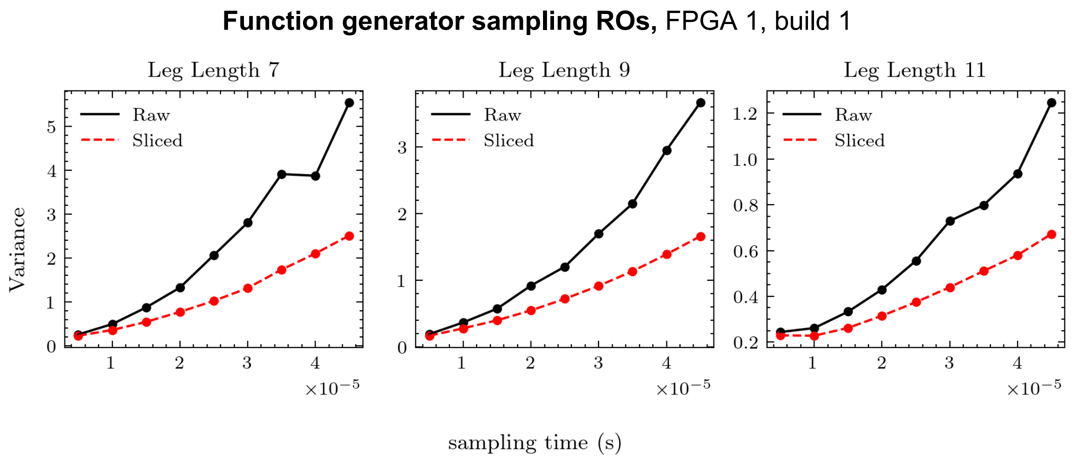

Figure 5.12: Reduction in variance due to applied correction for different leg lenghts.

Comparing this corrected variance of the RO with leg length 11, which has a period of approximately $4ns$, to the variance found in simulations with 2% jitter (Figure 4.2), the sampling time is almost 7 times as large before the same variance is obtained. This would mean the jitter in the hardware implementation is only 0.8% of the period. Comparing the RO with a leg length of 9 inverters to the Rust simulations for a $3ns$ period gives a jitter of approximately 1%, confirming that the initial estimation used for the Rust simulations was too high.

5.3.2. PLL

The second set of measurements tested the performance of the PLL, to see whether it would fit the ideal clock model. This is important information when implementing the noise source in a TRNG design. The only other option is to keep using the external function generator, which is very impractical to say the least.

As discussed when presenting the measurements results in Section 5.2, the measurements with the PLL showed some odd results compared to those performed using the function generator. An example of such a measurement is shown in Figure 5.13. There is a clear bimodal distribution, which looks like a convolution of the expected distribution with some sine wave.

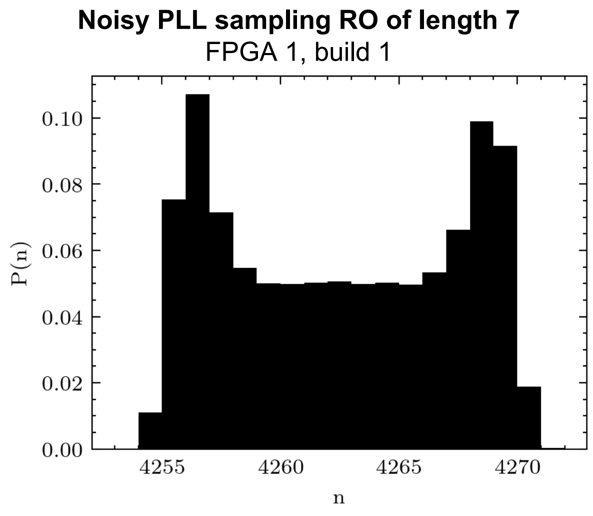

Figure 5.13: Impact of spread spectrum clocking on measurements results.

It turns out that this is almost exactly what happened. The PLL clock was taken from the PCIe IP block, which uses the PCIe reference clock from the PC motherboard as the reference for the PLL. This does not need to constitute a problem, as long as this reference clock is a nice, stable clock. However, the PCIe reference clock uses spread spectrum clocking. With spread spectrum clocking, the PCIe clock is frequency modulated with a triangle wave with a frequency between $30 - 33kHz$. This reduces the amount of electromagnetic Interference (EMI) associated with the signal's fundamental frequency [50].

This would mean that the bimodal distribution thus consists of a convolution of the expected distribution with this triangle wave. Figure 5.14 shows the frequency spectrum of the samples in Figure 5.13, obtained by applying the fast Fourier transform (FFT). The peak in the spectrum occurs at $31.3kHz$, which implies that spread spectrum clocking is indeed the problem.

To confirm these suspicions, the measurement of Figure 5.13 is repeated with a PLL using an onboard oscillator as the reference clock. The results of this measurement can be seen in Figure 5.15, and its spectrum in Figure 5.16. The histogram now resembles the expected distribution and the frequency at $31.3kHz$ has disappeared.

The results of these last measurements show that it should be possible to use the PLL as the sampling clock, when using a proper reference. They also show the importance of developing a stochastic model before testing. The variance of the noisy measurement results in Figure 5.13 is clearly larger than the variance of the results in Figure 5.15. This would most likely mean that the first results perform better in statistical tests, although it is shown that this variance is coming from a deterministic signal.

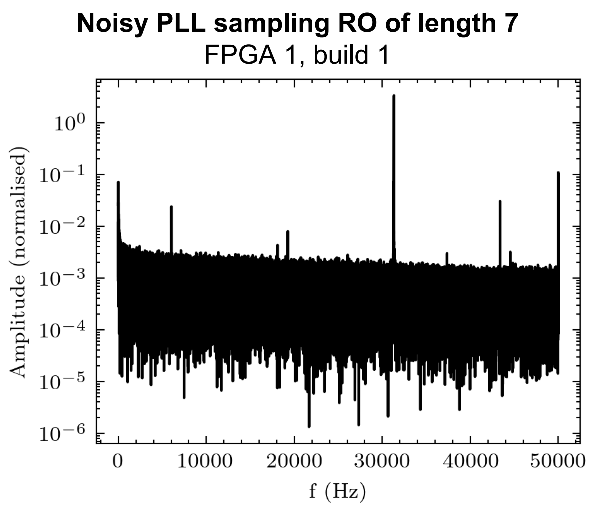

Figure 5.14: Frequency spectrum of the samples in Figure 5.13.

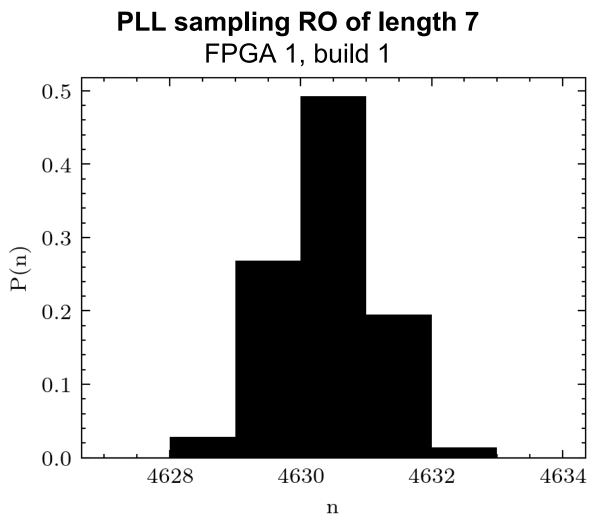

Figure 5.15: Measurement results that show expected distribution after properly implementing the PLL.

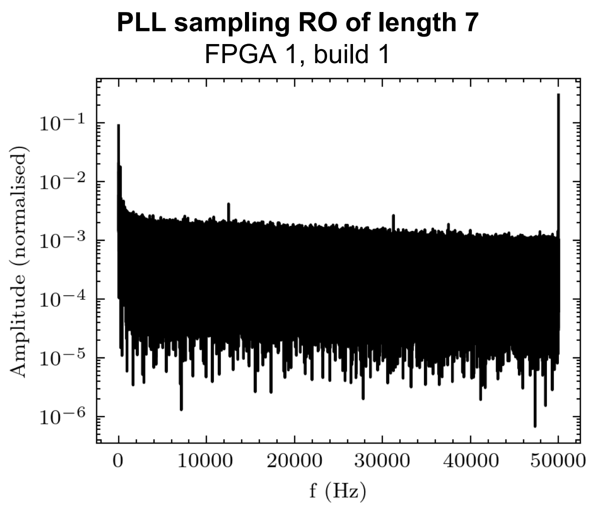

Figure 5.16: Frequency spectrum of the samples in Figure 5.15

5.3.3. Non-Ideal Clock

The third and final set of measurement aimed to confirm the increase in variance due to sampling with a second RO. To this end the same ROs that were sampled using the function generator are now also sampled with a RO.

As discussed in the test plan, the value for the clock divider n_2 should be such that the mean sampling time for the non-ideal clock matches that of the ideal clock measurements. Only then can a fair comparison between the two measurements be made.

To this end, the periods of the sample ROs are determined. They can be found in Table 5.2 for FPGA 1. There is quite some difference in the periods, also compared to those of the noise ROs. In particular, the sample RO with length 11 has wildly different periods between builds. The periods of the sample ROs in FPGA 2 on the other hand are again comparable to the ones presented here for FPGA 1. This confirms that the impact of the different builds is larger than the impact of using a different FPGA.

Table 5.2: Estimated periods of the sample clock ROs on FPGA 1

Leg length	Build 1	Build 2
7	$2.55ns$	$2.43ns$
9	$3.07ns$	$3.03ns$
11	$4.61ns$	$3.56ns$

Figure 5.17 shows the variance of the measurement results of sampling a RO with a second RO, for different leg lengths. The raw variance is again superlinear in time, which is why the same correction as discussed for the function generator measurements is applied. This corrected variance can be seen in the same figure, and shows a more linear trend.

Compared to Figure 5.12 the reduction in variance due to the correction has increased. In the ideal clock measurements, the variance was approximately halved for the longest sampling times, while in these measurements, only a third of the variance remains. A possible explanation for this is that now, not only the noise clock is influenced by global effects such as temperature and voltage changes, but also the sampling clock.

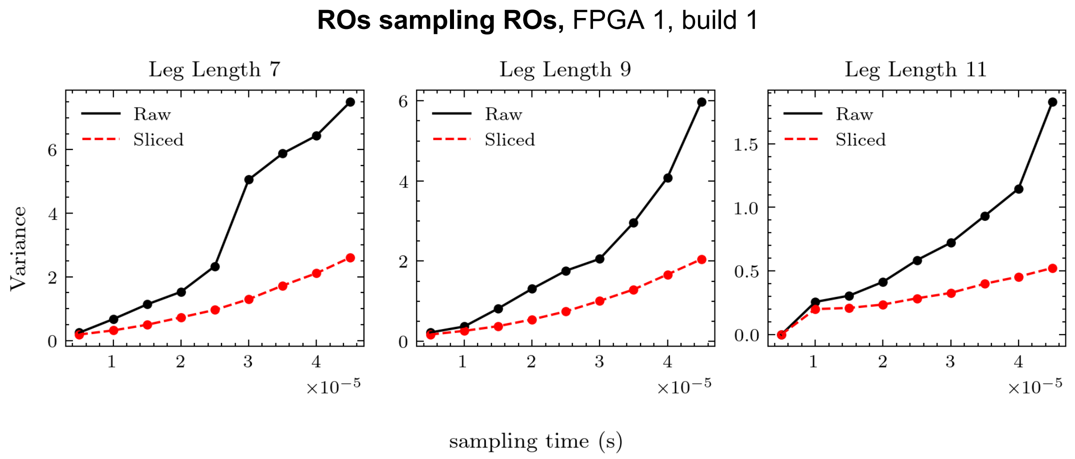

Figure 5.17: Impact of sampling time on raw and corrected variance for different leg lengths.

The measurement results do still show that the variance of the ROs decreases as the leg length, and thus the period, increases. This again confirms the result that better performance is achieved for smaller ROs.

5.3.4. Comparison Ideal and Non-Ideal Clock

After analysing the different measurement results using the function generator, the PLL and different ROs as sample clocks, it is time to compare the results to see if they validate the stochastic model. The assumption is still that the function generator approximates the ideal clock, and that sampling with a RO should double the variance.

A comparison between the raw variances of the measurement results obtained with the function generator and ROs for different leg lengths can be seen in Figure 5.18. The variance of the measurements sampled with the RO has increased compared to variance of the measuremets sampled with the function generator for almost all sampling times and leg lengths. However, as discussed before this raw variance also contains variance due to changes in the period of the ROs, and therefore gives skewed results.

Figure 5.19 shows the comparison between the results for the function generator and RO measurements, after the correction for the variance has been applied. Unfortunately, where the stochastic model and simulations predicted the measurements with a RO sample clock to have larger variance, this is no longer the case in the hardware tests conducted here.

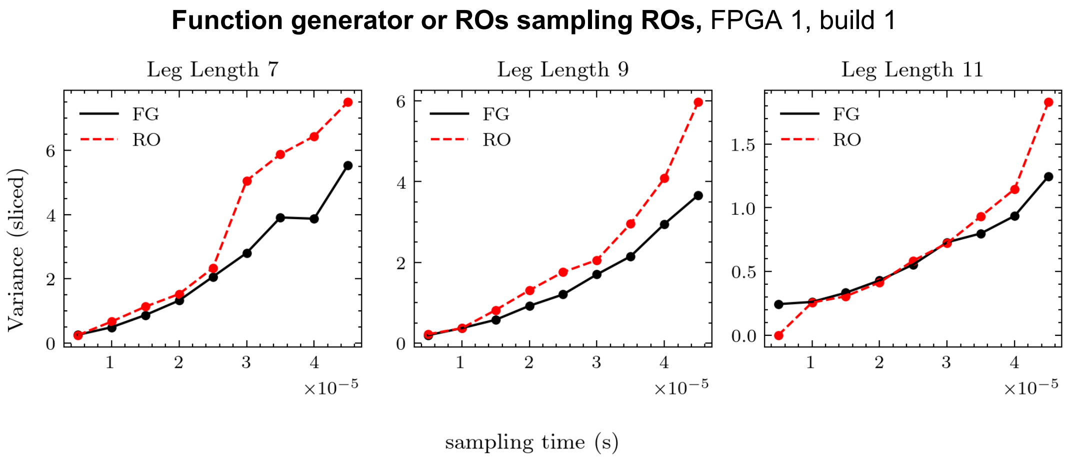

Figure 5.18: Comparison between the variance in ideal clock and non-ideal clock measurements.

These results clearly do not coincide with the mathematical findings, nor with the simulations in Chapter 4. The indication from the corrected variance plot even seems to be that using a sampling clock with more noise has little to no impact on the variance of the noise source and might even reduce it.

In Section 2.3 it was discussed that Fischer et al. [35] showed that using a second RO as the sample clock reduced the impact of global jitter. This would explain the reduction in variance when introducing a RO as sample clock. However, their results also showed that this is the case only when 1, the noise RO and sample RO have almost identical periods and 2, they are placed close together in hardware. Neither of these is true for the measurements conducted in this thesis.

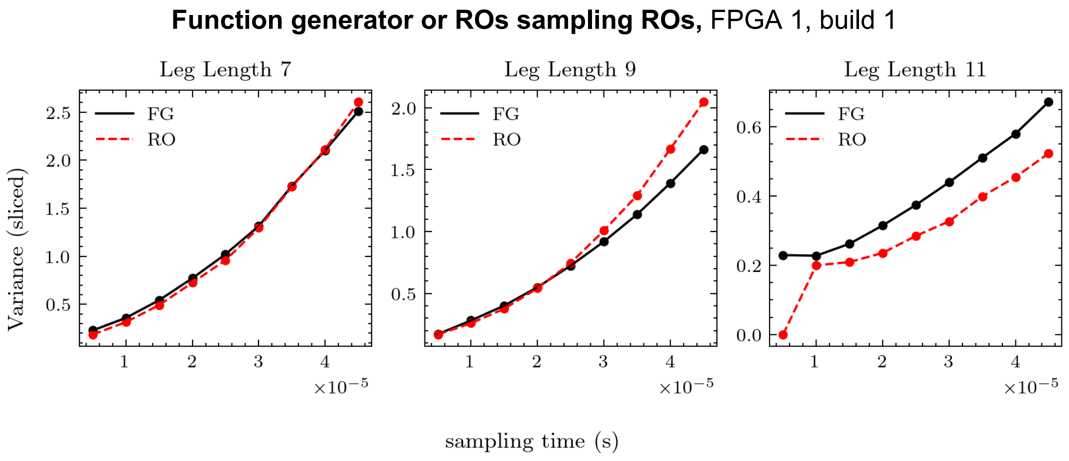

Figure 5.19: Comparison between the corrected variance in ideal clock and non-ideal clock measurements.

To test whether a correlation between two ROs might still exist, a design is made with two ROs of leg length 7, sampled at the same time using the function generator. Then, a moving average of the samples is taken. The results can be seen in Figure 5.20. On the y-axis, zero indicates the mean over all the samples for that RO. Note that again there are no placement constraints, and the periods of the ROs differ by almost 10%.

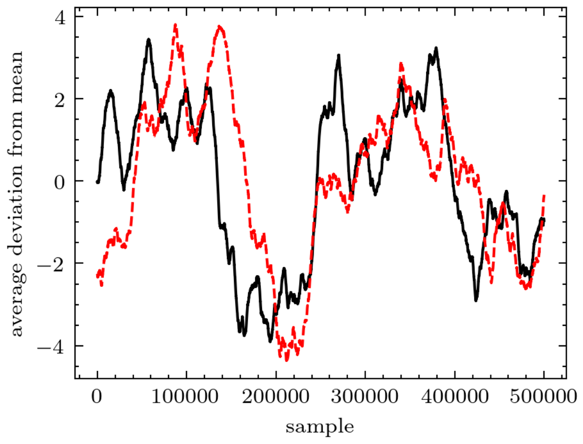

Figure 5.20: Moving average of two ROs sampled simultaneously.

From the plot, it is clear that there is at least some correlation between the ROs still. Calculating the Pearson correlation for the moving averages gives a value of 0.56, while the Pearson correlation for the samples is 0.24. Thus, there is quite some correlation between the ROs, but there is also still a lot of uncorrelated randomness, as indicated by the correlation for the samples being lower than that of the moving average.

It therefore seems that the sample RO at least partially compensates for the global jitter in the noise RO. This would indicate that although the total variance stays the same, a larger portion of this variance is due to local Gaussian jitter, and the influence of global jitter is reduced.

5.4. Discussion

The results presented in this chapter, obtained by measuring different configurations of the noise source in hardware using a function generator, PLL and ROs of different leg lengths, showed results that did not fully correspond with the mathematical and simulation results obtained so far. Although it is shown that for ROs with smaller periods, the variance increases quicker in time than for ROs with long periods, the increase was non-linear and did not double when sampling with a second RO.

This shows that noise other than the local Gaussian jitter of the ROs is present, skewing the results. It is no surprise that such noise exists. Other papers implementing RO-based TRNGs even show that these global deterministic jitters might lead to overestimation of the jitter [15, 16]. Sampling with a second RO could, under the right circumstances, reduce the impact of these global jitters by providing a differential measurement.

However, as discussed when analyzing the results, these circumstances were not met during the measurements. No constraints were placed on the routing of the ROs, and inspecting the place and route results showed that they were indeed not necessarily close together in hardware. This also yielded different periods for ROs with the same length, up to almost 30%. Sampling two ROs of the same length then showed correlations between the two, which would imply that at least some of the global noise still affects both ROs in the same manner.

If this is indeed the case, this could simply mean that a larger portion of the corrected variance is now due to local Gaussian noise in the ROs. On the other hand, the results in Figure 5.19 showed that the corrected variance was equal for both the function generator and RO measurements. This could be coincidental but was replicated in 2 different builds tested on 2 FPGAs.

Lastly, it must be noted that the applied correction to the variance is not very sophisticated. Better statistical analysis of the samples by, for example, combining it with the work on Allan variance in [17] could give a better understanding of what parts of the noise are due to the ROs.

5.5. Conclusion

In this chapter, an attempt was made to validate the mathematical models and conclusions. The test plan, including the physical test setup, was described. Then, the results were presented, after which they were analyzed. For the PLL measurements, the results showed a bimodal distribution, where an unimodal distribution was expected. It was found that the source of this inconsistency was in the PCIe reference clock. When using a proper reference clock, the PLL measurements did show the expected distribution. Some measurements with the function generator also showed the bimodal distribution. However, no satisfactory explanation was found for these measurements. It should be noted that these anomalies seemed to be almost independent of the hardware, only depending on the placement and routing.

Although the measurement results seemed to correspond well to the mathematical model at first, closer inspection of the observed variance in the counter outputs showed that it was non-linear. It was shown that this is at least partly due to the instability of the RO period, which can lead to overestimation of the variance and jitter. A simple correction based on calculating the average variance over slices of the data was presented. Lastly, the measurement results showed that the variance did not double when sampling with a non-ideal clock compared to the ideal clock. A possible explanation for this was given, but there is no compelling evidence to support this explanation. It is, therefore, also not possible to consider the model validated in hardware. This does not necessarily mean that the proposed noise source can not be used as a TRNG, but without validation of the stochastic model, no assurance can be provided for security applications.

6 Conclusion

This chapter first provides a summary and discussion of the highlights of this thesis in Section 6.1. Then, some potential future research directions are presented in Section 6.2.

6.1. Summary and Discussion

Chapter 1 gave the motivations for this thesis, an analysis of the state-of-the-art of random number generation, and an overview of the main contributions.

Chapter 2 provided an introduction to random number generation, focussing on FPGA based implementations. The different types of RNGs were discussed, and, using Cicek's classification, an overview of entropy sources was given. The chapter went into more detail on RO based designs, which are the focus of this thesis. A summary of security aspects of RO based designs, including vulnerabilities and different types of attacks, concluded the chapter.

Chapter 3 presented a design for a TRNG noise source based on counting the rising edges of a RO. A stochastic model was developed under the assumptions of an ideal sampling clock, which was later extended to use a non-ideal clock, such as a second RO. With these models, it is possible to calculate the minimum sampling time that gives sufficient min-entropy in the LSB of the counter. The other bits also contain entropy at this sampling time, improving the efficiency of the design. It was shown that ROs that use fewer inverters and thus less area should perform better. Using a second RO as a sampling clock allows for further reduction of the sampling time by a factor of 2 while achieving the same min-entropy.

Chapter 4 showed the results of two sets of simulations to verify the mathematical findings and conclusions from Chapter 3. The first simulations were carried out in Rust, using a very basic model of a RO period. After this, SPICE simulations were performed using Cadence Spectre. These simulations give a better insight into the expected behaviour of the noise source in hardware. Both simulations confirmed that ROs with shorter periods perform better and that using a second RO as the sampling clock halves the required sampling time to achieve the same min-entropy.

Chapter 5 attempted to validate the mathematical and simulation results in hardware using an Intel Arria 10 FPGA. After describing the test plan, the results were presented for measurements with a function generator, the FPGA's internal clock (a PLL), and ROs as sampling clocks. The function generator and PLL measurements should give a relatively good approximation of an ideal clock measurement. However, the PLL results unexpectedly showed bimodal distributions. It was found that this inconsistency was due to the PCIe reference clock. Some measurements with the function generator also showed a bimodal distribution, but no satisfactory explanation could be found.

Although the measurement results with the function generator and RO sampling clocks seemed to correspond well to the mathematical model at first, further inspection of the observed variance in the counter outputs showed non-linearity. This is at least partially due to the instability of the RO period, which leads to an overestimation of the variance. A simple correction was applied by calculating the average variance over slices of the data. After correction, the experiments did not show a larger counter variance when sampling with a second RO compared to sampling with the function generator. A possible explanation was given, but it was insufficient for the model to be considered validated.

The proposed design for the noise source, a ring oscillator incrementing a counter sampled after a sampling time, has some very appealing theoretical benefits compared to other RO based designs. First of all, the design benefits from having a RO with a short period, meaning fewer inverters and, thus, less area. Next to that, the counter is shown to have some entropy in the bits other than the LSB, which can be used in post-processing. This would increase the amount of output min-entropy by 60% without needing to increase the sampling time. The SPICE simulations confirmed the theoretical improvements a second RO as the sampling clock should give and confirmed that designs using smaller ROs are more efficient.

Unfortunately, validating the model in hardware proved to be more difficult. The experimental results showed a superlinear increase in variance, indicating other unmodeled noise components were present. Attempts to isolate the ROs or to measure the jitter directly were unsuccessful. Applying a correction after the measurements did improve the results somewhat, but not enough to consider the experiments a success. For this reason, no TRNG implementation based on the noise source was made.

6.2. Future Work

This section provides suggestions to enhance certain aspects discussed in this thesis. The suggestions are divided into three categories: verification, validation, and security.

- Verification
    1. The current SPICE simulations check for different temperatures and some noise bandwidth settings. A more detailed analysis could, for example, include more technology nodes, delay lines between elements, temperature and supply voltage noise, and other improvements to make the simulation results closer to resembling hardware implementations.
    2. In this work, the simulation results were only used to verify the proposed mathematical model. The next step would be to verify the randomness of these samples using, for example, the NIST [12] or AIS 20/31 [13] statistical test suites.
- Validation
    1. The attempts to validate the model in the Intel Arria 10 FPGA proved unsuccessful. The increase in variance was superlinear in time, which indicates that noise other than jitter was present. Future research in this direction could attempt to identify these noise sources and try to reduce their effects, or focus on better statistical analysis of the results in an attempt to remove noise from the measurements instead.
    2. As the SPICE simulations were successful, the next step would be to implement the ROs in an ASIC to see if comparable results will be obtained. Then, a TRNG design based on the validated model can also be tested using the AIS 20/31 test suite.
- Security
    1. The stochastic model of the noise source, combined with the SPICE simulation, shows that the entropy of the counter is largely dependent on the RO periods. As these are easily influenced by voltage and temperature changes, exhaustive testing is needed to ensure the noise source does not fail due to relatively simple heating fault attacks, for example.
    2. The paper by Bayon et al. [39] showed that it was possible to bias the output of a RO based TRNG using EM harmonic injection. The design, however, was based on XORing a number of parallel ROs to generate a single output bit. The noise source presented in this thesis might, therefore, be less susceptible to this type of attack, perhaps even depending on whether a second RO is used as the sample clock. Further research should be conducted to test this hypothesis.

Bibliography

[1] A. Kerckhoff, “La cryptographie militaire,” *Journal des sciences militaires*, vol. IX, pp. 5–38, Jan. 1883.

[2] K. Nohl, D. Evans, S. Starbug, and H. Plötz, “Reverse-engineering a cryptographic rfid tag,” in *Proceedings of the 17th Conference on Security Symposium*, ser. SS’08. USA: USENIX Association, 2008, p. 185–193.

[3] G. de Koning Gans, J.-H. Hoepman, and F. D. Garcia, “A practical attack on the mifare classic,” in *Smart Card Research and Advanced Applications*, G. Grimaud and F.-X. Standaert, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 267–282.

[4] M. J. Dworkin, “Advanced encryption standard (AES),” no. NIST FIPS 197-upd1, pp. NIST FIPS 197–upd1, May 2023. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197-upd1.pdf

[5] E. Barker, A. Roginsky, and R. Davis, “Recommendation for cryptographic key generation,” no. NIST SP 800-133r2, pp. NIST SP 800–133r2, Jun. 2020. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-133r2.pdf

[6] R. Gennaro, “Randomness in cryptography,” *IEEE Security & Privacy*, vol. 4, no. 2, pp. 64–67, 2006.

[7] M. Dworkin, “Recommendation for block cipher modes of operation: Methods and techniques,” no. NIST SP 800-38A, pp. NIST SP 800–38A, Dec. 2001.

[8] N. Heninger, Z. Durumeric, E. Wustrow, and J. A. Halderman, “Mining your ps and qs: Detection of widespread weak keys in network devices,” in *21st USENIX Security Symposium (USENIX Security 12)*. Bellevue, WA: USENIX Association, Aug. 2012, pp. 205–220. [Online]. Available: https://www.usenix.org/conference/usenixsecurity12/technical-sessions/presentation/heninger

[9] F. Galton, “Dice for statistical experiments,” *Nature*, vol. 42, no. 1070, pp. 13–14, May 1890.

[10] Rand Corporation, *A Million Random Digits with 100,000 Normal Deviates*. Free Press, 1955.

[11] Intel Corporation, “Intel 810 chipset design guide,” Jun. 1999.

[12] A. Rukhin, J. Soto, J. Nechvatal, E. Barker, S. Leigh, M. Levenson, D. Banks, A. Heckert, and J. Dray, “A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications,” Sep. 2010.

[13] W. Schindler and M. Peter, “A Proposal for Functionality Classes for Random Number Generators,” Sep. 2022, Version 2.35 - DRAFT.

[14] R. G. Brown, D. Eddelbuettel, and D. Bauer, “Dieharder: a random number test suite (version 3.31.1),” 2024. [Online]. Available: https://webhome.phy.duke.edu/~rgb/General/dieharder.php

[15] Y. Ma, J. Lin, T. Chen, C. Xu, Z. Liu, and J. Jing, “Entropy Evaluation for Oscillator-Based True Random Number Generators,” in *Advanced Information Systems Engineering*, D. Hutchison, T. Kanade, J. Kittler, J. M. Kleinberg, F. Mattern, J. C. Mitchell, M. Naor, O. Nierstrasz, C. Pandu Rangan, B. Steffen, M. Sudan, D. Terzopoulos, D. Tygar, M. Y. Vardi, G. Weikum, C. Salinesi, M. C. Norrie, and Ó. Pastor, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, vol. 7908, pp. 544–561.

[16] M. Baudet, D. Lubicz, J. Micolod, and A. Tassiaux, “On the Security of Oscillator-Based Random Number Generators,” *Journal of Cryptology*, vol. 24, no. 2, pp. 398–425, Apr. 2011.

[17] E. Noumon Allini, M. Skórski, O. Petura, F. Bernard, M. Laban, and V. Fischer, “Evaluation and Monitoring of Free Running Oscillators Serving as Source of Randomness,” *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pp. 214–242, Aug. 2018.

[18] W. Killmann and W. Schindler, “A Design for a Physical RNG with Robust Entropy Estimators,” in *Cryptographic Hardware and Embedded Systems – CHES 2008*, E. Oswald and P. Rohatgi, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, vol. 5154, pp. 146–163.

[19] Bundesamt für Sicherheit in der Informationstechnik, “Overview of Linux kernels with NTG.1- or DRG.3-compliant random number generator /dev/random.”

[20] M. S. Turan, E. Barker, J. Kelsey, K. A. McKay, M. L. Baish, and M. Boyle, “Recommendation for the entropy sources used for random bit generation,” National Institute of Standards and Technology, Gaithersburg, MD, Tech. Rep. NIST SP 800-90b, Jan. 2018.

[21] I. Cicek, A. E. Pusane, and G. Dundar, “A novel design method for discrete time chaos based true random number generators,” *Integration*, vol. 47, no. 1, pp. 38–47, Jan. 2014.

[22] X. Wang, H. Liu, R. Zhang, K. Liu, and H. Shinohara, “An Inverter-Based True Random Number Generator with 4-bit Von-Neumann Post-Processing Circuit,” in *2020 IEEE 63rd International Midwest Symposium on Circuits and Systems (MWSCAS)*. Springfield, MA, USA: IEEE, Aug. 2020, pp. 285–288.

[23] R. Zhang, S. Chen, C. Wan, and H. Shinohara, “High-throughput Von Neumann post-processing for random number generator,” in *2018 International Symposium on VLSI Design, Automation and Test (VLSI-DAT)*. Hsinchu: IEEE, Apr. 2018, pp. 1–4.

[24] R. Sivaraman, S. Rajagopalan, A. Sridevi, J. B. B. Rayappan, M. P. V. Annamalai, and A. Rengarajan, “Metastability-Induced TRNG Architecture on FPGA,” *Iranian Journal of Science and Technology, Transactions of Electrical Engineering*, vol. 44, no. 1, pp. 47–57, Mar. 2020.

[25] J. Bahrami, M. Ebrahimabadi, S. Guilley, J.-L. Danger, and N. Karimi, “Impact of Process Mismatch and Device Aging on SR-Latch Based True Random Number Generators,” in *Constructive Side-Channel Analysis and Secure Design*, R. Wacquez and N. Homma, Eds. Cham: Springer Nature Switzerland, 2024, vol. 14595, pp. 177–196.

[26] T. L. Liao, P. Y. Wan, and J.-J. Yan, “Design and Synchronization of Chaos-Based True Random Number Generators and Its FPGA Implementation,” *IEEE Access*, vol. 10, pp. 8279–8286, 2022.

[27] B. Sunar, W. Martin, and D. Stinson, “A Provably Secure True Random Number Generator with Built-In Tolerance to Active Attacks,” *IEEE Transactions on Computers*, vol. 56, no. 1, pp. 109–119, Jan. 2007.

[28] K. Wold and C. H. Tan, “Analysis and Enhancement of Random Number Generator in FPGA Based on Oscillator Rings,” *International Journal of Reconfigurable Computing*, vol. 2009, pp. 1–8, 2009.

[29] M. Varchola and M. Drutarovsky, “New High Entropy Element for FPGA Based True Random Number Generators,” in *Cryptographic Hardware and Embedded Systems, CHES 2010*, D. Hutchison, T. Kanade, J. Kittler, J. M. Kleinberg, F. Mattern, J. C. Mitchell, M. Naor, O. Nierstrasz, C. Pandu Rangan, B. Steffen, M. Sudan, D. Terzopoulos, D. Tygar, M. Y. Vardi, G. Weikum, S. Mangard, and F.-X. Standaert, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, vol. 6225, pp. 351–365.

[30] A. Cherkaoui, V. Fischer, L. Fesquet, and A. Aubert, “A Very High Speed True Random Number Generator with Entropy Assessment,” in *Cryptographic Hardware and Embedded Systems - CHES 2013*, G. Bertoni and J.-S. Coron, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, vol. 8086, pp. 179–196.

[31] B. Valtchanov, A. Aubert, F. Bernard, and V. Fischer, “Modeling and observing the jitter in ring oscillators implemented in FPGAs,” in *2008 11th IEEE Workshop on Design and Diagnostics of Electronic Circuits and Systems*. Bratislava, Slovakia: IEEE, Apr. 2008, pp. 1–6.

[32] J. J. L. Franco, E. Boemo, E. Castillo, and L. Parrilla, “Ring oscillators as thermal sensors in fpgas: Experiments in low voltage,” in *2010 VI Southern Programmable Logic Conference (SPL)*, 2010, pp. 133–137.

[33] E. Boemo and S. López-Buedo, “Thermal monitoring on fpgas using ring-oscillators,” in *Field-Programmable Logic and Applications*, W. Luk, P. Y. K. Cheung, and M. Glesner, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 1997, pp. 69–78.

[34] L. De Micco, C. Minchola, J. J. Leon-Franco, E. Boemo, and M. Antonelli, “An annotated guide to utilize ring-oscillators as thermal sensor in fpga technology,” in *2020 Argentine Conference on Electronics (CAE)*, 2020, pp. 1–7.

[35] V. Fischer, F. Bernard, N. Bochard, and M. Varchola, “Enhancing security of ring oscillator-based trng implemented in FPGA,” in *2008 International Conference on Field Programmable Logic and Applications*. Heidelberg, Germany: IEEE, 2008, pp. 245–250.

[36] P. Bayon, L. Bossuet, A. Aubert, and V. Fischer, “Electromagnetic analysis on ring oscillator-based true random number generators,” in *2013 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2013, pp. 1954–1957.

[37] K. Ngo and E. Dubrova, “Side-channel analysis of the random number generator in stm32 mcus,” in *Proceedings of the Great Lakes Symposium on VLSI 2022*, ser. GLSVLSI ’22. New York, NY, USA: Association for Computing Machinery, 2022, p. 15–20. [Online]. Available: https://doi.org/10.1145/3526241.3530324

[38] A. Markettos and S. Moore, “The frequency injection attack on ring-oscillator-based true random number generators,” in *Cryptographic Hardware and Embedded Systems - CHES 2009, 11th International Workshop, Lausanne, Switzerland, September 6-9, 2009, Proceedings*, ser. Lecture Notes in Computer Science, vol. 5747. Springer, 09 2009, pp. 317–331.

[39] P. Bayon, L. Bossuet, A. Aubert, V. Fischer, F. Poucheret, B. Robisson, and P. Maurine, “Contactless electromagnetic active attack on ring oscillator based true random number generator,” in *Constructive Side-Channel Analysis and Secure Design*, W. Schindler and S. A. Huss, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 151–166.

[40] L. Machado, A. Roca, and J. Cortadella, “Increasing the robustness of digital circuits with ring oscillator clocks,” in *2nd International Workshop on Resiliency in Embedded Electronic Systems: SwissTech Convention Centre, Lausanne, Switzerland, March 31st, 2017: final proceedings*, 2017, pp. 29 – 34. [Online]. Available: http://hdl.handle.net/2117/114922

[41] S. El Amraoui, R. Leveugle, and P. Maistri, “Choose your Path: Control of Ring Oscillators EMFI Susceptibility through FPGA P&R Constraints,” in *International Symposium on Design and Diagnostics of Electronic Circuits and Systems (DDECS 2024)*, ser. Proceedings, Kielce, Poland, Apr. 2024. [Online]. Available: https://hal.science/hal-04513560

[42] Z. Zhang and T. Su, “Behavioral analysis and immunity design of the ro-based trng under electromagnetic interference,” *Electronics*, vol. 10, no. 11, 2021. [Online]. Available: https://www.mdpi.com/2079-9292/10/11/1347

[43] H. Martin, P. Martin-Holgado, P. Peris-Lopez, Y. Morilla, and L. Entrena, “On the entropy of oscillator-based true random number generators under ionizing radiation.” *Entropy (Basel, Switzerland)*, vol. 20, Jul 2018.

[44] H. Martin, T. Korak, E. Millán, and M. Hutter, “Fault attacks on strngs: Impact of glitches, temperature, and underpowering on randomness,” *IEEE Transactions on Information Forensics and Security*, vol. 10, pp. 266 – 277, 02 2015.

[45] A. Meurer, C. P. Smith, M. Paprocki, O. Čertík, S. B. Kirpichev, M. Rocklin, A. Kumar, S. Ivanov, J. K. Moore, S. Singh, T. Rathnayake, S. Vig, B. E. Granger, R. P. Muller, F. Bonazzi, H. Gupta, S. Vats, F. Johansson, F. Pedregosa, M. J. Curry, A. R. Terrel, v. Roučka, A. Saboo, I. Fernando, S. Kulal, R. Cimrman, and A. Scopatz, “Sympy: symbolic computing in python,” *PeerJ Computer Science*, vol. 3, p. e103, Jan. 2017. [Online]. Available: https://doi.org/10.7717/peerj-cs.103

[46] E. B. Barker and J. M. Kelsey, “Recommendation for Random Number Generation Using Deterministic Random Bit Generators,” National Institute of Standards and Technology, Tech. Rep. NIST SP 800-90Ar1, Jun. 2015.

[47] The Rand Project Developers, “The Rust Rand Book,” Available: https://rust-random.github.io/book/, 2024, [Accessed: 21 May 2024].

[48] D. J. Bernstein *et al.*, “Chacha, a variant of salsa20,” in *Workshop record of SASC*, vol. 8, no. 1. Citeseer, 2008, pp. 3–5.

[49] BSIM Group, UC Berkely, “The BSIM Family,” Available: https://bsim.berkeley.edu/models/, [Accessed: 26 June 2024].

[50] “Spread spectrum clocking,” Microsemi Corporation, White Paper, June 2015.

A Verification

This appendix contains code snippets and some additional plots of both the Rust and SPICE simulation results. The Rust simulation results are presented first in Section A.1, followed by the SPICE results in Section A.2.

A.1. Rust Simulations

A.1.1. Rust Code

The code snippets below constitute the Rust implementation of the algorithms described in Section 4.1. By using a multithreaded approach, a lot of samples could be generated very quickly, which helped the verification of the model.

Listing A.1: Rust implementation of the algorithm for ideal clock simulations.

```rust
use rand::thread_rng;
use rand_distr::{Normal, Distribution};
use std::thread;
use std::sync::mpsc;

fn ideal_clock_sim(experiments: u64, mu: f64, sigma: f64, sample_time: f64
    ) -> Vec<i32> {
    let threads = 50;
    let thread_experiments = experiments / threads;
    let mut results: Vec<i32> = Vec::with_capacity(experiments as usize);
    let (tx, rx) = mpsc::channel();
    for _ in 0..threads {
        let tx_clone = tx.clone();
        thread::spawn(move || {
            let ro_normal = Normal::new(mu, sigma).unwrap();
            let mut rng = thread_rng;

            for _ in 0..thread_experiments {
                let mut ro_time = ro_normal.sample(&mut rng);
                let mut cnt: i32 = 0;
                while ro_time < sample_time {
                    ro_time += ro_normal.sample(&mut rng);
                    cnt += 1;
                }
                tx_clone.send(cnt).unwrap();
            }
        });
    }
    drop(tx);
    for received in rx {
        results.push(received)
    }
    results
}
```

Listing A.2: Rust implementation of the algorithm for non-ideal clock simulations.

```rust
use rand::thread_rng;
use rand_distr::{Normal, Distribution};
use std::thread;
use std::sync::mpsc;

fn non_ideal_clock_sim(experiments: u64, mu: f64, sigma: f64, n2: f64) ->
    Vec<i32> {
    let threads = 50;
    let thread_experiments = experiments / threads;
    let mut results: Vec<i32> = Vec::with_capacity(experiments as usize);
    let (tx, rx) = mpsc::channel();
    for _ in 0..threads {
        let tx_clone = tx.clone();
        thread::spawn(move || {
            let cl_normal = Normal::new(n2 * mu, n2.sqrt() * sigma)
            let ro_normal = Normal::new(mu, sigma).unwrap();
            let mut rng = thread_rng;

            for _ in 0..thread_experiments {
                let mut ro_time = ro_normal.sample(&mut rng);
                let mut cnt: i32 = 0;
                let sample_time = cl_normal.sample(&mut rng);
                while ro_time < sample_time {
                    ro_time += ro_normal.sample(&mut rng);
                    cnt += 1;
                }
                tx_clone.send(cnt).unwrap();
            }
        });
    }
    drop(tx);
    for received in rx {
        results.push(received)
    }
    results
}
```

A.1.2. Some Plots of the Ideal Model

The following figures show simulation results for the ideal clock simulations. Figure A.1 shows results for various sampling times for a RO period of 3ns. The sampling time is chosen so that the distribution follows the best-case for the min-entropy. This can be seen from the even distribution, which results in a fifty-fifty distribution of zeroes and ones in the LSB no matter the sampling time. Figure A.2 shows the worst case, where the distribution is odd, and some minimum sampling is needed to achieve the required min-entropy.

Figure A.3 shows the results for various sampling times for a RO period of 5ns. The width of the distribution has decreased for all sampling times compared to those in Figure A.2.

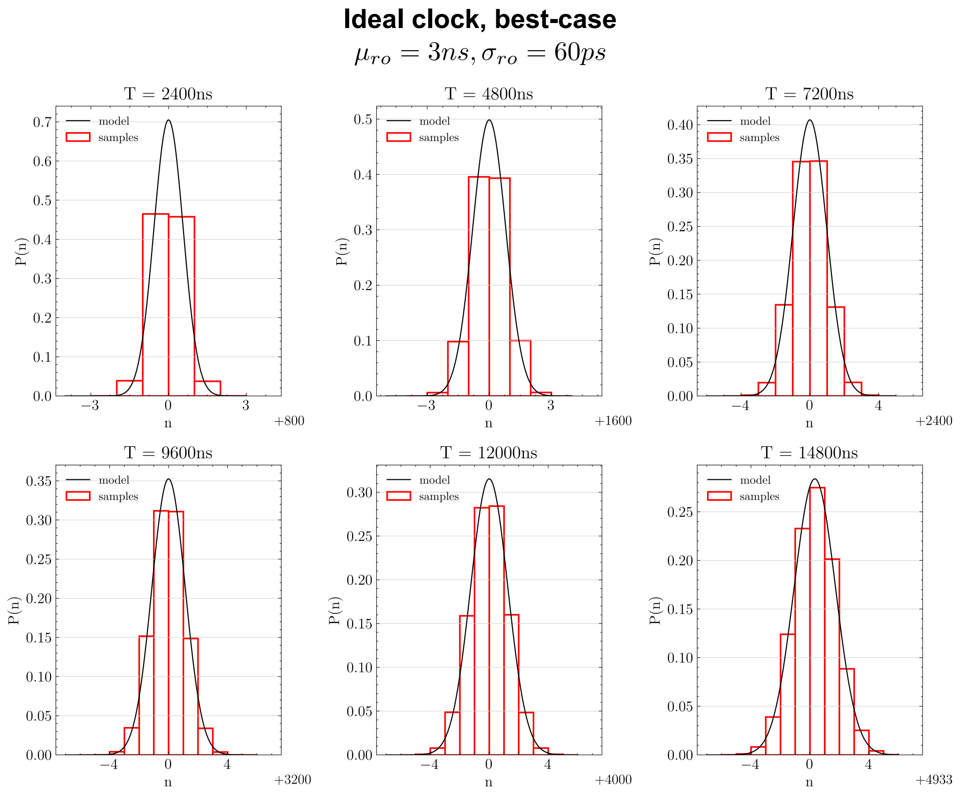

Figure A.1: Impact of sampling time T on distribution of samples.

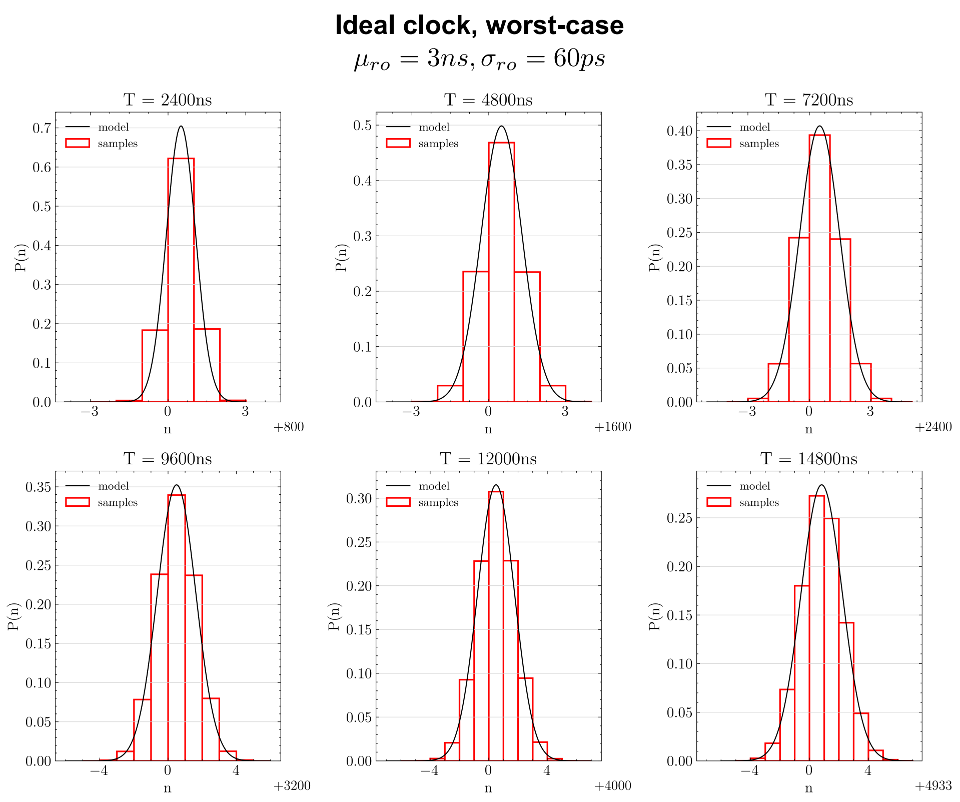

Figure A.2: Impact of sampling time T on distribution of samples.

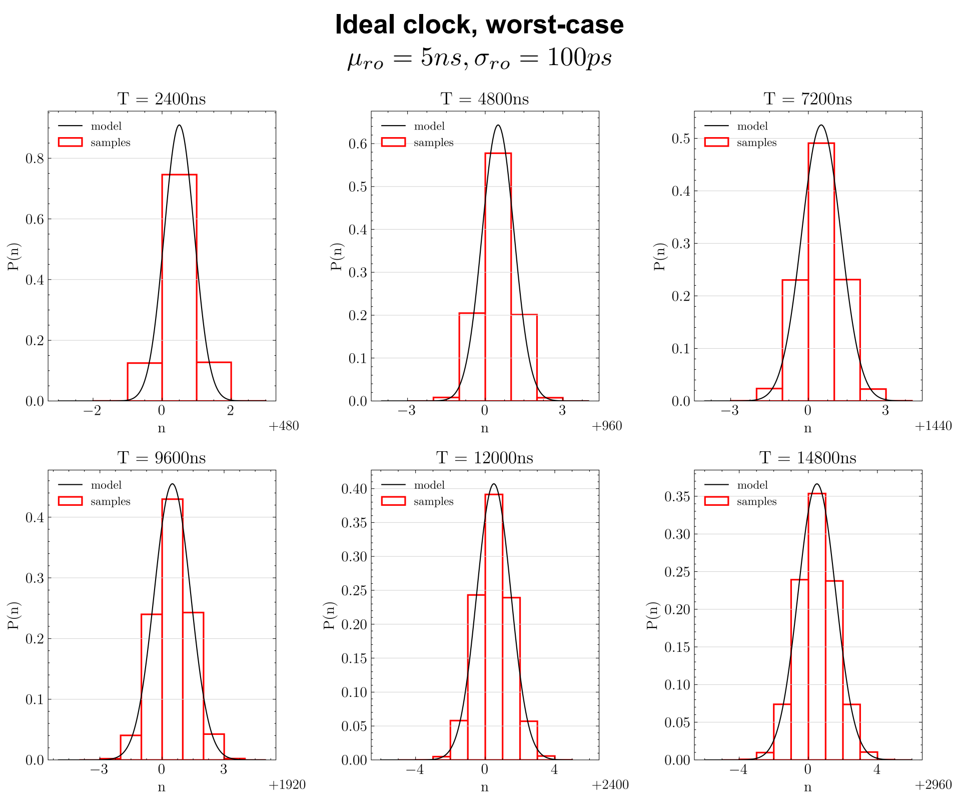

Figure A.3: Impact of sampling time T on distribution of samples.

A.1.3. Some Plots of the Non-Ideal Model

In this section, figures are presented that show some of the results of the non-ideal clock simulations. Figure A.4 shows results for various sampling times for a RO period of 3ns, sampled with a second RO with a period of 3ns, in the best-case scenario. The width of the distribution has increased compared to the distributions presented in Figure A.1. Figure A.5 shows simulations for the same RO periods, now in the worst-case scenario. Again, the difference between the odd and even distributions is very clear.

Figure A.6 shows results for RO periods of 5ns, showing the decline in variance for all sampling times due to a longer period.

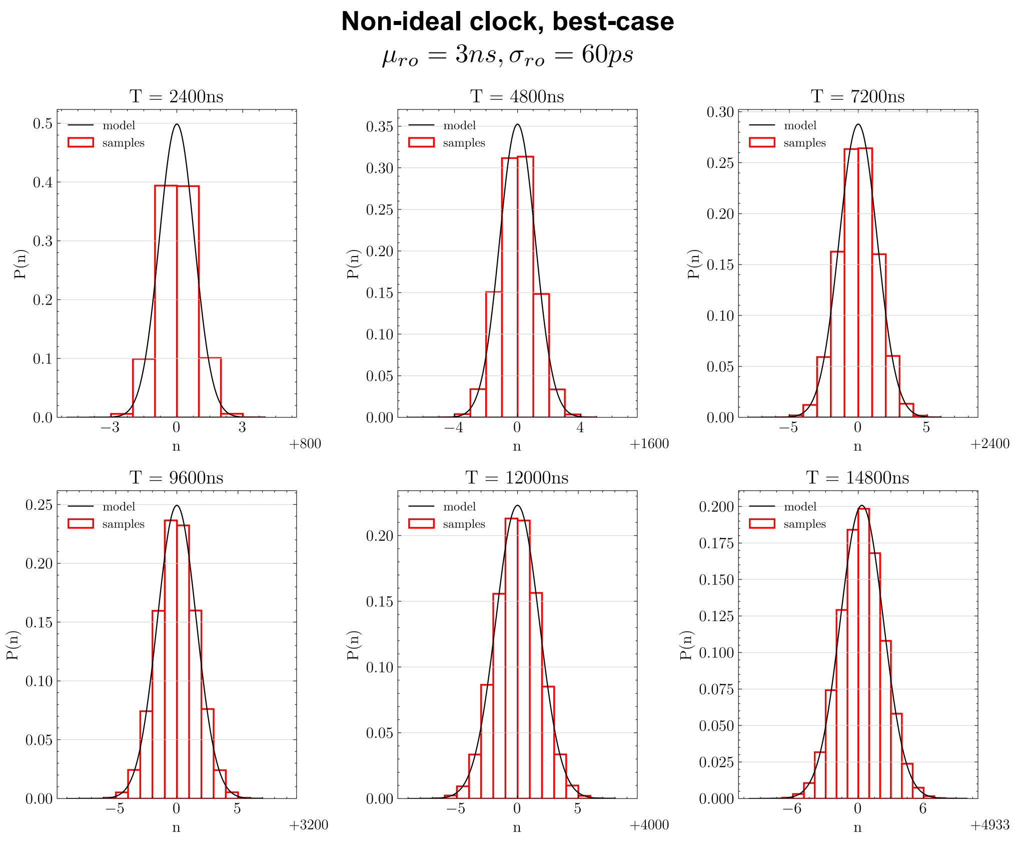

Figure A.4: Impact of sampling time T on distribution of samples.

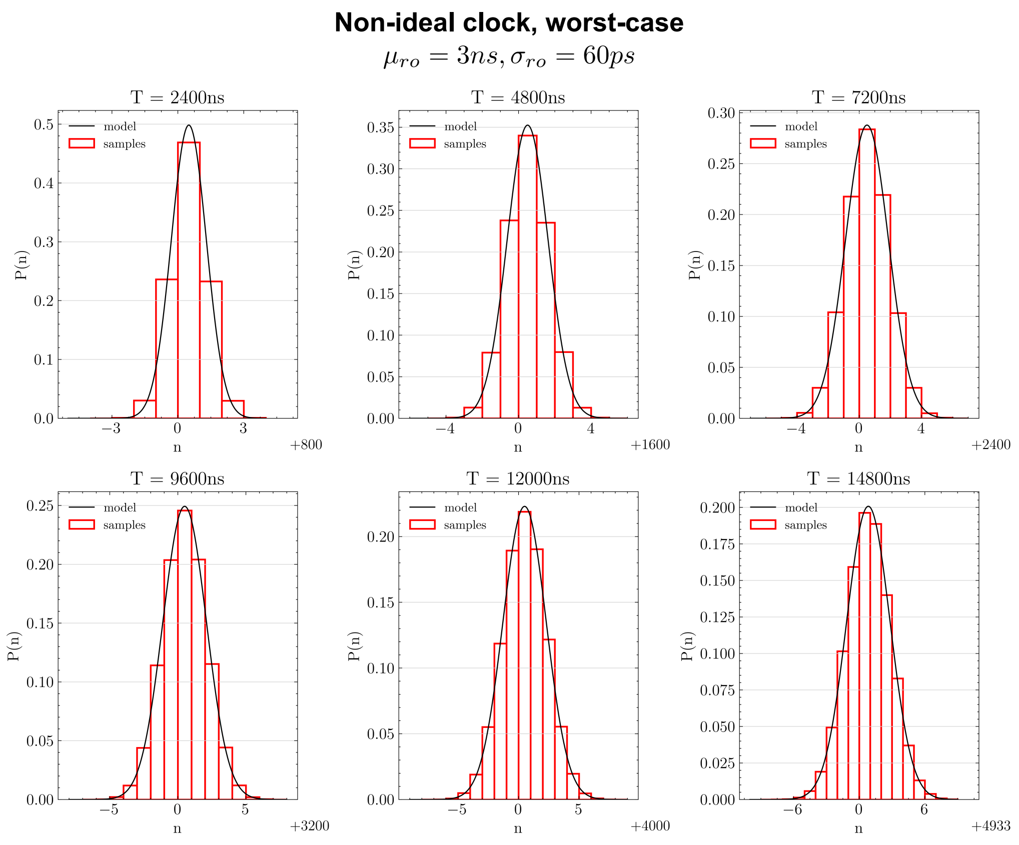

Figure A.5: Impact of sampling time T on distribution of samples.

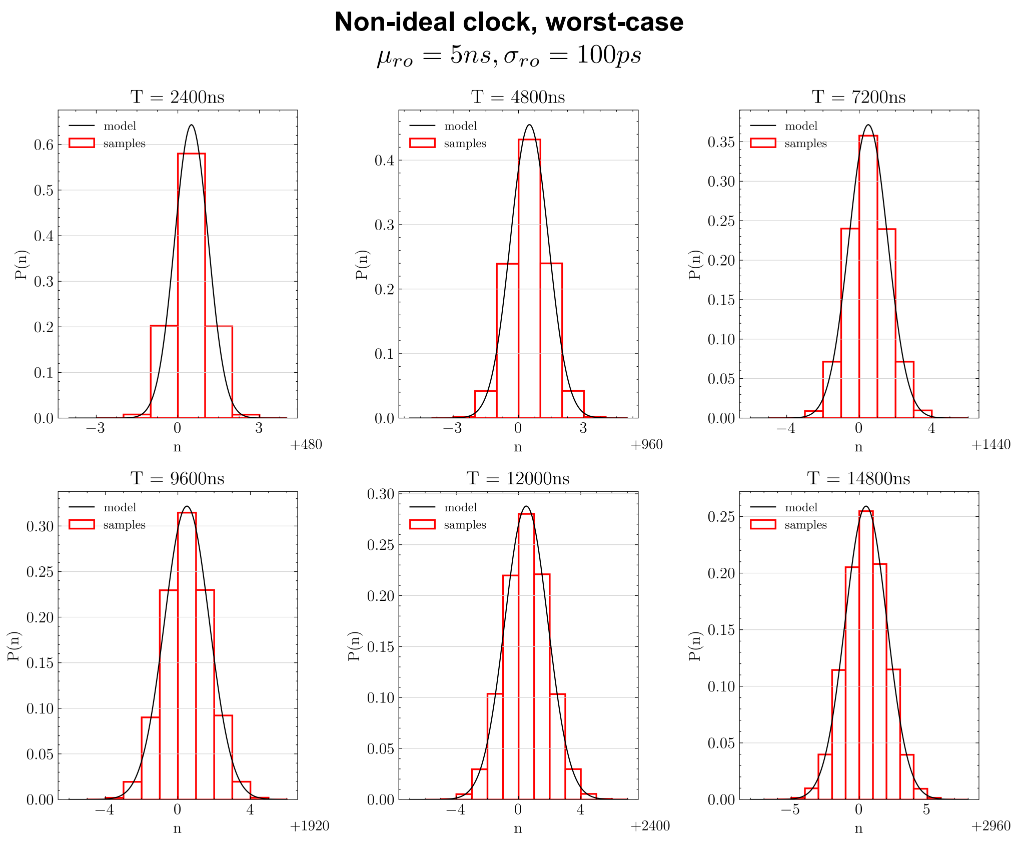

Figure A.6: Impact of sampling time T on distribution of samples.

A.2. Spectre Simulations

A.2.1. Plots Ideal Model

The following figures show some results of the ideal clock SPICE simulations. Figure A.7 shows simulation results for different sampling times for a RO consisting of 3 inverters. Figure A.8 shows the same RO for different sampling times, but now for a temperature of $-10°C$. Recall from the analysis in Chapter 4 that the noise source showed a larger variance for low temperatures.

Figure A.9 shows results for a RO with 7 inverters. The width of the distribution, and with it the variance, has clearly decreased compared to Figure A.7.

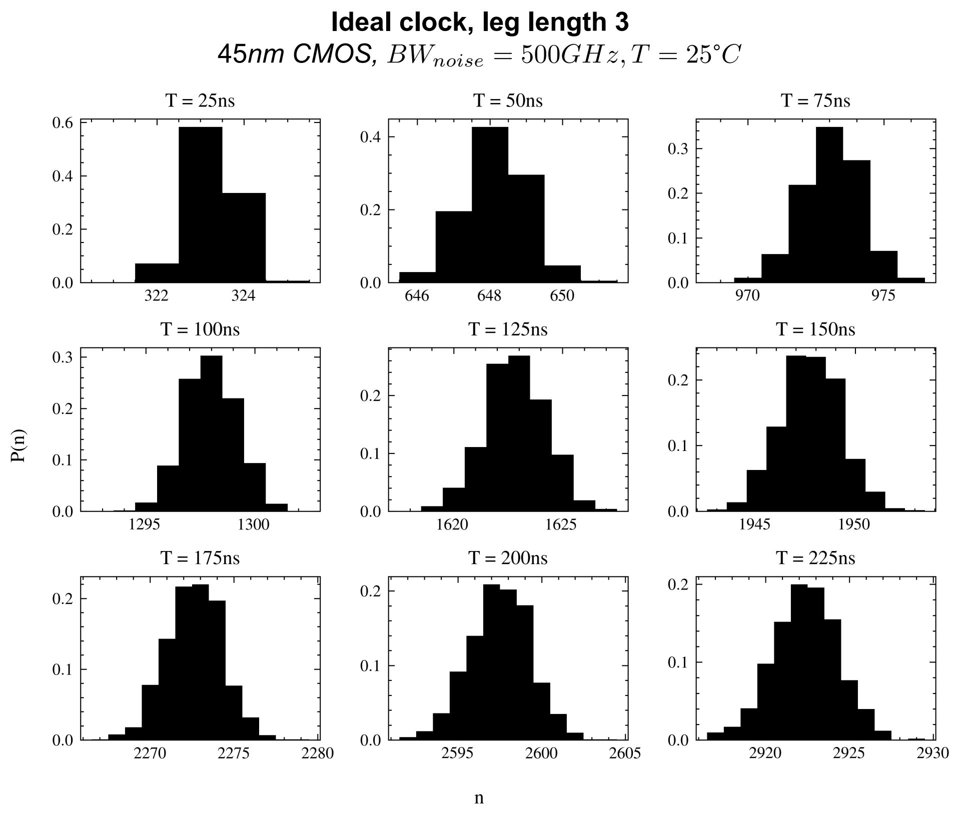

Figure A.7: Impact of sampling time T on distribution of samples.

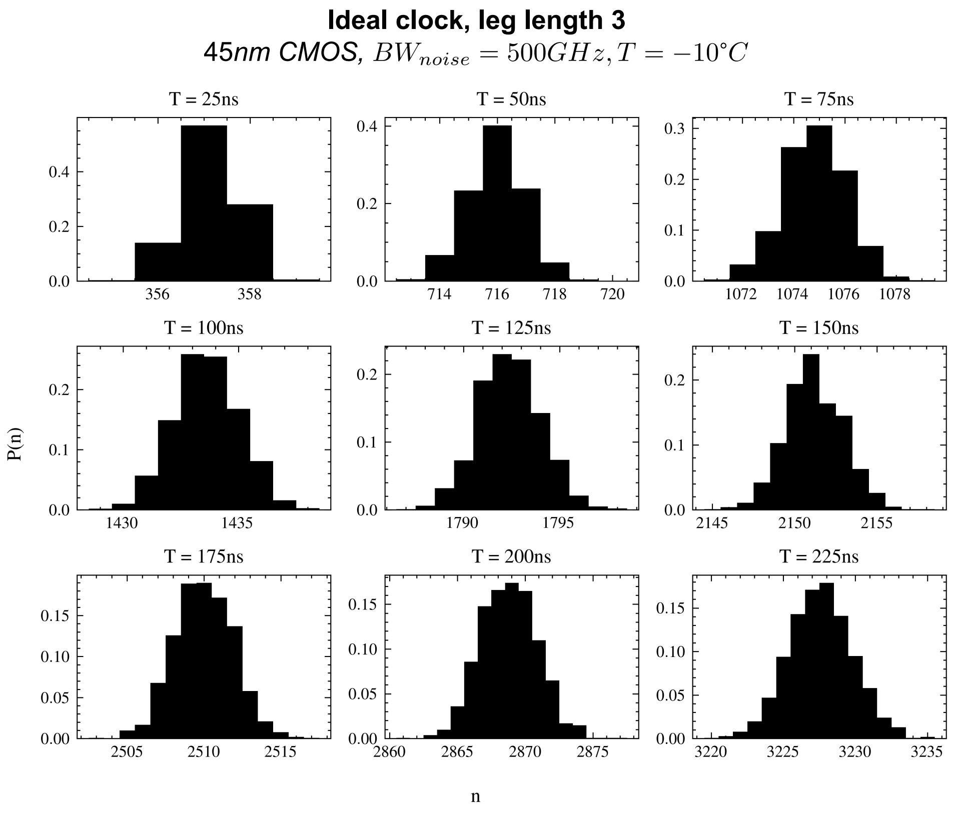

Figure A.8: Impact of sampling time T on distribution of samples.

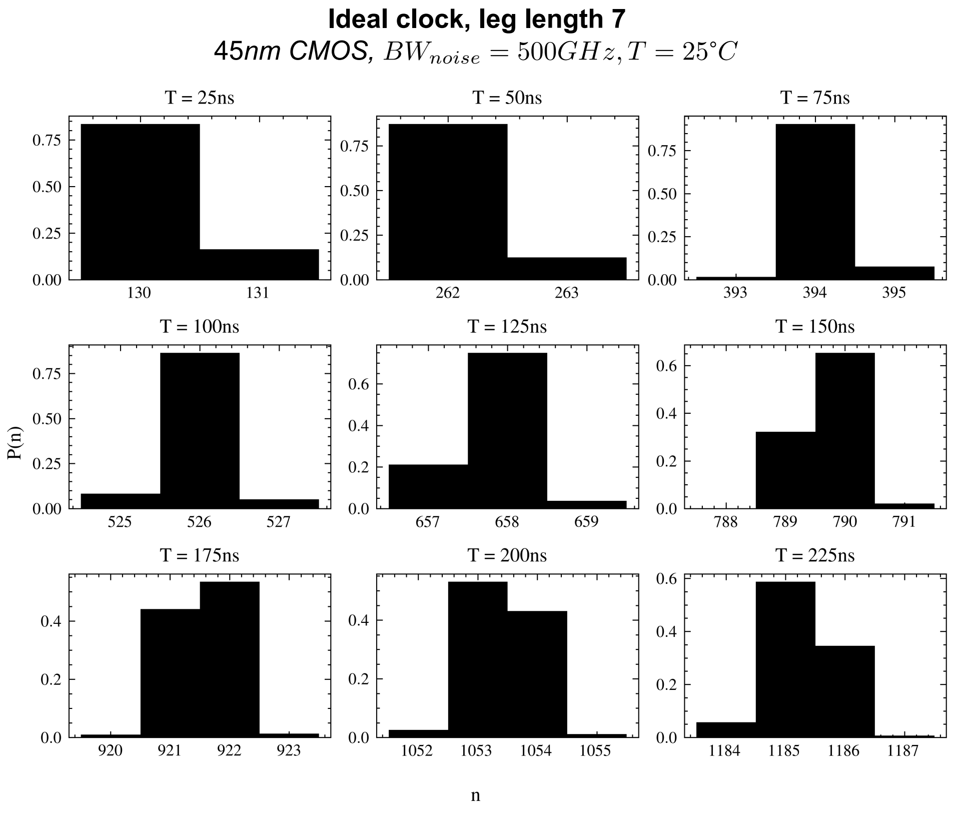

Figure A.9: Impact of sampling time T on distribution of samples.

A.2.2. Plots Non-Ideal Model

In this section, some results of the non-ideal clock simulations in SPICE are presented. Figure A.10 shows results for a RO of length 3, now sampled using a different simulated RO with 3 inverters as well. Figure A.11 shows the same simulation, with a temperature of $-10°C$. Figure A.12 shows the distributions when the ROs have leg length 7. All figures show a clear increase in variance compared to the figures of the ideal clock simulations.

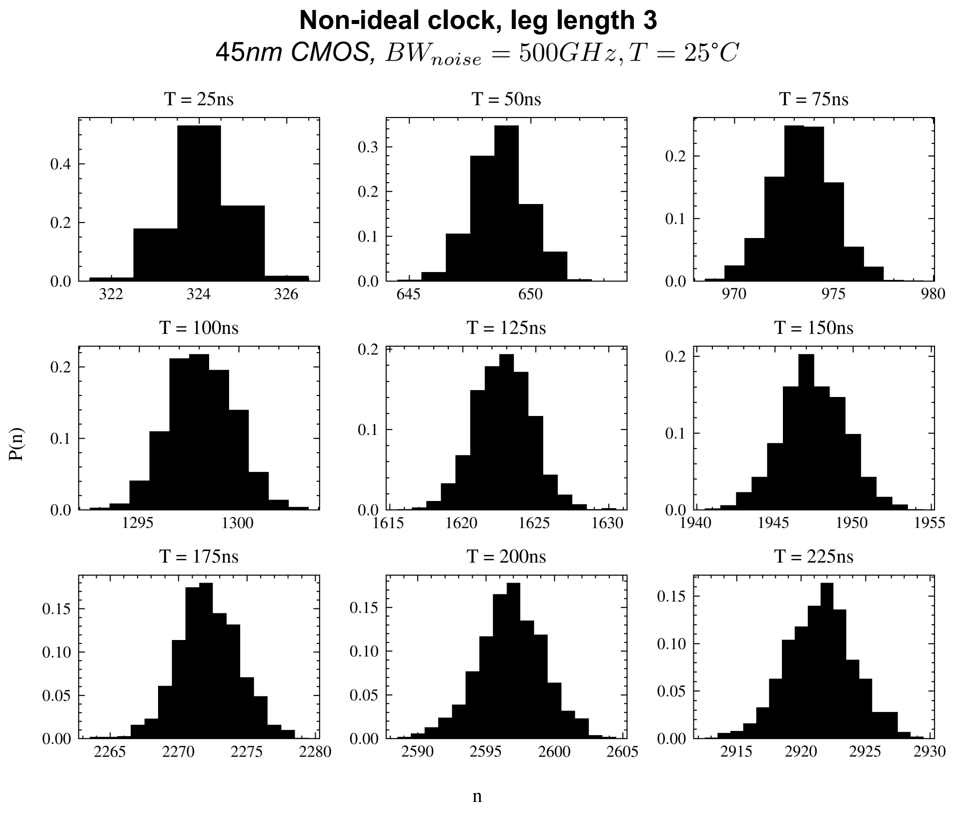

Figure A.10: Impact of sampling time T on distribution of samples.

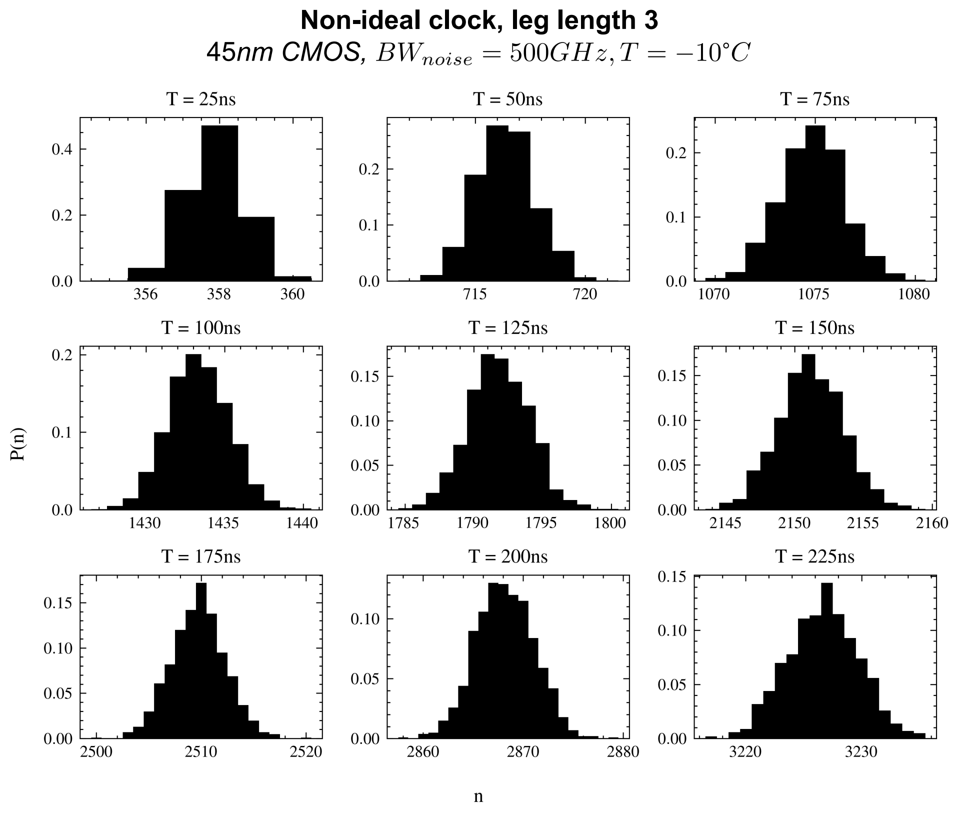

Figure A.11: Impact of sampling time T on distribution of samples.

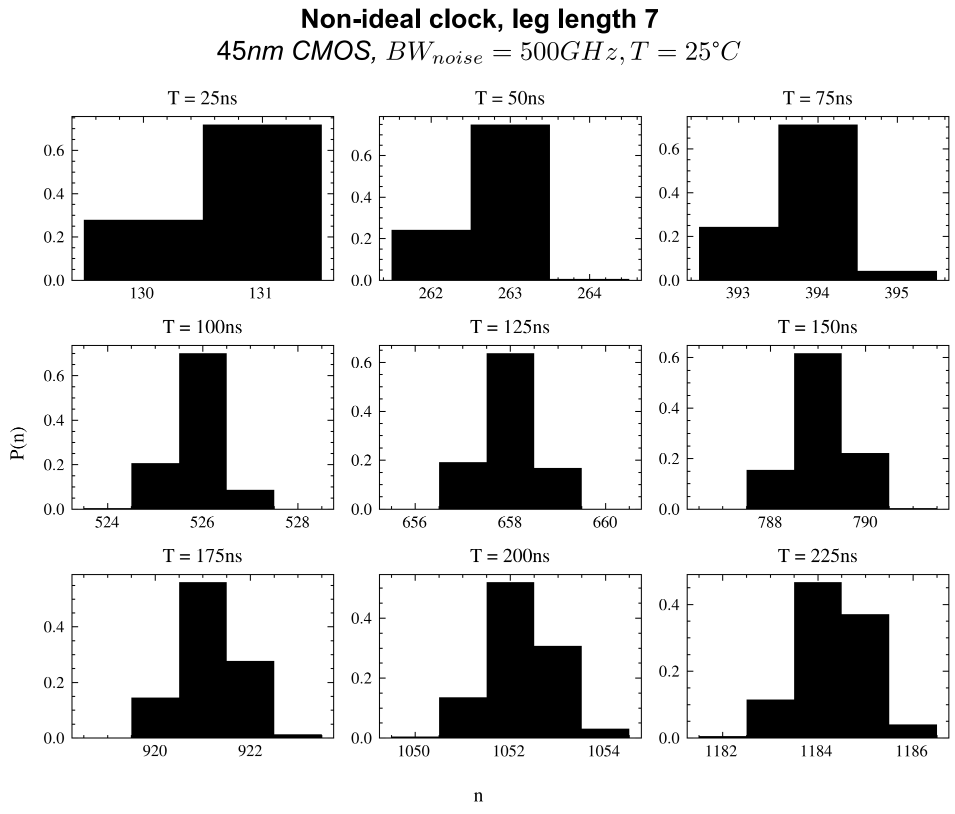

Figure A.12: Impact of sampling time T on distribution of samples.

B Validation

In this appendix, the measurement results from the experiments conducted on the Intel Arria 10 FPGA platform are presented. For the function generator and RO sample clock, all results are included for RO leg lengths 7, 9 and 11, both FPGAs, and both builds. For the PLL measurements, only the results for FPGA 1, build 1 are included.

The function generator measurements are in Section B.1, the PLL results in Section B.2, and the RO results in Section B.3.

B.1. Measurement Results Function Generator

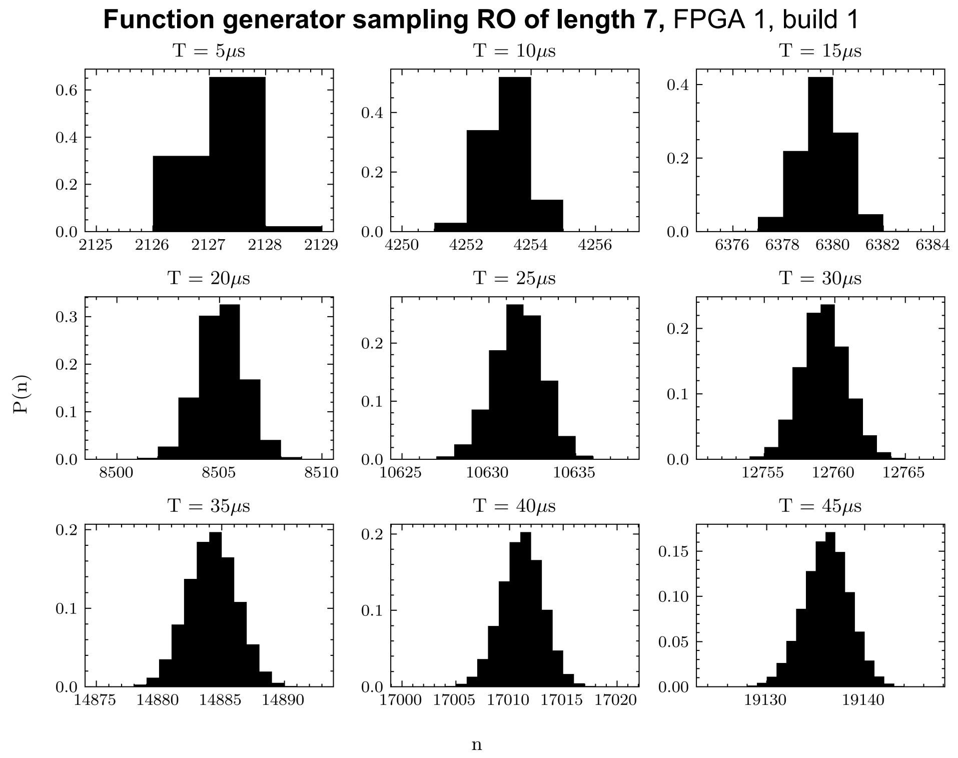

Figure B.1: Impact of sampling time T on distribution of samples.

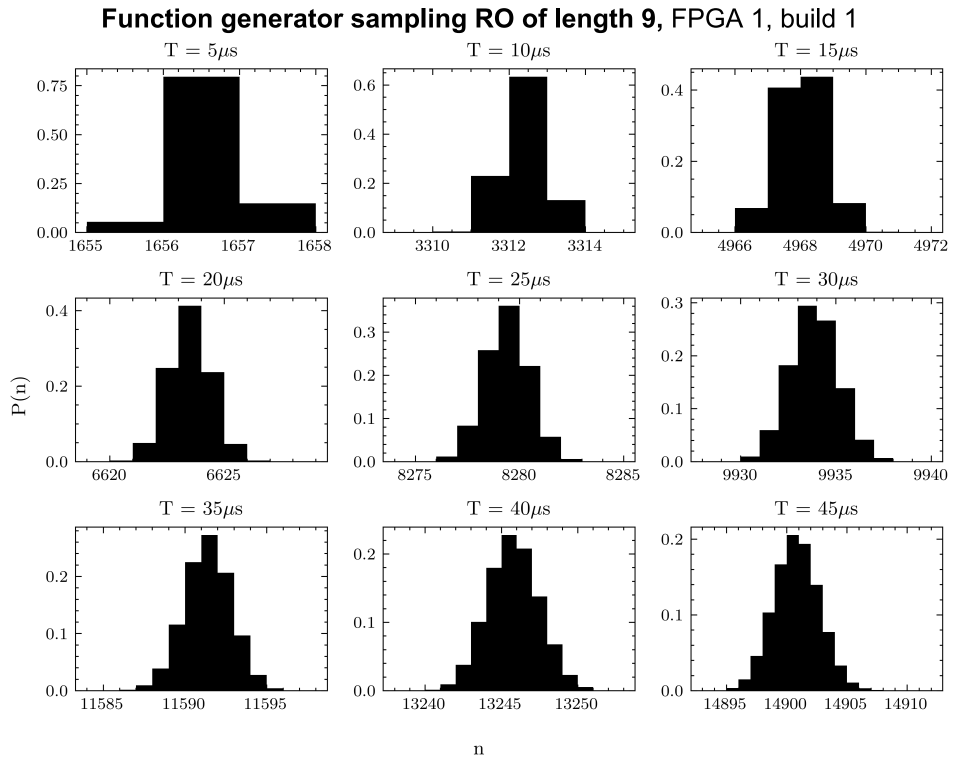

Figure B.2: Impact of sampling time T on distribution of samples.

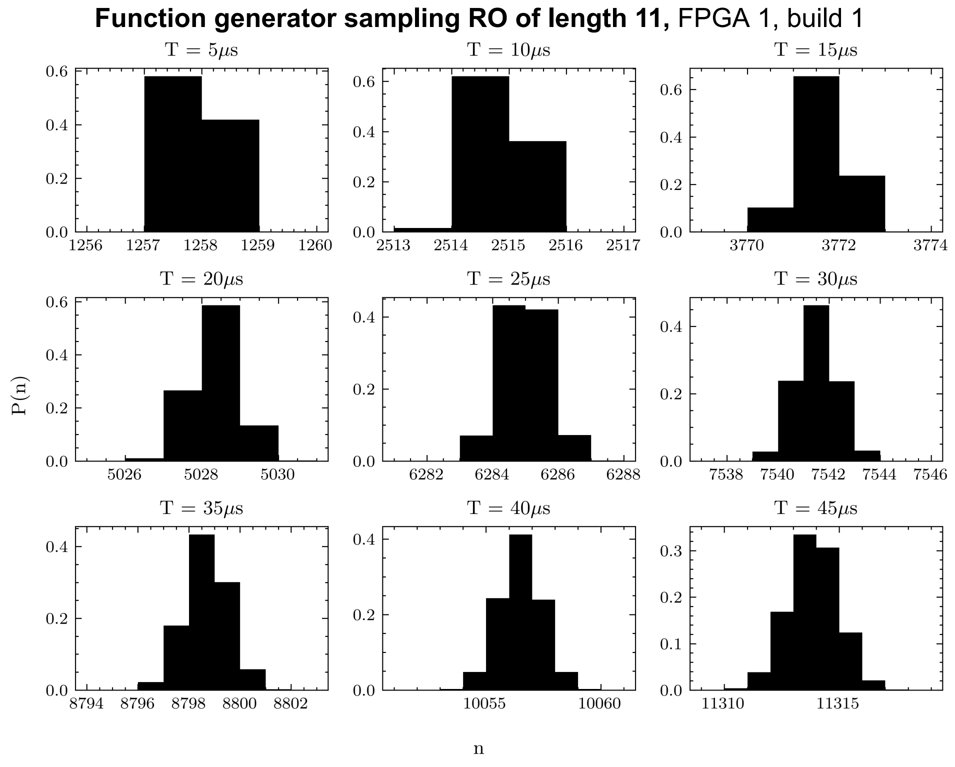

Figure B.3: Impact of sampling time T on distribution of samples.

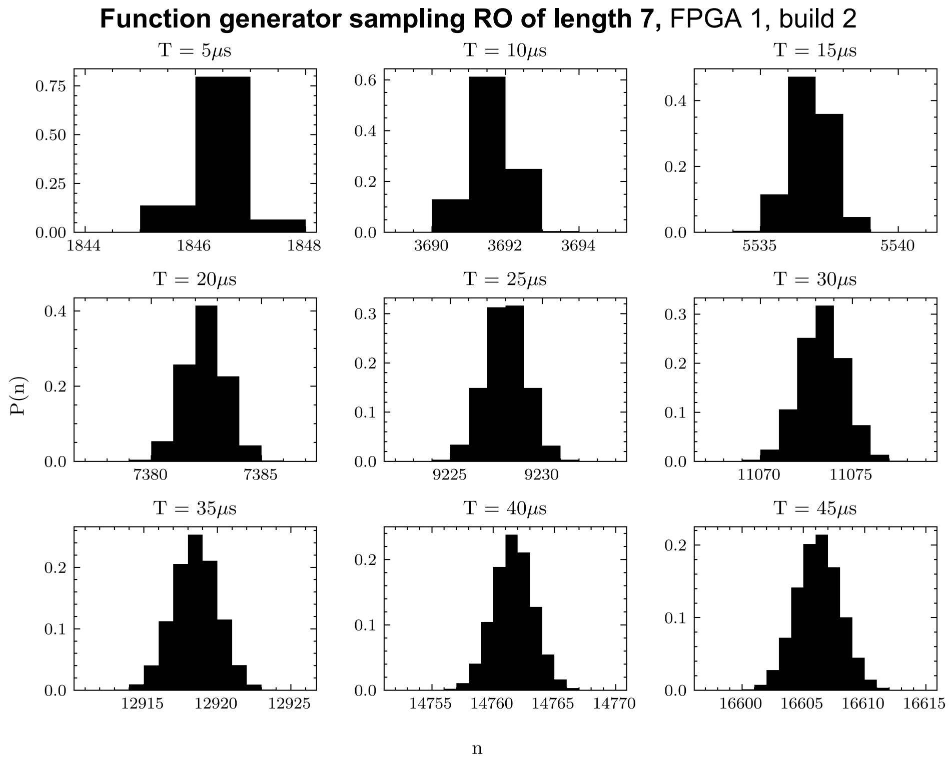

Figure B.4: Impact of sampling time T on distribution of samples.

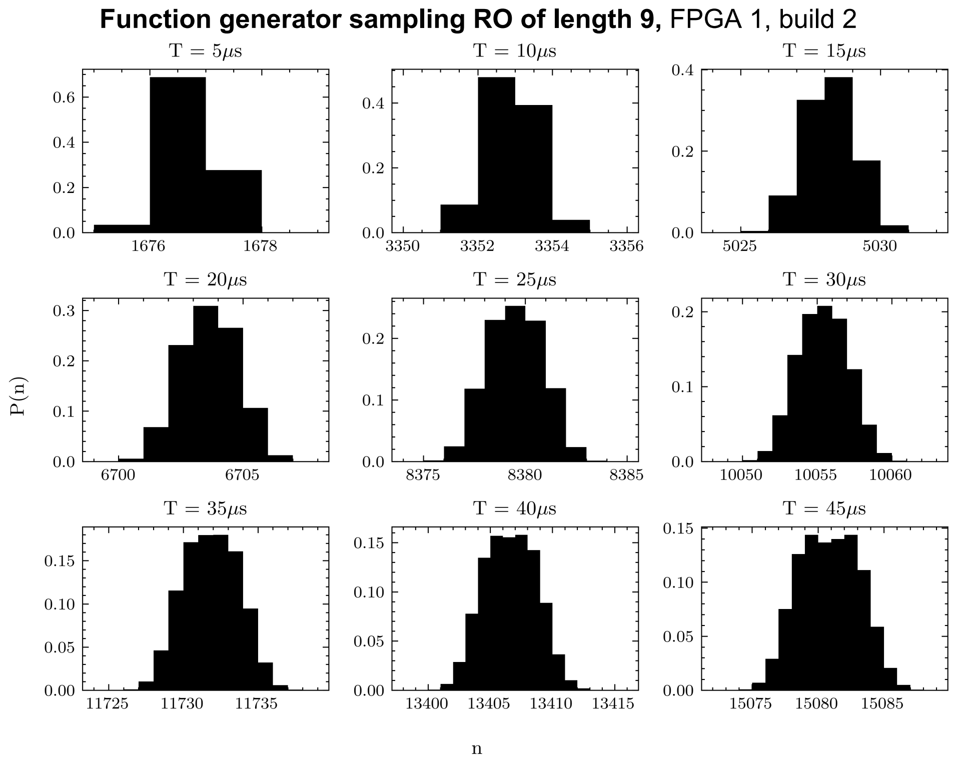

Figure B.5: Impact of sampling time T on distribution of samples.

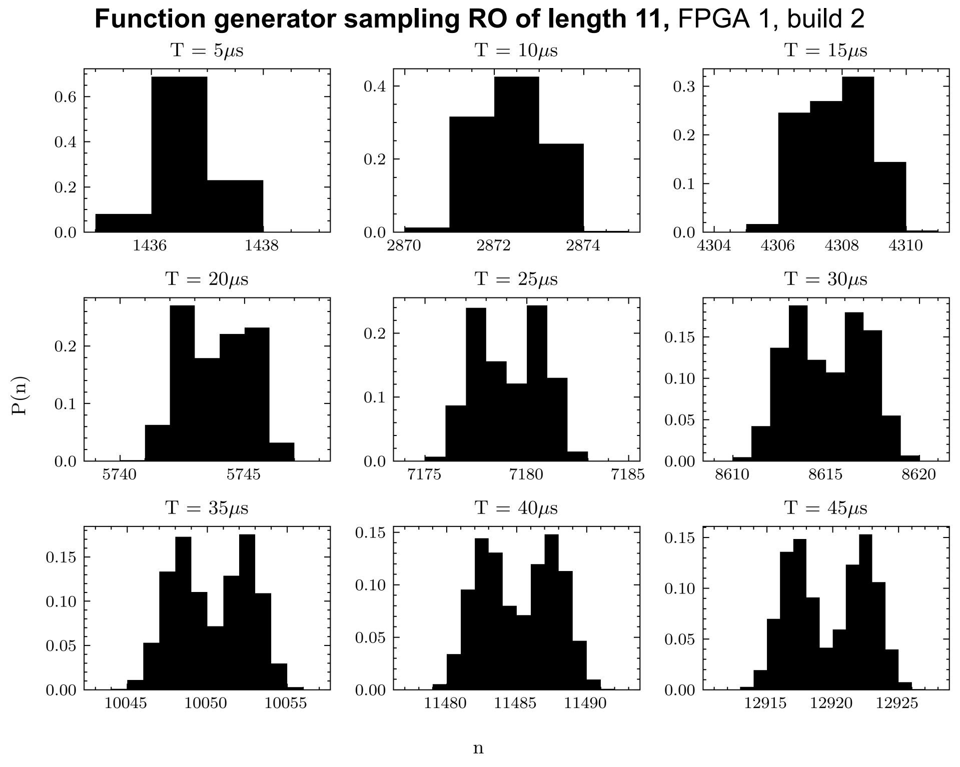

Figure B.6: Impact of sampling time T on distribution of samples.

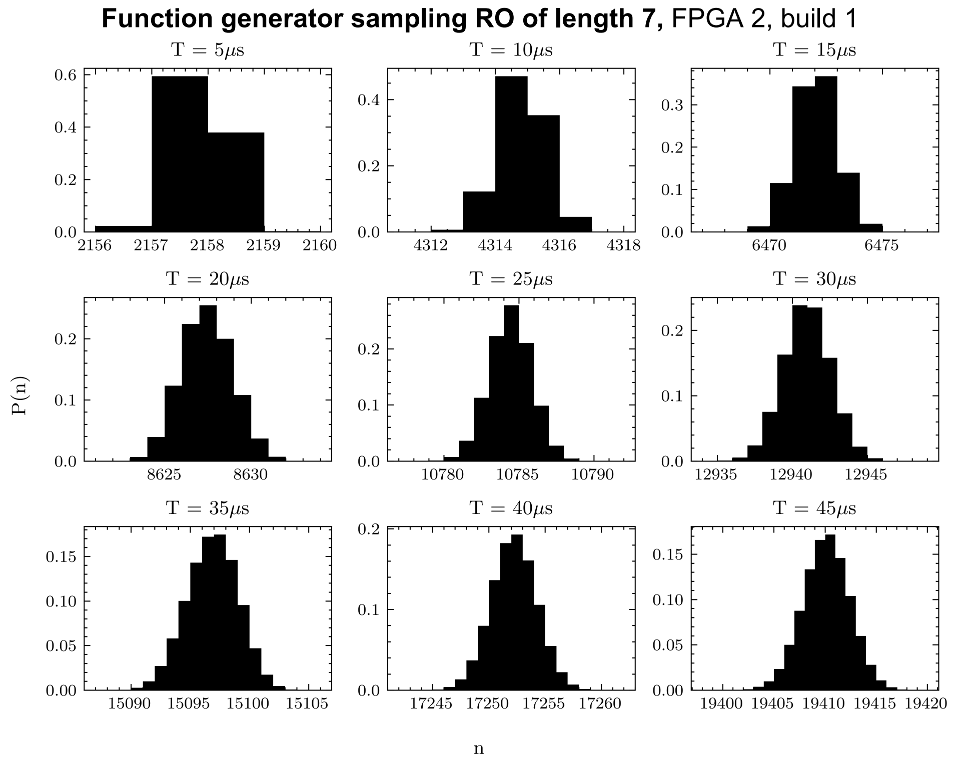

Figure B.7: Impact of sampling time T on distribution of samples.

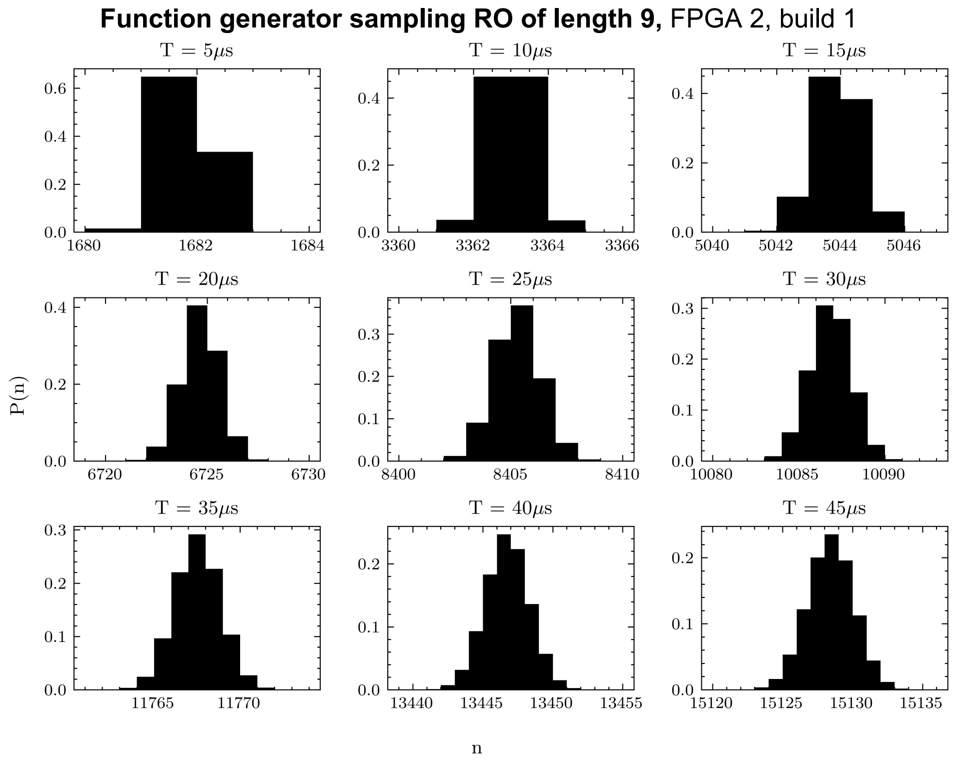

Figure B.8: Impact of sampling time T on distribution of samples.

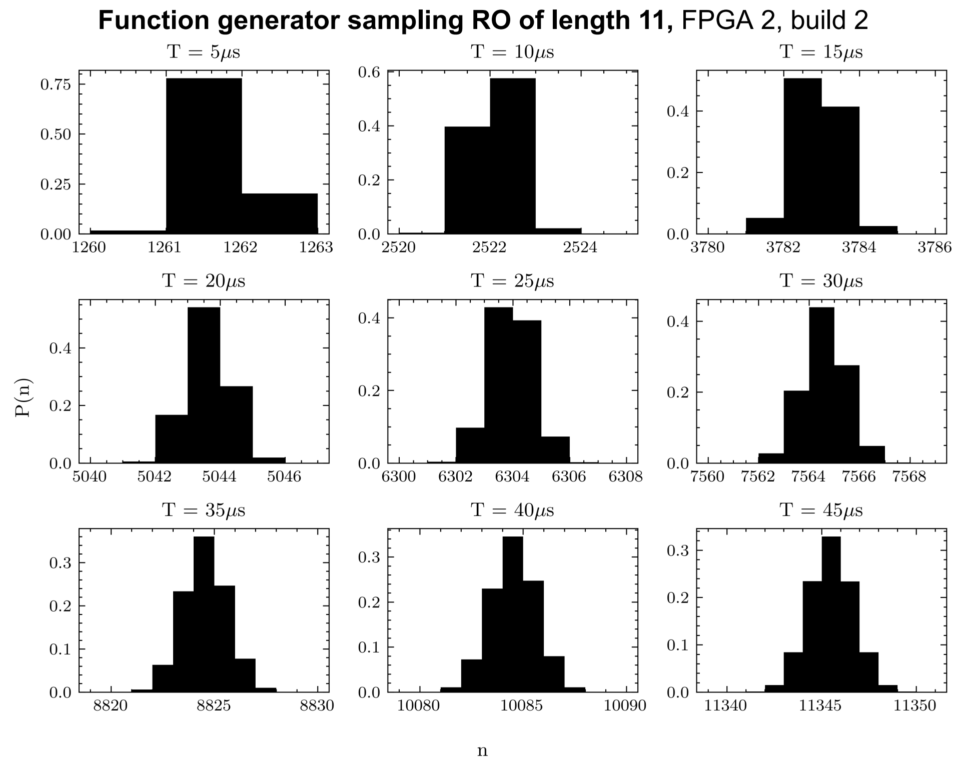

Figure B.9: Impact of sampling time T on distribution of samples.

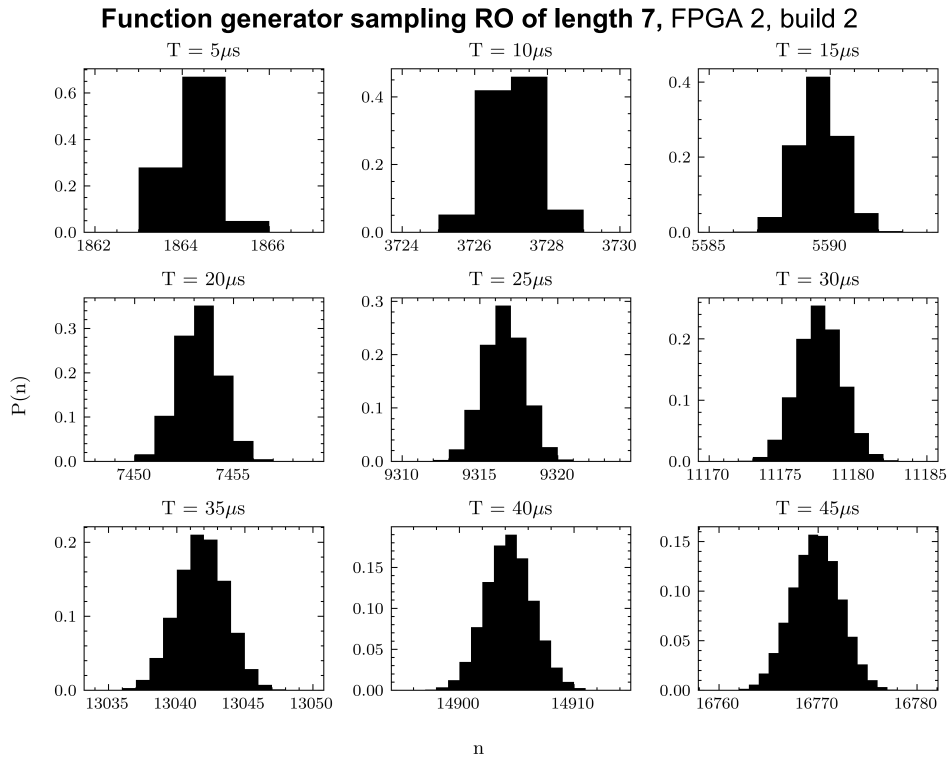

Figure B.10: Impact of sampling time T on distribution of samples.

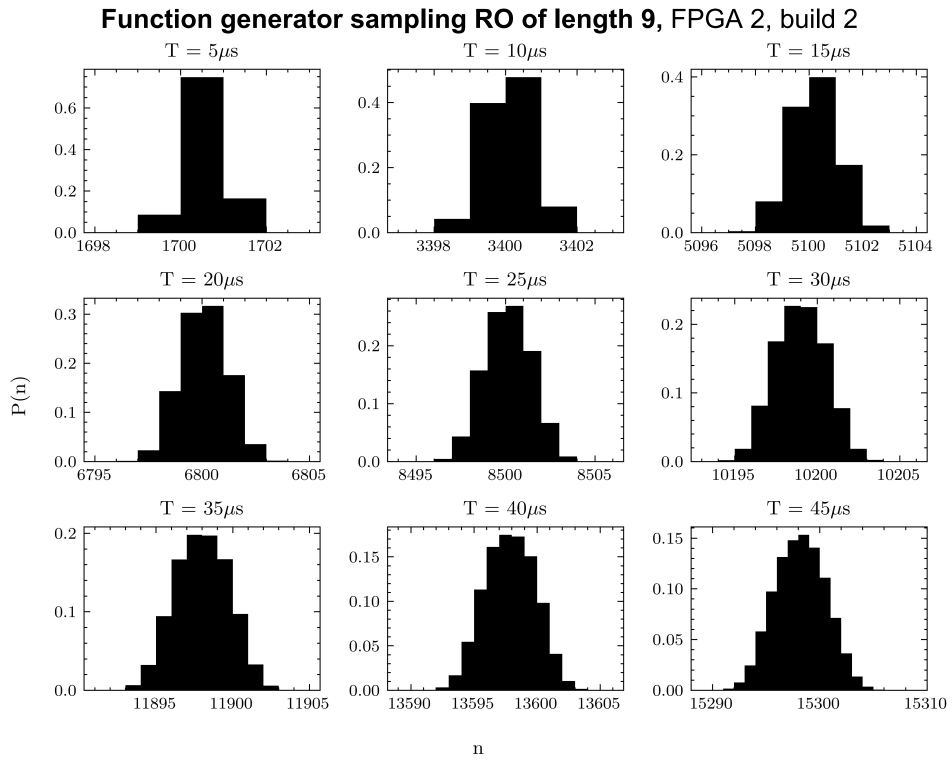

Figure B.11: Impact of sampling time T on distribution of samples.

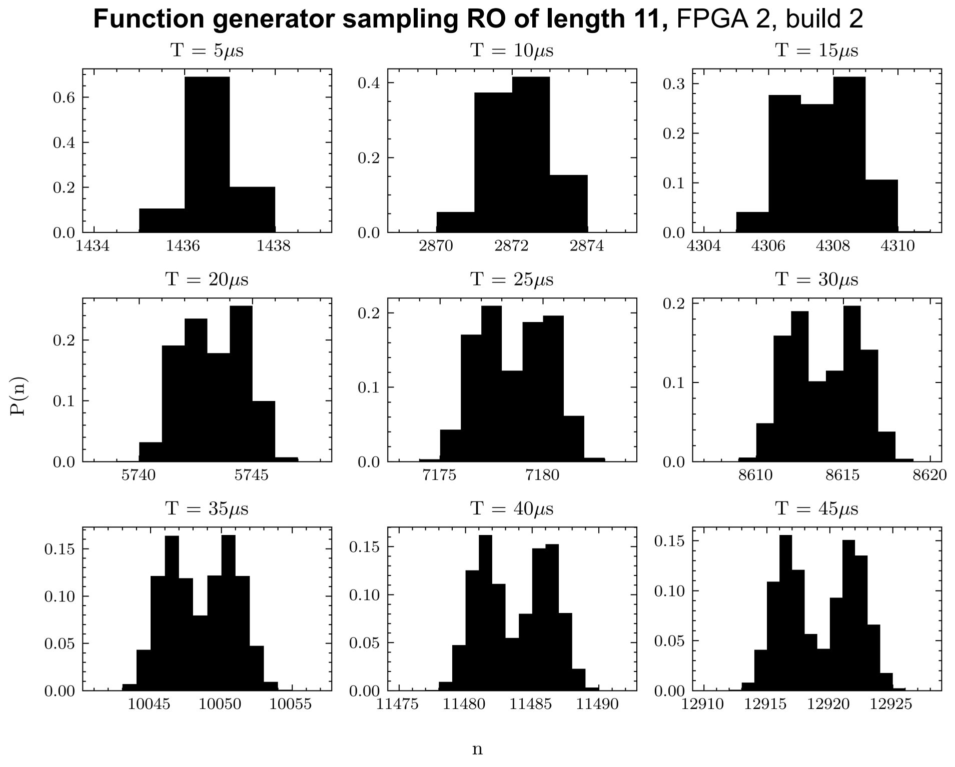

Figure B.12: Impact of sampling time T on distribution of samples.

B.2. Measurement Results PLL

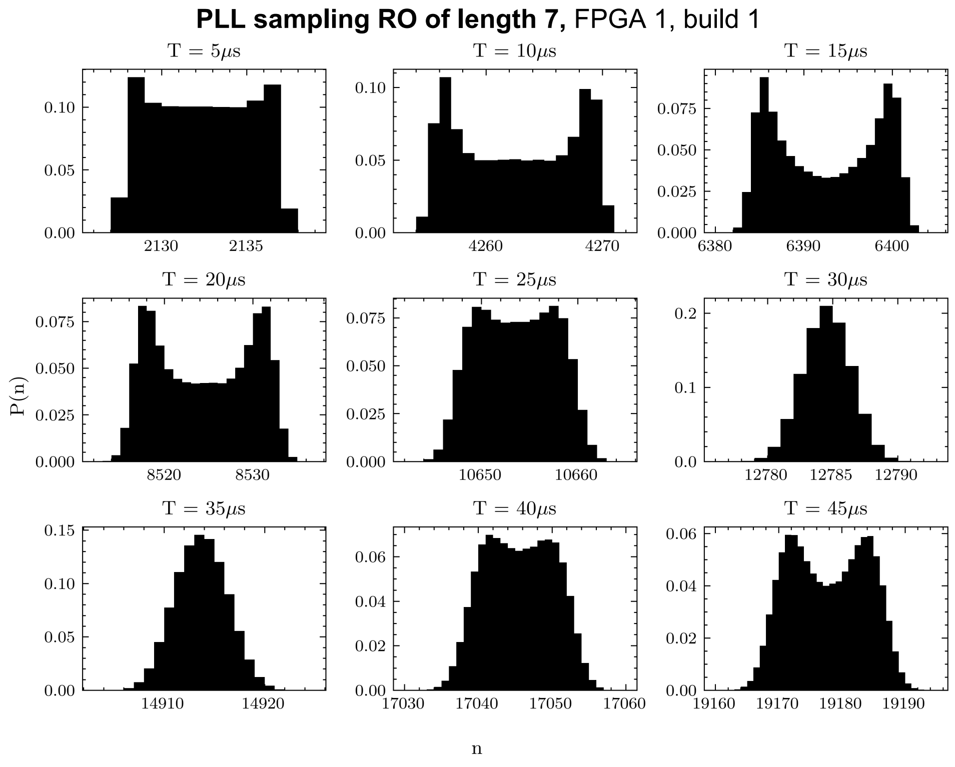

Figure B.13: Impact of sampling time T on distribution of samples.

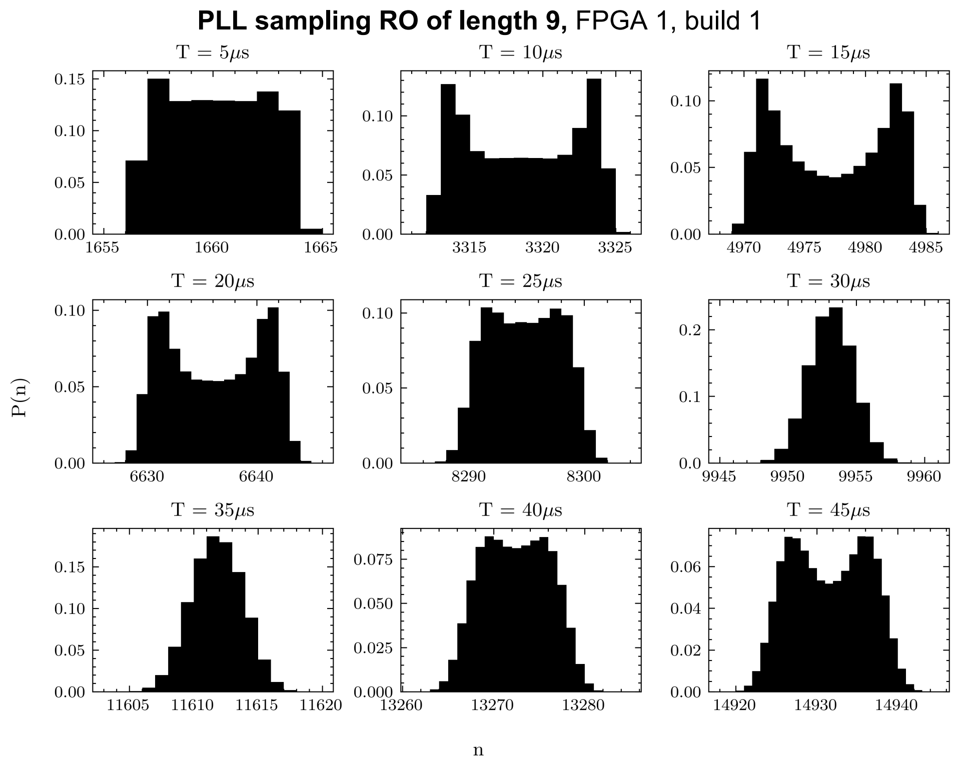

Figure B.14: Impact of sampling time T on distribution of samples.

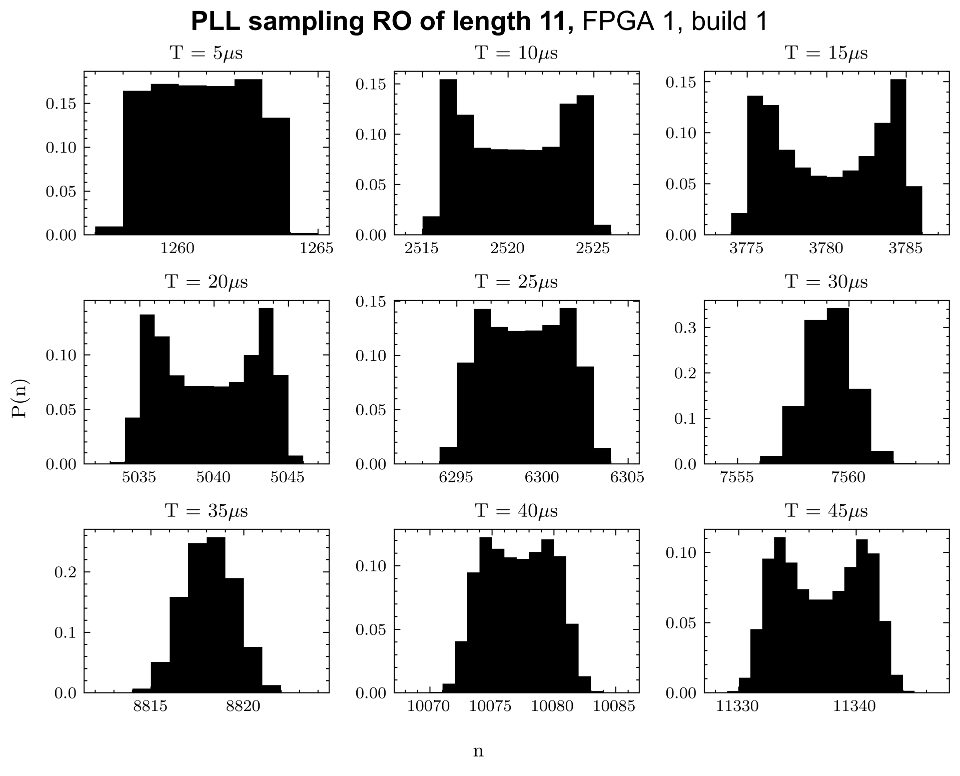

Figure B.15: Impact of sampling time T on distribution of samples.

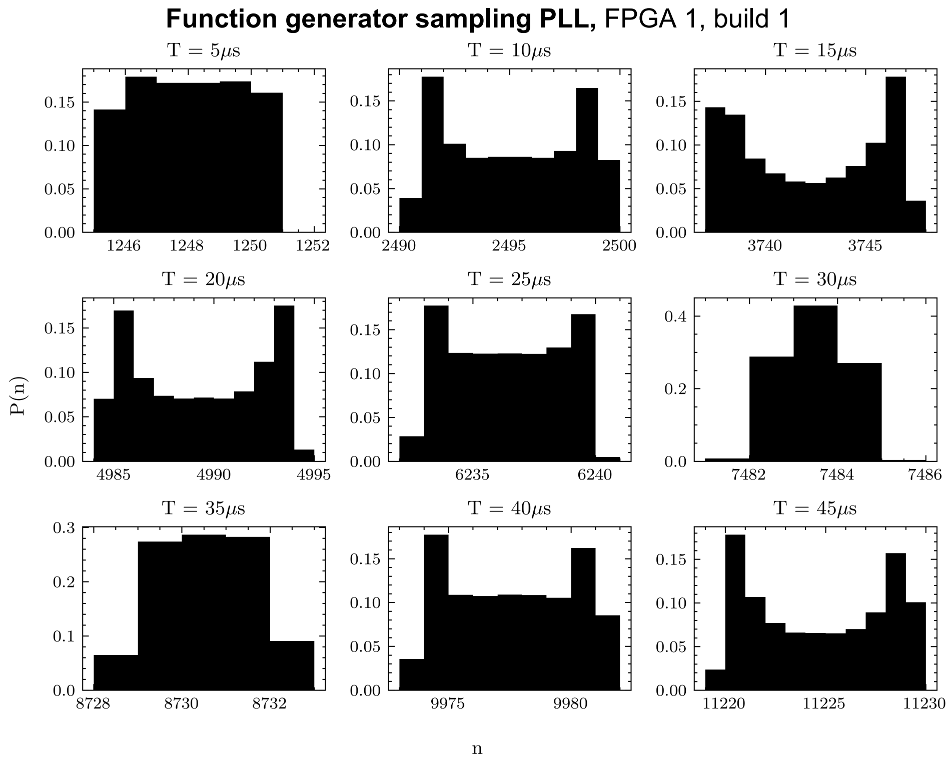

Figure B.16: Impact of sampling time T on distribution of samples.

B.3. Measurement Results Ring Oscillator

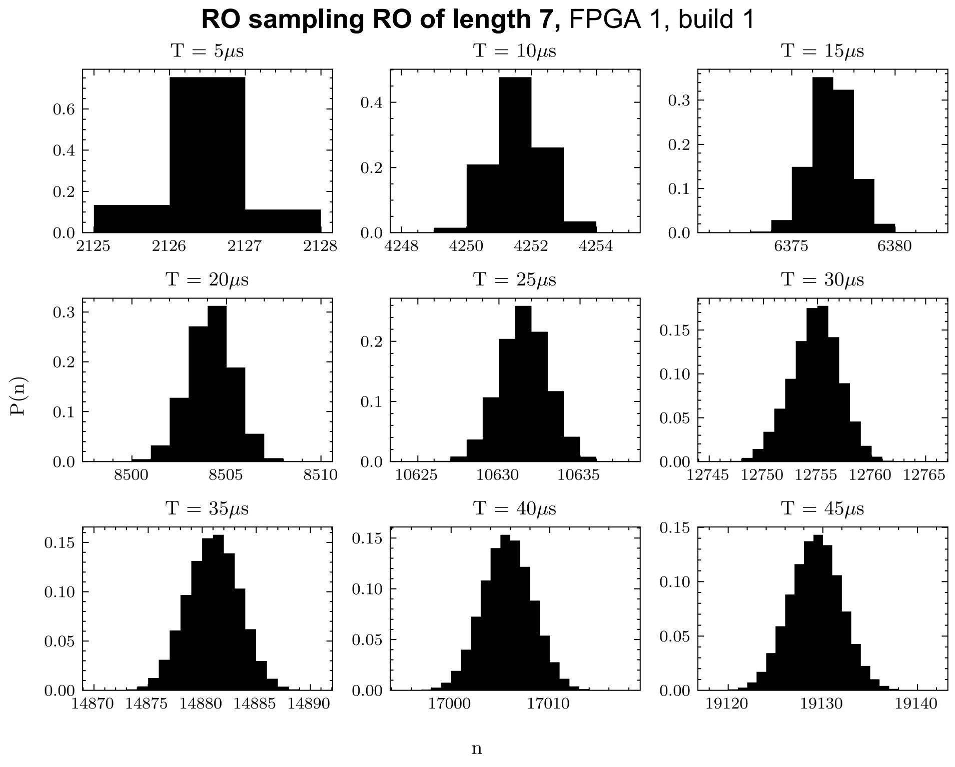

Figure B.17: Impact of sampling time T on distribution of samples.

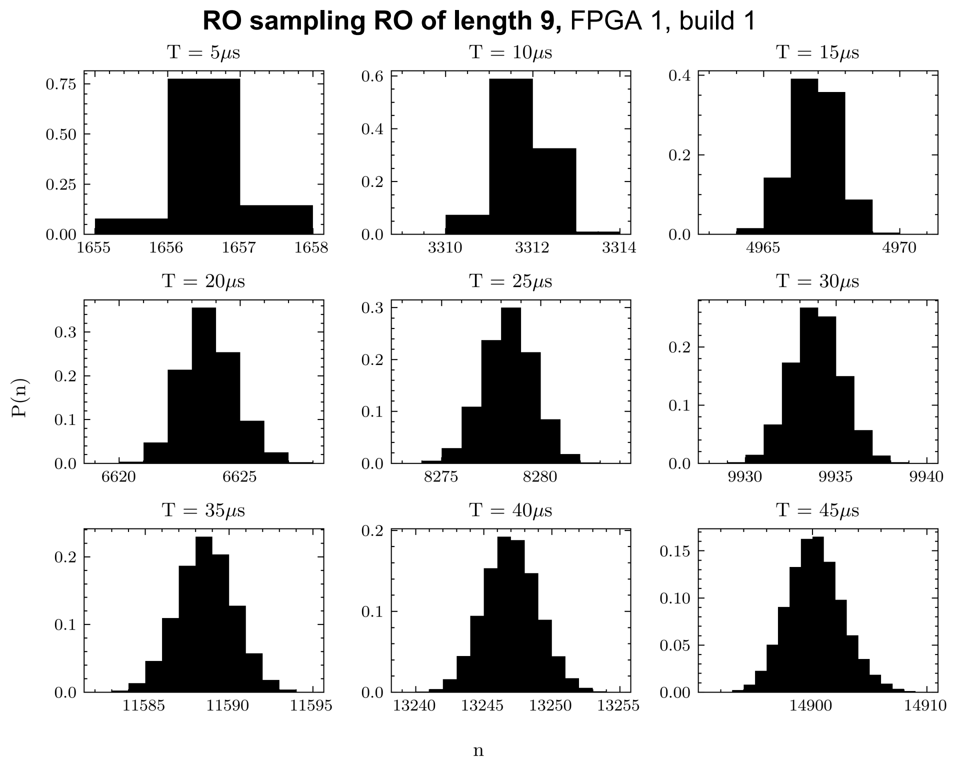

Figure B.18: Impact of sampling time T on distribution of samples.

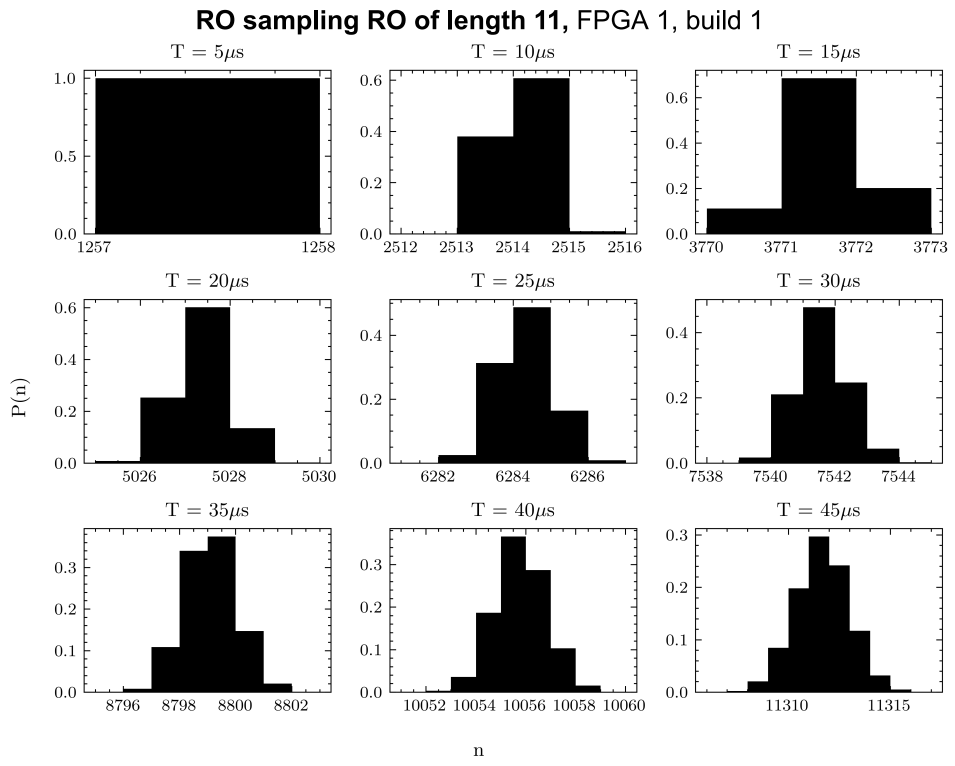

Figure B.19: Impact of sampling time T on distribution of samples.

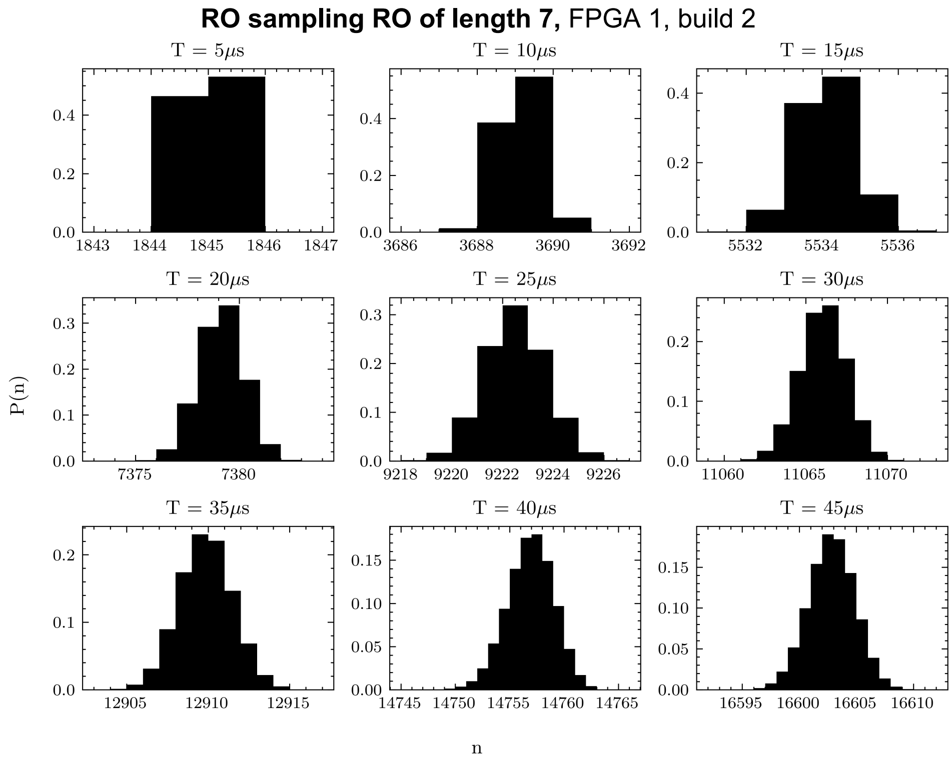

Figure B.20: Impact of sampling time T on distribution of samples.

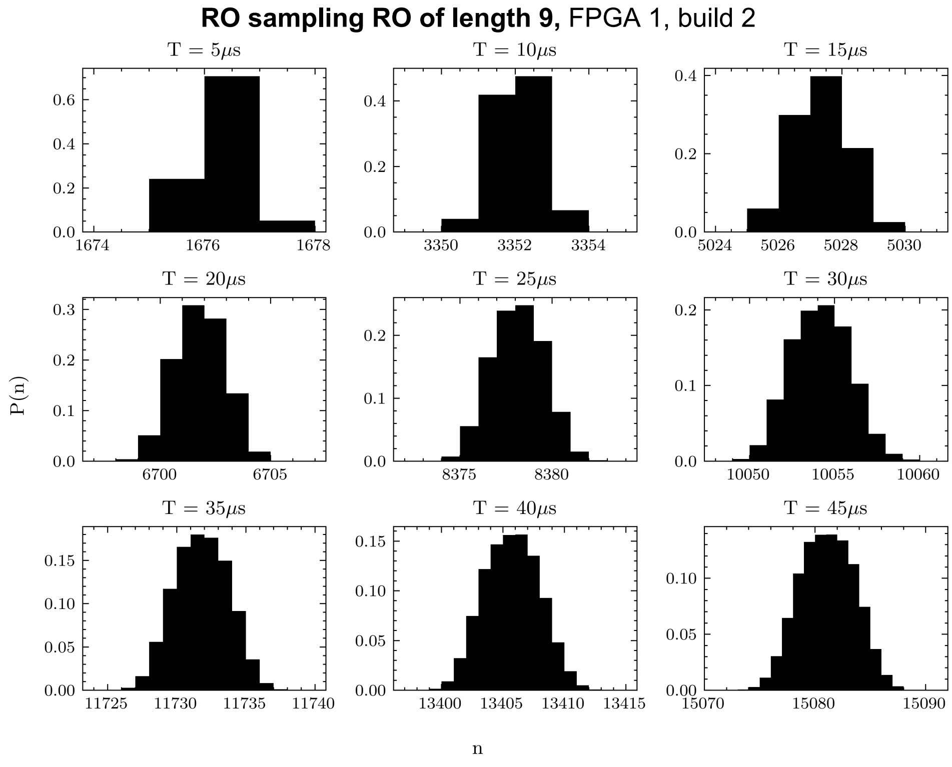

Figure B.21: Impact of sampling time T on distribution of samples.

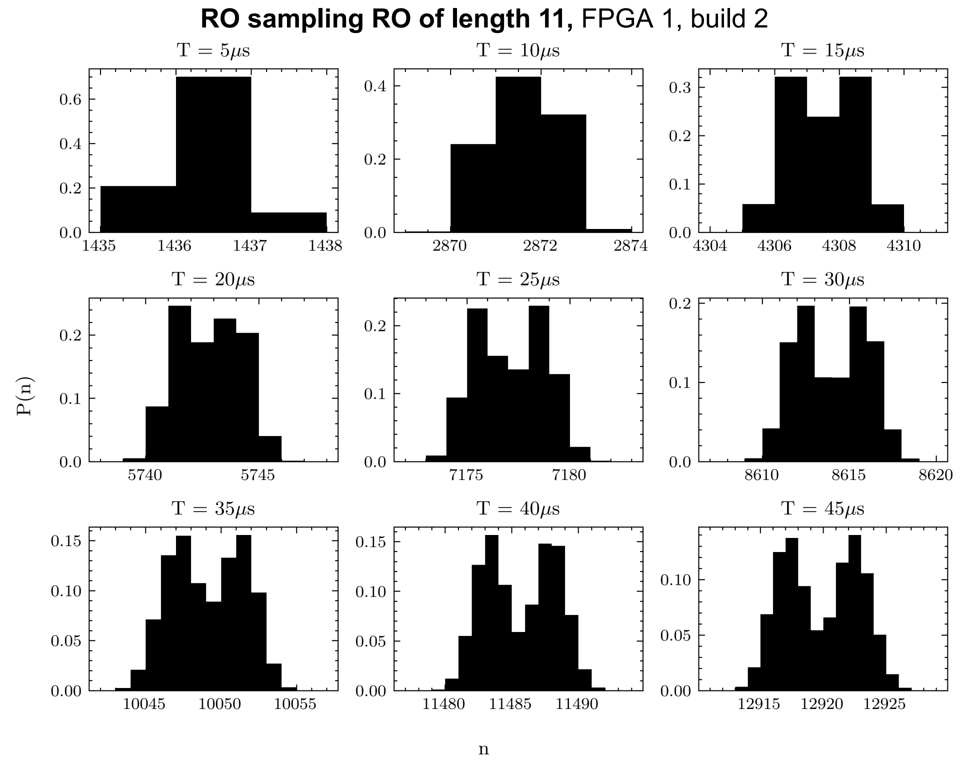

Figure B.22: Impact of sampling time T on distribution of samples.

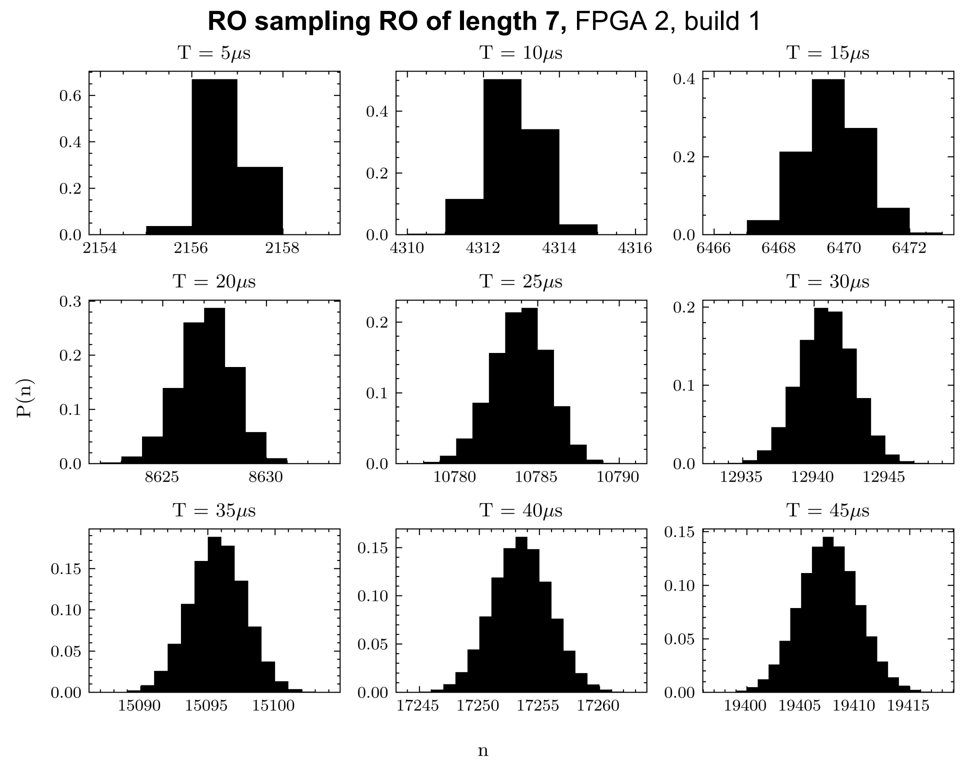

Figure B.23: Impact of sampling time T on distribution of samples.

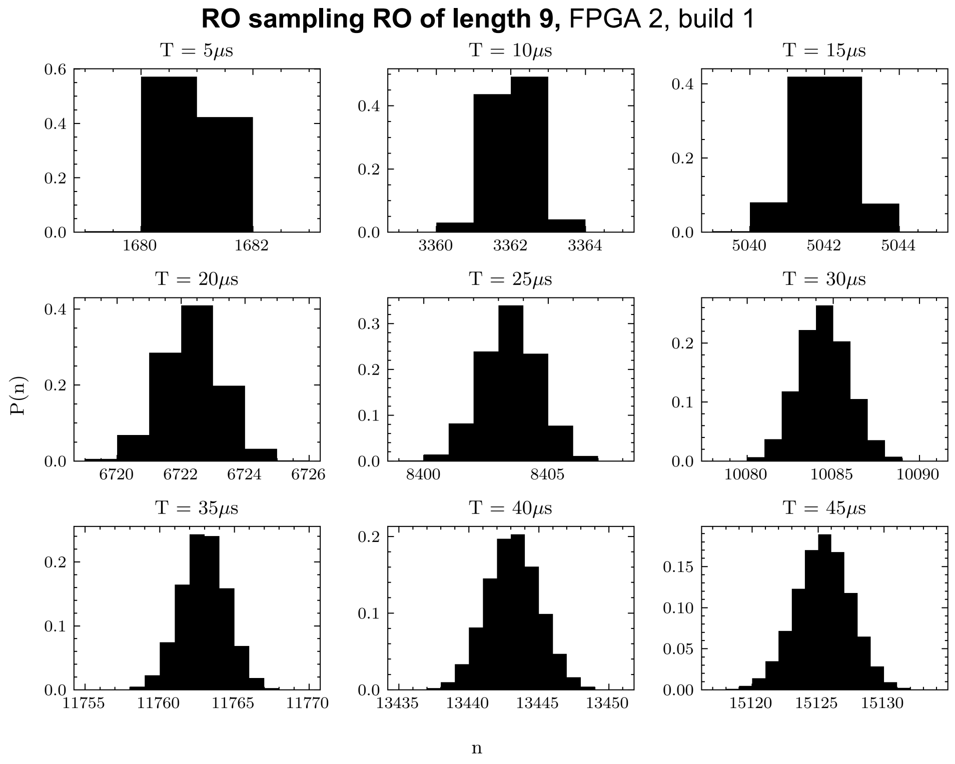

Figure B.24: Impact of sampling time T on distribution of samples.

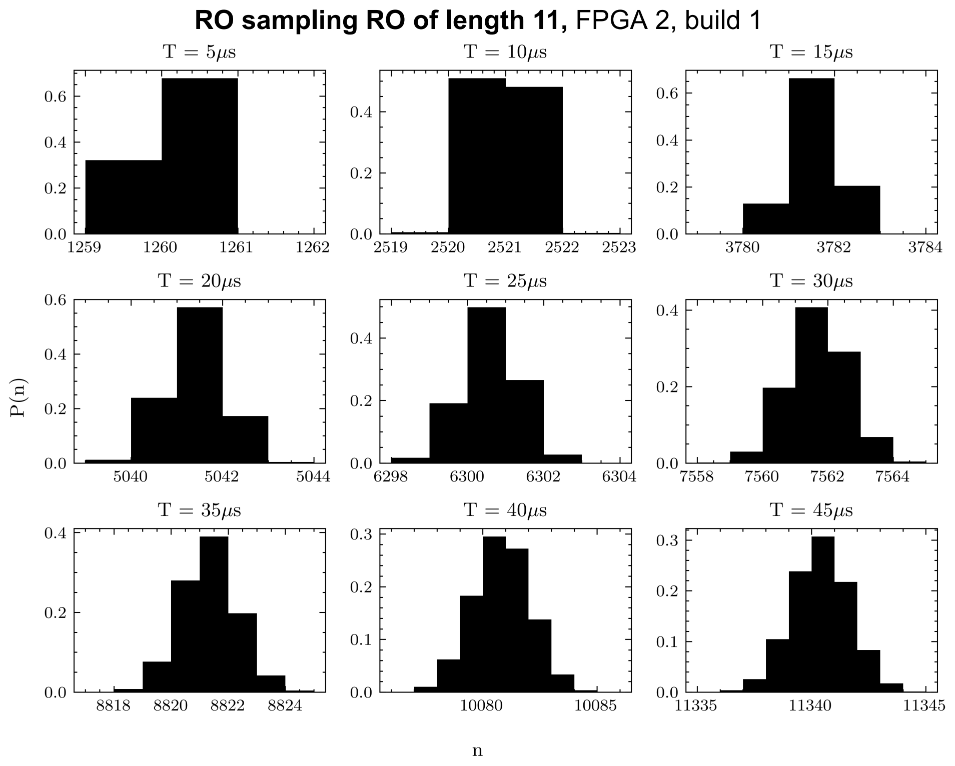

Figure B.25: Impact of sampling time T on distribution of samples.

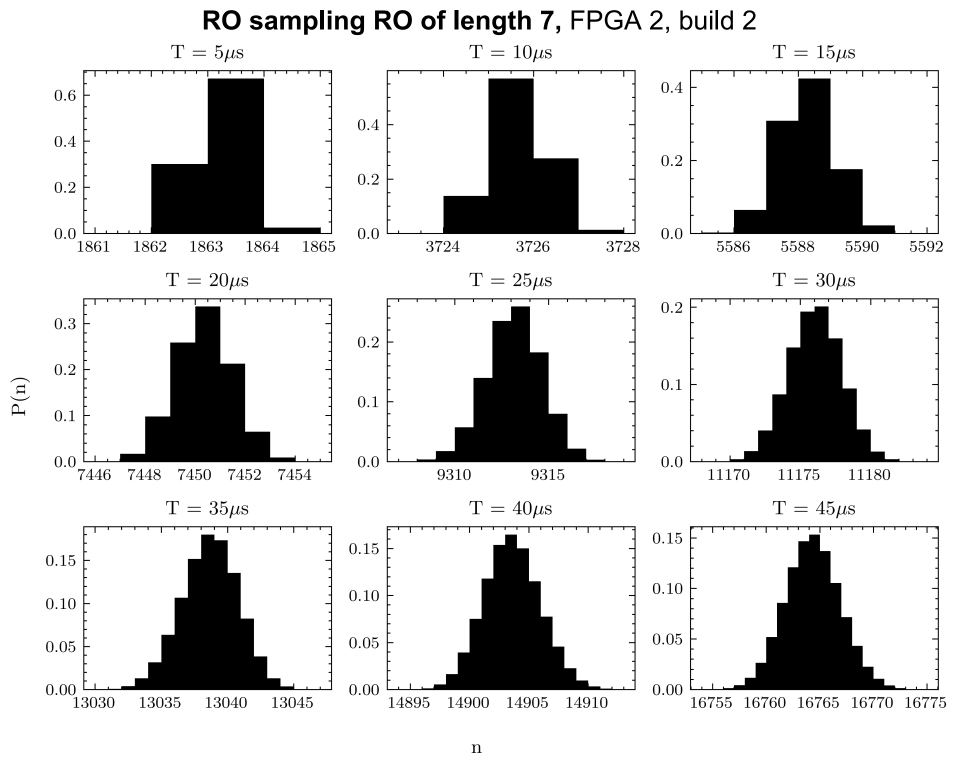

Figure B.26: Impact of sampling time T on distribution of samples.

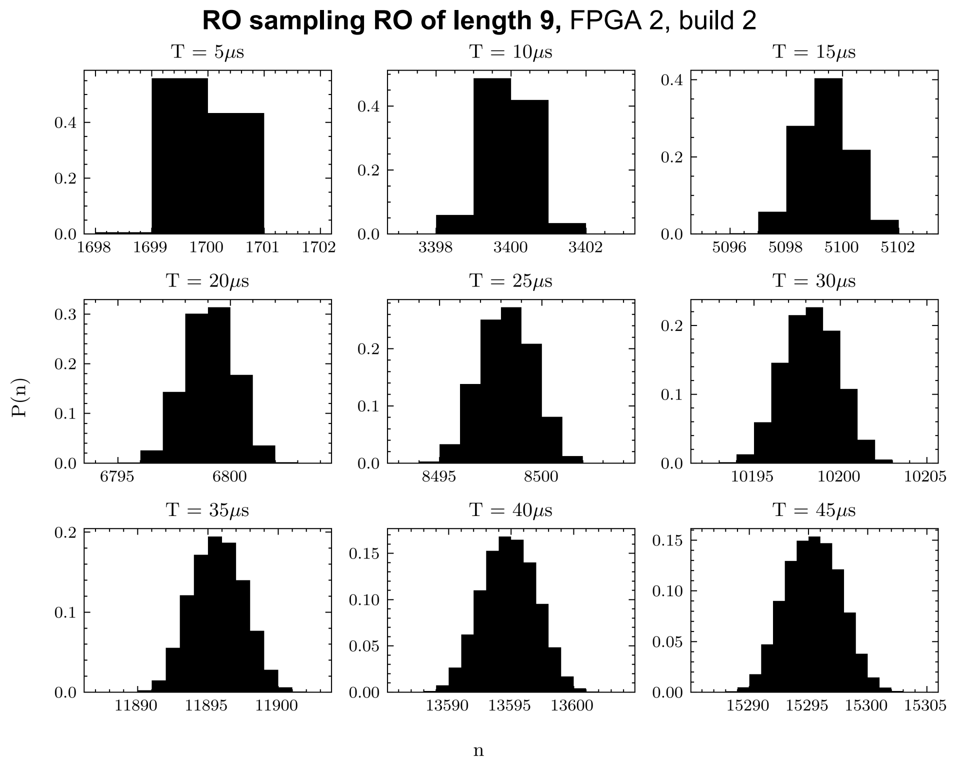

Figure B.27: Impact of sampling time T on distribution of samples.

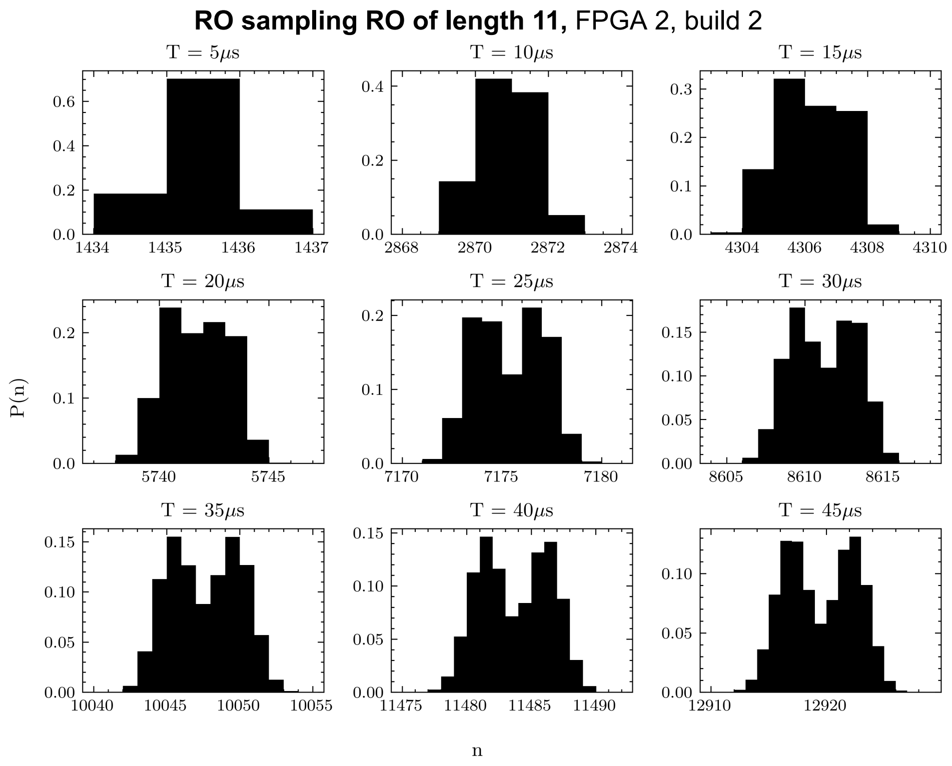

Figure B.28: Impact of sampling time T on distribution of samples.