N° d'ordre NNT : 2020LYSES019

THÈSE de DOCTORAT DE L'UNIVERSITÉ DE LYON

opéré au sein du

Laboratoire Hubert Curien

Ecole Doctorale N° 488

Sciences Ingénierie Santé

Spécialité: Informatique

Soutenue publiquement le 14 Septembre 2020 par :

Elie Noumon Allini

Caractérisation, évaluation et utilisation du jitter d'horloge comme source d'aléa dans la sécurité des données

---

Devant le jury composé de :

Bossuet, Lilian	Professeur	Université Jean Monnet	Président
Dutertre, Jean-Max	Professeur	École des Mines de Saint-Étienne	Rapporteur
Elbaz-Vincent, Philippe	Professeur	Université Grenoble-Alpes	Rapporteur
Fontaine, Caroline	Directrice de Recherche	CNRS / ENS Paris-Saclay	Examinatrice
Lubicz, David	Ingénieur de Recherche	DGA-MI / Université Rennes 1	Examinateur
Fischer, Viktor	Professeur	Université Jean Monnet	Directeur de thèse
Bernard, Florent	Maître de Conférences	Université Jean Monnet	Co-encadrant de thèse

PhD thesis from Université de Lyon

carried out at

Laboratoire Hubert Curien

Doctoral school N° 488

Sciences Engineering Health

Speciality: Computer Science

publicly defended on September 14^{th}, 2020 by:

Elie Noumon Allini

Characterization, evaluation and utilization of clock jitter as source of randomness in data security

---

In front of the jury consisting of:

Bossuet, Lilian	Professor	Université Jean Monnet	President
Dutertre, Jean-Max	Professor	École des Mines de Saint-Étienne	Reviewer
Elbaz-Vincent, Philippe	Professor	Université Grenoble-Alpes	Reviewer
Fontaine, Caroline	Senior Researcher	CNRS / ENS Paris-Saclay	Examiner
Lubicz, David	Research Engineer	DGA-MI / Université Rennes 1	Examiner
Fischer, Viktor	Professor	Université Jean Monnet	Supervisor
Bernard, Florent	Associate Professor	Université Jean Monnet	Co-supervisor

This work has received funding from the Direction Générale de l'Armement under grant agreement No 16810066.

Acknowledgements

This PhD was an enriching experience, on a personal, scientific and human dimension. I consider myself extremely privileged to have been able to carry out this experience in company of extraordinary people. I give thanks to God, the Father of my Lord Jesus Christ for this tremendous grace he has granted me.

To Viktor Fischer and Florent Bernard, I owe a big thanks for the confidence they have placed in me. I am very grateful to have benefited from their guidance during these years of doctoral studies. With them, I discovered new scientific disciplines, allowing me to increase my perspective. Their availability, their expertise have brought me a lot during these years. Their rigor pushed me beyond my limits, allowing me to improve enormously.

My gratitude also goes to DGA for the funding of my PhD thesis. My special thanks to David Lubicz who associated me with this certification work that is so important to DGA. Thank you David for your comments and questions that have greatly contributed to the quality of my work.

Many thanks to Jean-Max Dutertre and Philippe Elbaz-Vincent for reporting my work. The excellence of their remarks and comments reflects the time and care given to my work despite the difficulties related to the health crisis of Covid-19. I would also like to thank Lilian Bossuet for the wonderful welcome within the SESAM team. Thanks to this, I have been able to work serenely during these three years and devote myself exclusively to my PhD.

I would also like to thank Caroline Fontaine who introduced me to the world of research, thanks to my master's internship that she directed, she helped me open a path that led me to this PhD. Her advice and support went beyond my internship and was beneficial to me during my doctoral experience.

The major difficulty (and also the beauty) of this thesis is that it required expertise in so many fields that I lost count. I don't consider myself an expert in these fields, I had the grace of bene-

fiting from the expertise of the greats in these fields. Among these experts, I can name François Vernotte and Enrico Rubiola who helped me enormously to understand frequency stability. Chapter 2 of this thesis attests of their availability and the enriching discussions I had with them. You have my full acknowledgement.

My heartfelt thanks to Jean-Jacques Rousseau, who helped me enormously to get up to speed in electronics and automation. Our recurring discussions have given me a better understanding of this world that was entirely new to me. Thanks to this, I was able to understand better how filters and PLL work. It is therefore obvious that you played an important part in the success of my thesis. Many thanks to you JJR.

I would like to thank all the members of the SESAM team in which I carried out my thesis work. The good attitude and availability of each of its members contributed to a healthy working environment. Special thanks to Nathalie Bochard for her help with the electronic aspects, as well as for her dynamism. I was honored to share the same office with Brice Colombier, Ugo Murredu and Oto Petura. I hope I didn't traumatize them with my endless equations on the board. I will miss the atmosphere in this office and our discussions.

I also thank Alain Aubert, Pierre-Louis Cayrel and Vincent Grosso for their simplicity. A special mention to El Mehdi Benhani, our TrustZone expert, who started his PhD and completed it almost at the same time as me. I wish you a very good career, and by the way, a happy marriage. I do not forget of course to Damien Robissout and Gabriel Zaïd (affectionately Titi) the experts of the team in Deep Learning and side channel and wish them a good completion of their PhD.

My gratitude also goes to my family, who have been an unfailing source of support. Your encouragement, advice and constant presence have been crucial to the success of this thesis and I am deeply grateful for that.

I also thank Julia Leute, Olivier Marchal, and many others. They were so numerous to have contributed to the success of this PhD. Many thanks to all of you and may God bless you endlessly.

Contents

Introduction

Throughout history, information has proved itself to be an increasingly valuable asset. From it depended government decisions, military actions, business prospects and so much more [1]. This reflects the essential importance of information and, at the same time, shows how important it is to protect it. A common way of protecting information is cryptography [2, 3]. Indeed, by encrypting the information, it becomes inaccessible to any unauthorized party. The encryption process involves mathematical algorithms for which prime numbers are very important [3, Chapter 4]. However, there is a second class of numbers almost as important as prime numbers, namely random numbers [4, 5, 6].

Modern cryptography is based on a set of fundamental principles, one of which is the Kerckhoff Principles, which stipulate, among other things, that the security of any cryptographic construction must not be compromised even if everything related to that construction is made public, except the key [7]. This implies that the security of any cryptographic construction must be based solely on secret keys, and thus imposes high requirements on them. It is therefore understood that a cryptographic key can under no circumstances be exported outside the cryptosystem in clear. It is also essential that it is stored in a secure area to prevent unauthorized access to this key. However, protecting access to this key would be useless if an adversary can easily guess it. These keys have to be, in addition, unpredictable. This unpredictability is only accessible through random numbers, which are unpredictable in nature. We therefore need a process to generate random numbers, such a process is known as a random number generator (RNGs) [8, Section 5.1.1].

The random number generators can be divided into two main families, depending on the methods on which they are based. The first family is that of pseudo-random number generators . They are based on public algorithmic, and therefore deterministic, methods using an initial input called the seed [8, Definition 5.3]. They have high throughput and produce sequences numbers with good statistical properties [9, Chapter 1]. They are generally used as key generators in stream ciphers [10]. Due to the existence of an underlying algorithm, PRNGs are easy to implement in logical devices. However, because the algorithm is known, if the seed is not properly chosen the output

of the generator is predictable.

The second family of generators is the one using non-deterministic phenomena to generate true random numbers [11]. This justifies why they are called true random number generators (TRNGs). The idea for these types of generators is that if the underlying random phenomenon cannot be controlled, the output of the generator is unpredictable and/or uncontrollable. These phenomena can be physical (electronic noise, radioactive noise, etc) or non-physical (system time, disk operations, etc). The throughput of TRNGs is generally lower than that of PRNGs [12, Section 2.1.2], as it is limited by the phenomenon exploited and by the principle of entropy extraction. Due to their operating principle, the statistical characteristics of the TRNG outputs are closely related to the quality of the source of entropy, but also to the entropy extraction method [13].

Despite their lower throughput, TRNGs are often preferred for secure applications [14]. Indeed, they offer the possibility to have a higher entropy per bit compared to PRNGs [11, Section 2.5.1]. This advantage of TRNGs thus makes it possible to achieve a better level of security. However, before using a generator in practice, it is imperative that its principle and implementation within a cryptographic module are validated during an evaluation process [11, Section 3.3]. As the purpose of this process is to certify TRNGs as secure for cryptographic applications, it is reasonable to consider unsecured any generator that has not obtained a security certificate.

However, it should be noted that until late 1990s, TRNGs did not have any standard evaluation criteria [15, Section A.1]. In order to remedy this situation, government agencies have published criteria to be used as a reference in assessing the safety of TRNGs. To date, two standards are widely used. The first, published by the NIST [16, 17, 18], consists of a series of statistical tests applied to the output of the generator. The purpose of these tests is to determine whether or not the output of the generator appears random. However, it is possible to construct sequences that appear random using deterministic methods. The use of statistical methods is therefore insufficient to properly assess the safety of TRNGs. It is important to also take into account the design of the generator when evaluating it [19].

Such an evaluation of the security of a TRNG is a complex problem. Indeed, it requires to understand the mechanism of accumulation of the entropy of the underlying physical phenomenon and to characterize its extraction. The objective being to guarantee an entropy rate per bit close to 1 [11, Section 3.3]. Since entropy is a property related to random variables and not their realizations [13], it is necessary to propose a stochastic model of the TRNG (characterization of the digital noise source). This approach is the one published by the BSI (Bundesamt für Sicherheit in

der Informationstechnik), which is the standard used by default in Europe.

In order to guarantee the security of highly sensitive applications, such as military applications, the DGA-MI (Direction Générale de l'Armement-Maîtrise de l'Information) aims to propose an extension of the BSI approach. The purpose of this extension is to characterize the source of analog noise, as well as the various phenomena that occur there. This characterization is intended to lead to a stochastic model of physical noise in order to better understand its evolution. The DGA-MI also requires that measurement principles, compatible with entropy extraction methods, are developed, characterized and implemented. These measurement principles are intended to ensure that only the desired phenomena are used to generate random numbers.

David Lubicz, from DGA-MI, illustrated this approach on the elementary TRNG based on ring oscillators. In order to study its applicability, the DGA-MI desired its approach to be illustrated with TRNGs based on other principles. For this reason, the DGA-MI funded this thesis to study the applicability of their approach to TRNGs based on PLLs. During this thesis, we therefore studied PLL-based generators, in connection with the DGA-MI certification approach.

Objectives

- study of the DGA-MI certification approach aimed at certifying TRNGs for ultra-secure applications;
- suggestion of an embedded jitter measurement method ensuring that the measured quantity comes from the source of randomness;
- accurate estimate of the jitter proportion due to thermal noise;
- study of PLL as a source of randomness in order to:
  - determine the PLL settling time,
  - identify the influence of the different parameters of the PLL on the quality of the randomness,
  - determine the level of dependency between jitter realizations,
  - choose the parameters of the PLL that reduce or eliminate the influence of deterministic jitter,
  - assess the assumptions made to improve the stochastic model of the PLL-based generator.

Tout au long de l'histoire, l'information s'est révélée être un atout de plus en plus précieux. De celle-ci dépendaient les décisions gouvernementales, les actions militaires, les perspectives commerciales et bien plus encore [1]. Cela reflète l'importance essentielle de l'information et, en même temps, montre à quel point il est important de la protéger. Un moyen fréquemment utilisé pour protéger l'information est la cryptographie [2, 3]. En effet, en chiffrant l'information, celle-ci devient inaccessible à toute partie non autorisée. Le processus de chiffrement fait appel à des algorithmes mathématiques pour lesquels les nombres premiers sont très importants [3, Chapitre 4]. Cependant, il existe une deuxième classe de nombres presque aussi importante que les nombres premiers, à savoir les nombres aléatoires [4, 5, 6].

La cryptographie moderne repose sur un ensemble de principes fondamentaux, au nombre desquels figurent les principes de Kerckhoff, qui stipulent, entre autres, que la sécurité de toute construction cryptographique ne doit pas être compromise même si tout ce qui a trait à cette construction est rendu public, à l'exception de la clé [7]. Cela implique que la sécurité de toute construction cryptographique doit être basée uniquement sur des clés secrètes, ce qui impose des exigences strictes à leur égard. Il est donc compréhensible qu'une clé cryptographique ne puisse en aucun cas être exportée en clair en dehors du système cryptographique qui l'utilise. Il est également essentiel qu'elle soit stockée dans une zone sécurisée afin d'empêcher tout accès non autorisé à cette clé. Cependant, il serait inutile de protéger l'accès à cette clé si un adversaire peut facilement la deviner. Ces clés doivent, en outre, être imprédictibles. Cette imprédictibilité n'est accessible qu'à travers des nombres aléatoires, qui sont par nature impossibles à deviner. Nous avons donc besoin d'un processus permettant de générer des nombres aléatoires, un tel processus est connu sous le nom de générateur de nombres aléatoires (abrégé en RNG, de l'anglais Random Number Generator) [8, Section 5.1.1].

Les générateurs de nombres aléatoires peuvent être répartis en deux familles, selon les méthodes sur lesquelles ils sont basés. La première famille est celle des générateurs de nombres pseudo-aléatoires (abrégé en PRNG, de l'anglais Pseudo-Random Number Generator). Ils sont basés sur des méthodes algorithmiques, et par conséquent déterministes, publiques utilisant une entrée initiale appelée "graine" [8, Définition 5.3]. Ils ont un débit élevé et produisent des suites de nombres ayant de bonnes propriétés statistiques [9, Chapitre 1]. Ils sont généralement utilisés comme générateurs de clés dans les algorithmes de chiffrement par flot [10]. En raison de l'existence d'un algorithme sous-jacent, les PRNGs sont faciles à implanter dans les circuits logiques. Cependant, comme l'algorithme est connu, si la graine n'est pas correctement choisie, la sortie du générateur devient alors prédictible.

La seconde famille de générateurs est celle qui utilise des phénomènes non déterministes pour générer de véritables nombres aléatoires [11]. C'est pourquoi on les appelle des générateurs de nombres aléatoires véritables (abrégé en TRNG, de l'anglais True Random Number Generator). L'idée de ces types de générateurs est que si le phénomène aléatoire sous-jacent ne peut être contrôlé, la sortie du générateur est imprévisible et/ou incontrôlable. Ces phénomènes peuvent être physiques (bruit électronique, bruit de radioactivité, etc.) ou non physiques (horloge du système, opérations sur le disque, etc.). Le débit des TRNGs est généralement inférieur à celui des PRNGs [12, Section 2.1.2], car il est limité par le phénomène exploité et par le principe d'extraction de l'entropie. En raison de leur principe de fonctionnement, les caractéristiques statistiques des sorties de TRNGs sont étroitement liées à la qualité de la source d'entropie, mais aussi à la méthode d'extraction de l'entropie [13].

Malgré leur débit plus faible, les TRNGs sont souvent préférés pour les applications sécurisées [14]. En effet, ils offrent la possibilité d'avoir une entropie par bit plus élevée que celle des PRNGs [11, Section 2.5.1]. Cet avantage des TRNGs permet donc d'atteindre un meilleur niveau de sécurité. Toutefois, avant d'utiliser un générateur en pratique, il est impératif que son principe et son implantation au sein d'un module cryptographique soient validés lors d'un processus d'évaluation [11, Section 3.3]. Le but de ce processus étant de certifier la fiabilité des TRNGs pour les applications cryptographiques, il est raisonnable de considérer comme non fiable tout générateur qui n'a pas obtenu de certificat de sécurité.

Toutefois, il convient de noter que jusqu'à la fin des années 1990, les TRNGs ne disposaient pas de critères d'évaluation standard [15, Section A.1]. Afin de remédier à cette situation, les agences gouvernementales ont publié des critères à utiliser comme référence pour l'évaluation de la sécurité des TRNGs. À ce jour, deux normes sont largement utilisées. La première, publiée par le NIST [16, 17, 18], consiste en une série de tests statistiques appliqués à la sortie du générateur. Le but de ces tests est de déterminer si la sortie du générateur paraît aléatoire ou non. Cependant, il est possible de construire des suites de nombres qui paraissent aléatoires en utilisant des méthodes déterministes. L'utilisation de méthodes statistiques est donc insuffisante pour évaluer correctement la sécurité des TRNGs. Il est important de prendre également en compte la conception du générateur lors de son évaluation [19].

Une telle évaluation de la sécurité d'un TRNG est un problème complexe. En effet, elle nécessite de comprendre le mécanisme d'accumulation de l'entropie du phénomène physique sous-jacent et de caractériser son extraction. L'objectif est de garantir un taux d'entropie par bit proche de 1 [11, Section 3.3]. L'entropie étant une propriété liée à des variables aléatoires et non à leurs

réalisations [13], il est nécessaire de proposer un modèle stochastique du TRNG (caractérisation de la source de bruit numérique). Cette approche est celle publiée par le BSI (Bundesamt für Sicherheit in der Informationstechnik), qui est la norme utilisée par défaut en Europe.

Afin de garantir la sécurité des applications très sensibles, telles que les applications militaires, la DGA-MI (Direction Générale de l'Armement-Maîtrise de l'Information) a pour objectif de proposer une extension de l'approche BSI. Cette extension a pour but de caractériser la source de bruit analogique, ainsi que les différents phénomènes qui s'y produisent. Cette caractérisation doit conduire à un modèle stochastique du bruit physique afin de mieux comprendre son évolution. La DGA-MI exige également que des principes de mesure, compatibles avec les méthodes d'extraction de l'entropie, soient développés, caractérisés et implantés. Ces principes de mesure visent à garantir que seuls les phénomènes souhaités sont utilisés pour générer des nombres aléatoires.

David Lubicz, de la DGA-MI, a illustré cette approche sur le TRNG élémentaire basé sur des oscillateurs en anneau. Afin d'étudier la faisabilité de cette approche, la DGA-MI a souhaité qu'elle soit illustrée par des TRNGs basés sur d'autres principes. Pour cette raison, la DGA-MI a financé cette thèse pour étudier les possibilités d'application de son approche aux TRNGs basés sur les PLLs. Dans le cadre de cette thèse, nous avons donc étudié les générateurs à base de PLL, en lien avec la démarche de certification de la DGA-MI.

Objectifs de la thèse

- étude de la démarche d'évaluation de la DGA-MI visant à certifier les TRNG pour des applications ultra-sécurisées ;
- suggestion d'une méthode embarquée de mesure du jitter assurant que la quantité mesurée provient de la source d'aléa voulue ;
- une estimation précise de la proportion du jitter due au bruit thermique ;
- étude de la PLL comme source d'aléa afin de :
  - le temps de réponse de la PLL,
  - identifier l'influence des différents paramètres de la PLL sur la qualité de l'aléa,
  - déterminer le niveau de dépendance entre les réalisations du jitter,
  - choisir les paramètres de la PLL qui réduisent ou éliminent l'influence du jitter déterministe,

– évaluer les hypothèses faites pour améliorer le modèle stochastique du générateur à base de PLL.

Chapter 1

Random numbers in cryptography: state-of-the-art

Contents

In this chapter, we introduce the notion of random number generator in general and present a state-of-the-art of the generation of random numbers. Because random numbers are very important in cryptography, they are subject to very strict requirements. These requirements are discussed therein and justify the fact that one cannot just pick any random number generator and use it for cryptographic applications. An emphasis will be put on generators of true random

numbers, especially those that are hardware based, which are the type of interest for this PhD thesis.

1.1 Random number generators

For various purposes, computers need to access random numbers. They are used in weather prediction [20], gaming [21], cryptography [22, Section 4.6] and so much more [23]. Depending on the intended application, different levels of requirements have to be met, leading to different methods for generating random numbers. For example, weather forecasting does not need actual random numbers, they just need to look random, whereas random numbers used in cryptography are subject to very strict requirements [24, 17].

Knowing that computers are deterministic, it appears obvious that they cannot produce sequences of random numbers. Indeed, any sequence of numbers generated by a computer should by all mean contain a pattern which is a mathematical formula describing the process of generation [25, Section 4]. Actually, generating a sequence of random numbers is not an easy task, and there is no unique way of doing so. Any of the various methods used to generate sequences of random numbers is called a random number generator (RNG) [8, Definition 5.1].

In order to obtain a sequence of random numbers, a widespread method consists in using an algorithm which produces sequences of numbers for which the statistical properties approach that of sequences of actual random numbers. By construction, sequences of numbers generated that way are not random. They only mimic sequences of random numbers and are often called pseudorandom numbers [26].

1.1.1 Pseudorandom number generators

A pseudorandom number generator (PRNG) is an algorithm that generates sequences of pseudorandom numbers [27]. The input of a PRNG, called the seed, determines the initial value of the sequence of numbers. In most cases, it is a recursive algorithm for which the output sequence (x_n) is defined as [28, Section 2.2.1]:

$$\begin{cases} x_0 = seed, \\ \forall n \in \mathbb{N}, \quad x_{n+1} = f(x_n), \end{cases} \tag{1.1}$$

where f is a given function. Different nomenclatures exist in the literature to designate PRNGs. Indeed, because the process is deterministic, some authors name them deterministic random number generators (DRNGs) [29]. Moreover, it is frequent that the generated sequence (x_n)

consists of binary numbers, which motivates the nomenclature of pseudorandom bit generators (PRBGs) [30] or deterministic random bit generators (DRBGs) [31].

As the name indicates, the output of a PRNG is not truly random. Indeed, Equation (1.1) shows that the complete sequence is determined by the initial seed and the function f which is publicly known. This implies that the generated sequences represent a fraction of all possible sequences. As a consequence, any sequence generated using Equation (1.1) will definitely repeat numbers if the algorithm keeps running long enough [32, Section 3.2.1]. Indeed, when generating a sequence of numbers of a given length, say n bits, it is impossible to produce $2^n + 1$ numbers without having one of them repeating itself. Because the process is deterministic, as soon as a number is repeated, the sequence has to cycle because of Equation (1.1). The period of such sequence is the smallest positive integer ρ such that $x_{\rho+n} = x_n$, for any integer $n \geqslant n_0$, where n_0 is some positive integer [27, Section 1.4].

Although PRNGs do not produce true random sequences of numbers, the statistical properties of their output make them suitable for many applications [26]. Moreover, these methods have very high throughput, making them more efficient than other methods for generating random numbers [33, Section 8.1]. Various algorithms have been proposed to generate random numbers, the first one, as described in Algorithm 1, is due to von Neumann who wanted to simulate processes involved in nuclear fusion [34].

Algorithm 1 Middle-square method

Require: The initial value $seed$ of the sequence and the length len of the generated sequence.
Ensure: Output a sequence of random looking numbers of length len.

```
n ← NUMBEROFDIGITS(seed)
seq ← MAKEEMPTYLIST()
ADDTOLIST(seq, seed)
for i = 1 to len − 1 do
    next_value ← seed^2
    next_value ← PADWITHLEADINGZEROS(next_value, 2 × n)
    seed ← EXTRACTMIDDLEDIGITS(next_value, n)
    ADDTOLIST(seq, seed)
end for
return seq
```

Even though some values of the seed yield sequences of hundreds of distinct numbers [35], this method displayed many drawbacks which made it not suited for generation of true random numbers. Indeed, most of the generated sequences were either too short (about 142 distinct numbers) or were degenerating to zero [36, 37, 38, 39].

Using multiplicative congruential methods, Lehmer proposed another method for generating sequences of pseudorandom numbers [40]. It was then generalized and became the linear congruential generator (LCG) [41]. The sequence (x_n) of generated numbers is defined as:

$$\begin{cases} x_0 = seed, \\ \forall n \in \mathbb{N}, \quad x_{n+1} = ax_n + b \mod m, \end{cases} \tag{1.2}$$

where a is called the multiplier, b the increment and m the modulus. Values of a, b and m are integers and constant for a given implementation. This method has the advantage of producing sequences of numbers which look random for appropriate values of parameters. A discussion on how to choose these parameters is given by Knuth [32, Section 3.2.1].

It is however important to recall that in his initial version, Lehmer took $b = 0$. Following this idea, Tausworthe proposed another generator which produces a sequence of numbers (x_n) defined as follows [42]:

$$\forall n \in \mathbb{N}, \quad n > k \Longrightarrow x_n = \sum_{i=1}^{k} a_i x_{n-i} \mod m, \tag{1.3}$$

where $(a_i)_{1 \leqslant i \leqslant k}$ is a given vector of integers, the seed in this case being the vector $(x_i)_{0 \leqslant i \leqslant k-1}$. In his proposal, Tausworthe used $m = 2$, but was later generalized by Knuth to any prime modulus for better results [32, Section 3.2.2]. This type of generator is best known as a multiple recurrence generator (MRG) [43], the special case $m = 2$ being designated as the linear feedback shift register (LFSR) or the Tausworthe PRNG [44, Chaper 2]. From the Tausworthe generator, Lewis and Payne derived the generalized feedback shift register (GFSR) generator, based on:

$$\forall k \in \mathbb{N}, \quad a_k = a_{k-p+q} \oplus a_{k-p}, \tag{1.4}$$

where each $(a_k)_{k \in \mathbb{N}}$ is the vector of generated bits, given $(a_k)_{0 \leqslant k \leqslant p-1}$ and the primitive trinomial $x^p + x^q + 1$ in GF(2) [45]. Output of the GFSR is usually presented as a sequence of words $(w_k)_{k \in \mathbb{N}}$ rather than a sequence of bits $(a_k)_{k \in \mathbb{N}}$. Each word w_k is obtained as:

$$\forall k \in \mathbb{N}, \quad w_k = (a_i)_{kn \leqslant i \leqslant kn-1}. \tag{1.5}$$

In terms of w_k's, the output sequence of the GFSR is given as:

$$\forall k \in \mathbb{N}, \quad w_k = w_{k-p+q} \oplus w_{k-p}, \tag{1.6}$$

where $\oplus$ is considered element-wise. Matsumoto and Kurita modified Equation (1.4) into:

$$\forall k \in \mathbb{N}, \quad w_k = w_{k-p+q} \oplus w_{k-p}A, \tag{1.7}$$

where A is a $n \times n$ matrix of bits. The generator based on Equation (1.7) is known as the twisted GFSR (TGFSR) [46]. Various other algorithmic methods for generating pseudorandom numbers were proposed [9, Chapter 2]. The most widely used is a variant of the TGFSR, the Mersenne Twister developed by Matsumoto and Nishimura [47]. Its name derives from the fact that its period is a Mersenne prime[1], generally $2^{19937} - 1$. The Mersenne Twister is currently (at the time of writing) being used as the default RNG in several programming languages such as C++ [49], Python [50] and R [51].

Despite its popularity and the statistical qualities of its output, the Mersenne Twister is not secure [47, 52]. Its use for cryptographic applications is therefore prohibited, the same goes for all above-mentioned PRNGs. For cryptographic applications, statistical qualities of the sequence of generated numbers is necessary but not sufficient to guarantee security. Additional properties, more demanding, are required for any generator that should produce sequences of random numbers for cryptography [24, Section 5.3]. Basic properties are:

- forward secrecy which ensures that, given $n-1$ successive bits of the output sequence, no adversary should be able to predict the nth bit with probability greater than $\frac{1}{2}$;
- enhanced backward secrecy which is the guarantee that previously generated numbers cannot be compromised by neither the current or future output values, nor by the knowledge of the state of the generator at a given time.

Any PRNG that meet these requirements is called a cryptographically secure PRNG (CSPRNG) [25]. For practical reasons, requirements set for a CSPRNG are relative to an efficient algorithm, more precisely a polynomial-time one [53].

The most famous example of CSPRNGs is the Blum Blum Shub (BBS) algorithm based on the recurrence [54]:

$$\forall k \in \mathbb{N}, \quad x_{k+1} = x_k^2 \mod n, \tag{1.8}$$

where n is a Blum integer[2]. Even though historically important, its use for nowadays security purposes is deprecated [9, Section 6.2]. Other CSPRNGs, like Fortuna, are preferred to BBS for providing better security level [56, Chapter 10]. This algorithm consists of three parts:

[1] A prime number p is a Mersenne prime if there exists $n \in \mathbb{N}$ such that $p = 2^n - 1$ [48].

[2] A positive integer n is said to be a Blum integer if there exist two distinct primes p and q such that [55]:

$$p \equiv 3 \mod 4, \quad q \equiv 3 \mod 4 \quad \text{and} \quad n = pq.$$

- an entropy collector which retrieves real random data from various sources, and uses this data to reseed the generator;
- a seed management system which keeps a file containing a secure seed at the disposal of the machine in case of a system reboot or a new switch on;
- the generator itself which is a block cipher in counter mode.

PRNGs attract attention because their are computationally efficient and do not require any special device. Moreover, the generation process is deterministic. Therefore, given the same seed, the algorithm will always produce the same sequence of numbers again. This property of PRNGs is appreciated in areas where the reproducibility of simulations is desired. However it is not the case in cryptography where we want the generated sequences to be unpredictable and non reproducible. Also, PRNGs need to be seeded by a genuine random seed, raising the problem of obtaining that seed. More to the point, PRNGS produce sequences of numbers which contain some pattern. This can lead to security breaches that can weaken the overall cryptographic construction. Methods that produce true random numbers are therefore required.

1.1.2 True random number generators

In the special case of cryptography, the use of *true* random numbers is required. The procedure through which these actual random numbers are generated is called a true random number generator (TRNG). Unlike PRNGs, generators of this type are not algorithmic. They are actually apparatus which exploit a well defined physical process to extract randomness for the generation of numbers. The exploited physical phenomenon must be random, meaning, it should not be possible to describe nor predict its evolution in a deterministic way, no matter the level of knowledge one has on that phenomenon.

TRNG's main advantage is to provide an output which is random indeed, since the exploited physical process is random. Hence, they offer the insurance that no one should be able to predict the generated numbers, even with a perfect knowledge of the architecture and the functionalities of the generator. Their design also requires that no one should be able to synchronize two identical generators to produce the sequence of numbers [57]. This requirement is actually one huge difference between TRNGs and PRNGs. Indeed, it implies that no TRNG must accept an initial state, which is not the case for PRNGs which cannot operate without specifying an initial state (namely, the seed).

The general principle of a TRNG, as depicted in Figure 1.1, consists of a source of randomness which generates a random analog signal. This analog signal then feeds a digitizer which produces samples of the analog signal, called digitized analog signal [24], or digitized data [17]. In general, the raw digital signal consists of binary digits which are then (eventually) sent to a post-processor which enhances the statistical and cryptographic qualities of the generated bit stream. For this reason, we will consider in the remainder of this thesis, that a TRNG actually produces a sequence of random bits.

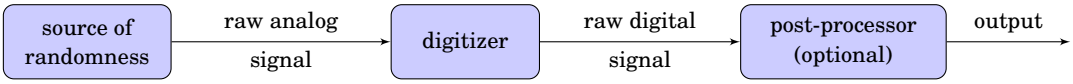


Figure 1.1: General structure of a TRNG.

1.1.2.1 Source of randomness

The source of randomness is actually the most critical part of any TRNG, since the general behavior of the generator depends on it. It must therefore be a non manipulable random process. Depending on the process used, one can distinguish physical TRNGs (P-TRNGs) and non physical TRNGs (NP-TRNGs). The idea with NP-TRNGs is that if a huge amount of data from different sources are collected and mapped onto a shorter sequence (a hash function, for instance), the output value will appear random to an observer who neither knows the source data nor is able to control them. The Linux RNG /dev/random which uses processes like disk operations and interrupts is an example of such generators [58]. Since we are interested in the origin of random behavior, and therefore in the characterization of the source of randomness, we will focus on P-TRNGs.

With P-TRNGs, randomness comes from a physical microscopic random process such as radioactive decay [59], metastability [60], biological data [61] or electronic noises like noise from Zener diodes [13] or Johnson noise [62, 63, 64]. The use of these sources of randomness yields various designs of P-TRNG. In this thesis, we will exclusively deal with generators implemented in logic devices like Application Specific Integrated Circuits (ASICs) and Field Programmable Gate Arrays (FPGAs). For these generators, the source of randomness consists of electronic noises which often manifest themselves as frequency instabilities of electronic oscillators such as Ring Oscillators (ROs) [65], Transient Effect Ring Oscillators (TEROs) [66], Self-Timed Ring Oscillators (STRs) [67] or Voltage-Controlled Oscillators (VCOs) [68]. They are often referred to as oscillator-based TRNGs [69], and their general structure is depicted in Figure 1.2.

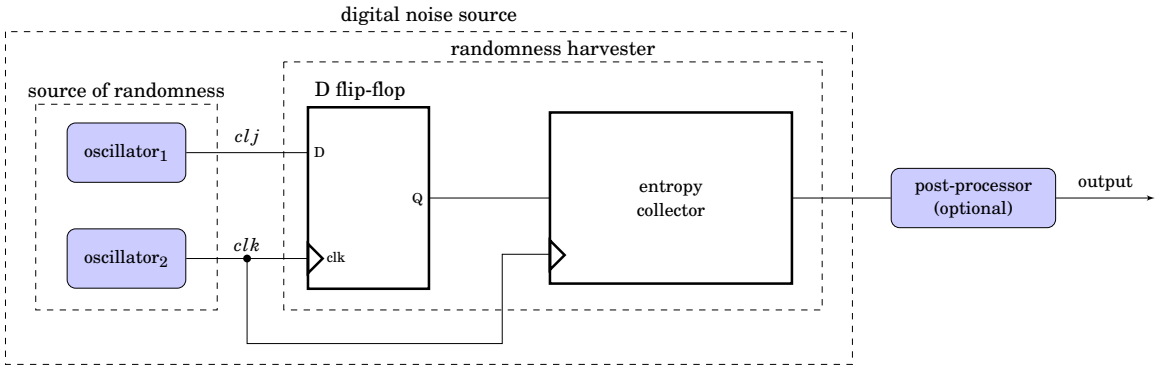


Figure 1.2: Oscillator-based TRNG.

1.1.2.2 Randomness harvester

Because the source of randomness is a physical process, it produces a random analog signal. This signal cannot be used directly to produce random bits in logic devices. One therefore needs to convert it in a digital signal, procedure which is called digitization [24]. In logic devices, the digitization is closely related to the randomness extraction procedure. The most widespread randomness extraction method consists in sampling the random phenomenon at discrete times. This is done using a D flip-flop which produces a 1-bit sample of the random signal at each clock cycle [68, Section 3].

In order to assess the randomness, it is necessary to have a way of quantifying it. A common measure of randomness is the "entropy" which will be dealt with in more details in Section1.3. To ensure enough entropy at the output, it is common to allow the random process to evolve over a long period of time. This results in an entropy accumulation which yields optimal security [70]. In hardware, this is usually done with an entropy collector such as a decimator which produces one random bit out of a given number of samples [71]. In the remainder of this thesis, we will consider the association of the D flip-flop and the entropy collector as the digitizer. This choice is justified by the fact that it is part of the TRNG design and that the entropy collector is not optional like the post-processor. Even though this process uses an analog signal to produce a digital one, the term harvesting mechanism seems more appropriate than digitization since it describes the process through which randomness is extracted from the real world phenomenon.

1.1.2.3 Post-processor

The main philosophy of a TRNG is to produce sequences that cannot be recognized from those generated by an ideal RNG, *i.e.* they must be unbiased and uniformly distributed. However, due to imperfections that can occur both in the physical process and the harvester, the generated se-

quence may be biased. Hence it may contain a specific bit value which appears more often than the other one [72]. This is actually a serious security issue since an adversary who knows that bias can take advantage of it and guess the next values of a sequence with probability higher than 0.5 [24, Section 5.1]. To avoid such situations, the raw digital signal must go through a post-processing step before the output [10].

The goal of this post-processing step is to eliminate (or at least, reduce as much as possible) imperfections in the raw digital signal. To achieve this goal, the post-processor usually compresses the input bit stream. The output has therefore a smaller length with better statistical qualities [73]. It is important to understand that the post-processor does not add any entropy to the bit stream, it just makes it look more random. This only increases the robustness of the generator while reducing its output bit rate. There exist various techniques for designing a post-processor, ranging from ad hoc methods to more elaborate ones such as cryptographic hash functions or resilient functions [74, 75, 73].

Although there is no proper definition of which part of a TRNG should be considered as post-processor, one commonly considers the post-processor as a complex process designed to reduce imperfections present in the raw digital signal. Post-processing algorithms usually take a lot of resources and are therefore not suitable for direct implementation in hardware. Note that the use of a post-processor is not mandatory, indeed when the raw digital signal already exhibits good statistical qualities, the use of a post-processor is not relevant.

From all that precedes, one understands that the main part of interest in a TRNG is its source of randomness. Any randomness that will be used to generate random bits comes from there. It is thus of vital importance to focus on this part and understand various phenomena occurring in it.

1.2 Sources of randomness in logic devices

In Section 1.1.2, we explained that the source of randomness is the most crucial part of any TRNG. Cautions must then be taken in selecting that source. In general, the design of a good TRNG is very challenging. The identification and correct exploitation of the source of randomness is by far the most challenging task in the design of a TRNG. Those implemented in logic devices are no exception to that, even though their implementation is simpler than other types of TRNGs [76, 69]. Since the security of these constructions relies on the secrecy of the key, it is important that keys are generated within the system to avoid any transport across an uncontrolled area. This explains why we focus on TRNGs that can be implemented in logic devices. Indeed, such TRNGs

constitute a source of random numbers directly available to the cryptographic constructions for various use [10]. In this section we discuss various sources of randomness present in logic devices.

1.2.1 Commonly used sources of randomness

Diverse electronic phenomena occur in logic devices. Some of them exhibit random behavior which might be used to generate random numbers. Among the most encountered electronic random phenomena, the ones listed below are commonly used.

- Metastability which is the ability of a digital electronic system to persist in an unstable equilibrium for an indefinite time [77]. This phenomenon is rare and therefore very difficult to sample. It is consequently difficult to be sure that the output bit really depends on metastability.
- Oscillatory metastability which is the ability of a bi-stable electronic device (for example, an RS flip-flop) to oscillate between its two unstable equilibrium states, for an indefinite period [78]. It can be considered as a special case of metastability. They therefore have the same limitations. In addition, this is a phenomenon typical of a small class of bistable circuits, which raises the problem of its use on electronic circuits in general.
- Start-up value of flip-flops (or a memory element) to a random state either after power-up or periodically [79]. This is due to the fact that the behavior of the device is not always defined before the occurrence of a valid clock signal. However, there are different ways to prevent it. One of them being to initialize the flip-flop, which can be done by an adversary, and thus compromise the generation of random numbers.
- Clock jitter which is a short-term variation of an event from its ideal position [69]. This phenomenon is usually unwanted because it negatively affects most communication and high-speed systems [80]. It is however inevitable and uncontrollable, making it a prime candidate to be used as source of randomness [81, Section 1.1.1]. This is the reason why the clock jitter is a widespread source of randomness used for TRNGs implemented in logic devices [82, 71, 64, 83, 65, 84, 69, 85]. We will therefore focus on the clock jitter in the remainder of this thesis.

1.2.2 Clock jitter as a source of randomness

In logic devices, actions of digital circuits are coordinated by a special signal called the clock signal [86]. In an ideal model, this signal is a square wave oscillating between two states (high and low) with a duty cycle of 50% and a stable period. However, various electronic noises affect logic devices, causing the clock period (period of the clock signal) to fluctuate around its ideal value.

This fluctuation of the period, called clock jitter, causes edges of real-world clock signals to occur slightly earlier or later than expected as shown in Figure 1.3 [87, Chapter 8].

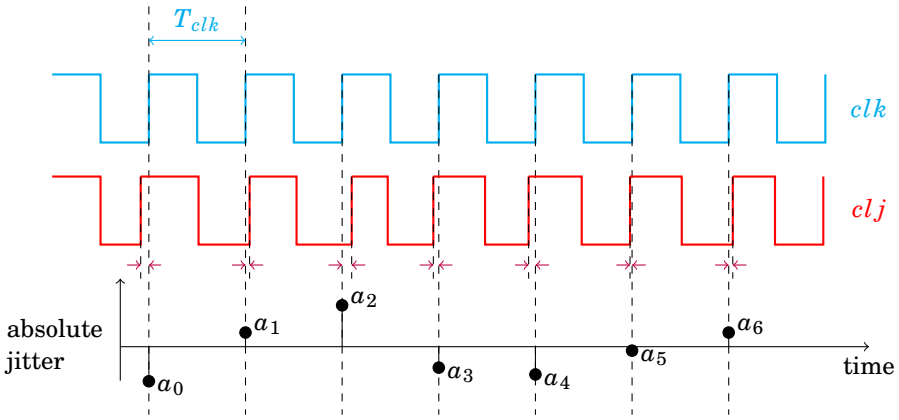


Figure 1.3: Illustration of clock jitter (clk: reference clock signal, clj: jittered clock signal, a_k: values of the jitter).

Based on the jitter measurement method used, various definitions of the clock jitter can be met in the literature. Most of these definitions are not standardized and therefore used with different meanings, depending on the application and the author's background. This variety of terms used to express the concept of jitter leads to a lot of misunderstanding and confusion in the study of clock jitter. Da Dalt and Sheikholeslami showed that each of these different definitions actually falls into one of the four fundamental definitions of jitter [81, Chapter 2].

Generally speaking, jitter is the deviation of the instant at which a given event occurs, relative to a reference time frame. In the context of this thesis, as it is generally the case in the field of hardware based TRNG, the event we consider is the occurrence of (rising and falling) edges of the sampled clock signal. Note that the choice of the reference time frame is arbitrary, and is usually made in two ways: either the edges of the clock under investigation are compared to the edges of another clock, or they are compared to some previous edges of the same clock. The first approach leads to the definition of absolute and relative jitter, while the second leads to the definition of period jitter. The period jitter can be extended into a fourth jitter definition, namely the N-period jitter. These four jitter definitions are actually related to each other and will be discussed next [14].

1.2.2.1 Absolute jitter

Let us consider an ideal clock signal clk, with a clock period T. If we assume that the first rising edge occurs at time $t = 0$, then the subsequent rising edges occur at exactly $t = kT$, where $k \in \mathbb{N}$.

In the case of a non-ideal clock clj, with nominal period[3] T, the rising edges occur at times t_k which deviate from their ideal values kT. The absolute jitter of clj (at the k-th period) is thus defined as the time displacement of the k-th rising edge of clj with respect to the corresponding edge of the ideal clock clk (see Figure 1.4). Mathematically speaking, it is expressed as:

$$a_k := t_k - k \cdot T. \tag{1.9}$$

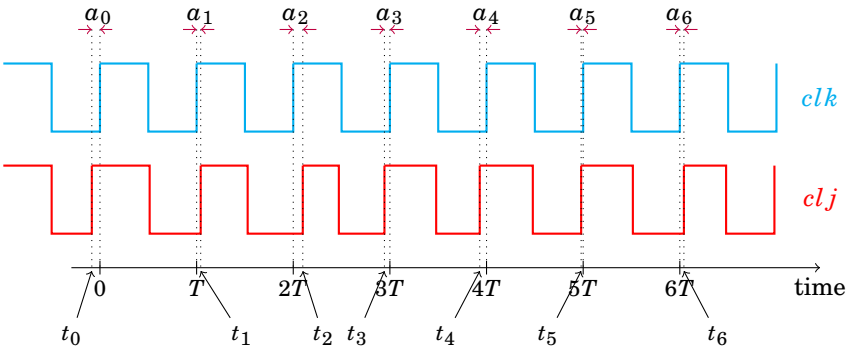


Figure 1.4: Absolute jitter as a time deviation.

In Figure 1.3, the absolute jitter is a discrete-time random signal. Mathematics of random signals will be used in Chapter 2 to study and characterize the jitter.Because the signal clj and its ideal version clk both have the same period, (a_k) is a zero-mean process. In some cases, when clk and clj are phase shifted, the positions of clk and clj can display an offset t_{os} such that (a_k) has mean value t_{os}. In that case, the absolute jitter is defined as:

$$a_k := t_k - k \cdot T - t_{os}, \tag{1.10}$$

in a way that (a_k) is still a zero-mean process. For this reason, we will consider the absolute jitter to be a zero-mean random process.

Note that various definition exist for the absolute jitter. Indeed, it corresponds to the time error (TE) defined by the telecommunication standardization sector of the International Telecommunication Union [88, Section 4.5.13].

1.2.2.2 Relative jitter

There-above, the edges of a clock signal were compared to those of its ideal version. However, no ideal clock signal exists in the real world. Thus, instead of proceeding as shown above, one can think of comparing edges of a clock signal s_1 to those of another (non-ideal) one s_0 having the same nominal period T. The situation is actually the same as depicted in Figures 1.3 and 1.4.

[3] This means that each period of clj is different, but its mean period is T.

The only difference is that clj is replaced by s_1 and clk by s_0, both s_0 and s_1 being non-ideal clock signals.

This setting leads to the definition of the relative jitter as a discrete-time random process r_k, where the element r_k is the time displacement of the k-th rising edge $t_k^{s_1}$ of s_1 with respect to the corresponding edge $t_k^{s_0}$ of s_0. Mathematically, it is thus expressed as:

$$r_k := t_k^{s_1} - t_k^{s_0}. \tag{1.11}$$

Since both s_0 and s_1 have respective absolute jitter processes $\left(a_k^{s_0}\right)$ and $\left(a_k^{s_1}\right)$, it is possible to express the relative jitter in term of absolute jitters of $s0$ and s_1. By the use of Equation (1.9), we can rewrite Equation (1.11) as:

$$r_k = a_k^{s_1} - a_k^{s_0}. \tag{1.12}$$

Note that for the same considerations as for the absolute jitter, the relative jitter is a zero-mean random process.

1.2.2.3 Period jitter

The two jitter definitions provided above are based on comparing the edges of the clock signal to the edges of another clock signal. However, it is also possible to compare the position of an edge to the position of the previous edge of the same clock signal. Assume that, for a clock signal, a specific rising edge occurs at time t_k and the next one at time t_{k+1}, then $t_{k+1} - t_k$ represents one realization of the clock period. Comparing this realization of the clock period to its nominal value leads to the definition of the period jitter.

The period jitter is defined as a discrete-time random process (p_k), for which each p_k is the time deviation of the k-th clock period from its nominal value. Figure 1.5 illustrates this concept and shows that the k-th clock period is actually the time difference of the $(k+1)$-th and k-th rising edges. The k-th sample of the period jitter can then be mathematically expressed as:

$$p_k := (t_{k+1} - t_k) - T, \tag{1.13}$$

where T is the nominal period of the clock signal. If we call $T_k := t_{k+1} - t_k$, the current clock period, Equation (1.13) becomes:

$$p_k = T_k - T. \tag{1.14}$$

Note that period jitter can also be expressed in term of absolute jitter. Indeed, using Equation (1.9), Equation (1.13) becomes:

$$p_k = a_{k+1} - a_k. \tag{1.15}$$

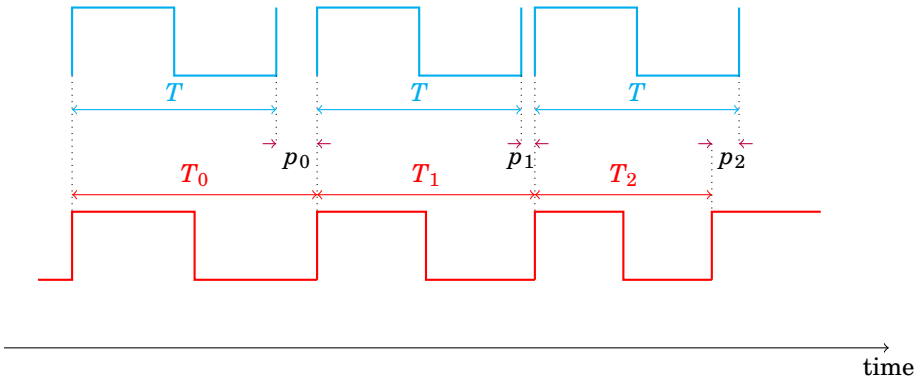


Figure 1.5: Illustration of the period jitter.

The period jitter is sometimes (wrongly) named cycle-to-cycle jitter [89, 90]. Indeed, from the JEDEC standard JESD65B [91, Page 10], cycle-to-cycle jitter is the variation in cycle time of a signal between adjacent cycles, over a random sample of adjacent cycle pair. It is thus the difference between two consecutive clock periods and it indicates how much one period of the clock differs from the previous one. If we denote by (cc_k) the random process of cycle-to-cycle jitter, one has:

$$cc_k := T_{k+1} - T_k, \tag{1.16}$$

which can be expressed in terms of period jitter as:

$$cc_k := p_{k+1} - p_k. \tag{1.17}$$

This latter equation shows that these two notions are not the same, even though they are related.

1.2.2.4 N-Period jitter

Because the designer of a circuit aims at reducing the clock jitter, it appears that clock jitter is very small compared to the clock period (approximately 1‰ of the clock period). In order to guarantee enough entropy, it is common practice to compare the time deviation of a clock edge not to the immediate preceding one, but to the N-th previous one, as illustrated in Figure 1.6.

This procedure leads to the definition of N-period jitter as the discrete-time random process $(p_k(N))$, where each element $p_k(N)$ is the deviation around the nominal value of the position of one clock edge with respect to the N-th previous edge. Mathematically, $p_k(N)$ is expressed as the deviation of the time difference between the k-th and the $(k+N)$-th edges from the nominal value NT, that is:

$$p_k(N) := (t_{k+N} - t_k) - NT. \tag{1.18}$$

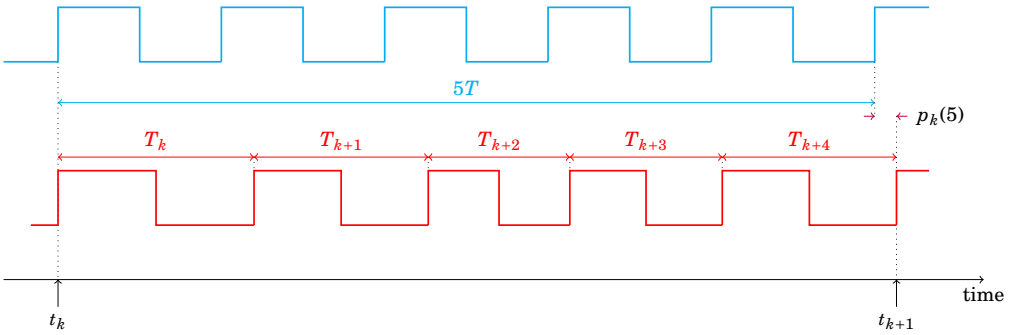


Figure 1.6: Illustration of the N-period jitter.

From this definition, one can observe that Equation (1.13) is a special case of Equation (1.18) for $N = 1$. It thus appears that N-period jitter is a generalization of the period jitter discussed above. Moreover, considering that $t_{k+N} - t_k$ is the duration of the N periods of the clock following the k-edge, the N-period jitter can be written as:

$$p_k(N) = \left(\sum_{i=k}^{k+N-1} T_i \right) - NT, \tag{1.19}$$

where T_i indicates the i-th clock period. From Equation (1.19), one can derive:

$$p_k(N) = \sum_{i=k}^{k+N-1} (T_i - T) = \sum_{i=k}^{k+N-1} p_i, \tag{1.20}$$

where each p_i represents the period jitter of the i-th clock period. It then follows that the N-period jitter is the sum of the period jitter over N consecutive periods. Because it originates from the accumulation of the jitter over consecutive periods, various authors refer the N-period jitter as the accumulated jitter [92, 84, 93, 69, 66, 60, 94]. As in the case of period jitter, it is possible to express the N-period jitter in terms of the absolute jitter:

$$p_k(N) = a_{k+N} - a_k. \tag{1.21}$$

Equation (1.21) shows that the N-period jitter is what the International Telecommunication Union refers to as the time interval error [88, Section 4.5.14].

The various above-mentioned types of jitter are all mutually related. This shows that from a physical point of view, these various definitions refer to the same phenomenon affecting the clock signal. Their specificities come from the method used to measure this phenomenon.

In real-world, signals are all jittered, so it is the relative jitter we often have access to. However, when two signals are independent from each other, it is possible to transfer the jitter of one of these signals to the other. This results in an ideal signal, and a jittered one containing the

contribution of the jitters of both signals, allowing the absolute jitter to be evaluated [95]. Since this approach simplifies computations, we will focus on the absolute jitter, which will be simply denoted jitter in the remainder of this thesis.

1.2.3 Jitter sources

In section 1.2.2, we briefly explained what the clock jitter is and its different representations. In order to have a better understanding of this phenomenon, it is common to break it down and identify its different parts. Since each component of the jitter has its own characteristic and physical meaning, the separation of the jitter can be used to investigate its cause. The separation of the jitter is usually made based the nature of the jitter component as depicted in Figure 1.7 [81, Section 2.2.6].

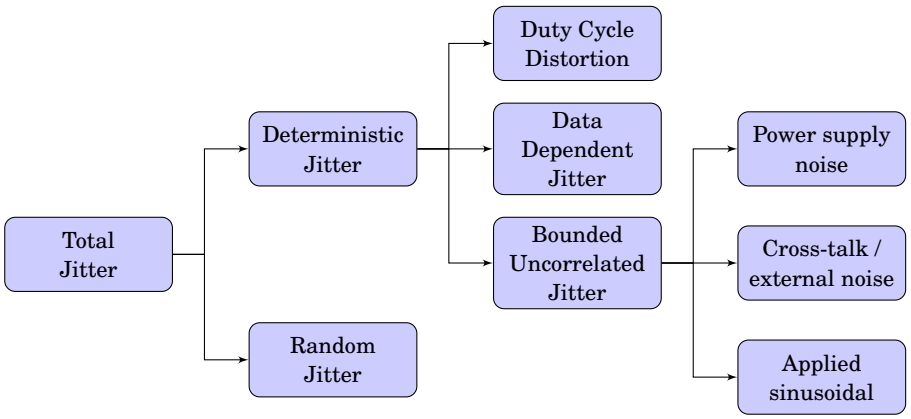


Figure 1.7: Overview of the jitter components.

In Figure 1.7, we see that jitter has two main components: a random component and a deterministic component. There are several methods to determine if the jitter is of random or deterministic origin [96, Section 8]. The simplest one is the observation of the histogram. A histogram in the shape of a Gaussian is often indicative of a random jitter.

Random jitter includes all components for which the probability density is not bounded [96, Section 7.2.2]. This means that the range of the random jitter is not limited. In electronic circuits, it is produced by electronic noises such as thermal noise, flicker noise and shot noise [96, Section 9.2.2.1]. It is generally described as a random phenomenon with a zero mean Gaussian distribution, and therefore characterized by its variance or standard deviation.

The deterministic jitter consists of the components of the jitter for which the probability density is bounded [96, Section 7.2.3]. The most common causes of deterministic jitter are instabilities

from the power supply, cross-talk from other signals or channels, distortion of the duty cycle, limitation of the channel bandwidth. Depending on the origin mechanism, it is possible to divide the deterministic jitter into several sub-categories, the most common being listed below.

- Duty cycle distortion: due to asymmetries in the duty cycle when both rising and falling edges of a clock signal are used in a given application.
- Data dependent jitter: specific to the data pattern transmitted in the same path as the signal under test.
- Bounded-uncorrelated jitter: present in the serial digital data, but bears no correlation with the transmitted data. It has three main sources: (1) power supply noise that affects the launched signal, (2) cross-talk that occurs during transmission and (3) sinusoidal applied to the receiver input for jitter tolerance measurements.

To ensure that generated numbers are random, it is necessary that the only jitter components used are random. In other words, it is imperative to mitigate the influences of deterministic components of the jitter. However, some random components present security risks, and therefore need to be eliminated as well. In order to respond more effectively to security issues related to the generation of random numbers, it is possible to consider another subdivision of jitter, taking into account not only its nature, but also its source (local or global) as shown in Figure 1.8.

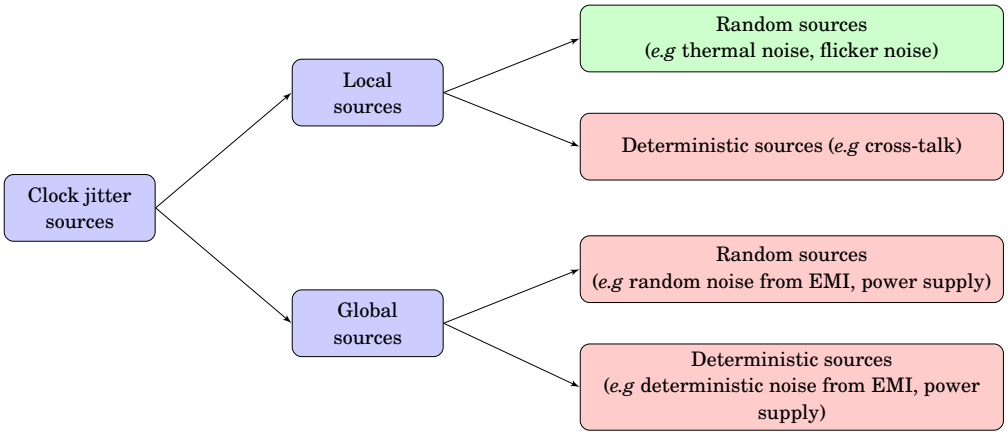


Figure 1.8: Overview of the jitter sources.

Global sources (random and deterministic) are external sources, and therefore potentially manipulable. An adversary can therefore a priori exploit these sources in order to deteriorate the quality of the randomness, and thus reduce the security of cryptographic systems. Thus, in addition to reducing the effects of deterministic sources, it is important to reduce the effects of global sources on the generation of random numbers. For this purpose, it is possible to use two identical

oscillators in Figure 1.2. Indeed, because they are subjected to the same phenomena, they will display the same effects due to these global sources. Thus, as a differential principle, these effects will cancel each other during sampling [97].

Random local sources are internal and thus non-manipulable. They are therefore the only recommended ones for generating random numbers for cryptography. Consequently, it is mandatory that the jitter measurement methods evaluate only random local components. This problem is not simple and will be discussed in more detail in Chapter 2.

1.3 Entropy

The term *entropy* was first introduced in 1865 by Clausius in thermodynamics [98]. It derives from the combination of an ancient English word and a Greek word, which together mean *inside transformation*. The entropy characterizes the level of disorganization, diversity, uncertainty or randomness of a given system [99]. It is therefore a suitable tool to evaluate the quality of a sequence of random numbers. In the field of information theory, the notion of entropy was brought by Shannon [100], then generalized by Rényi [101]. We will give a brief overview of various entropies used in information theory and which will be required for the rest of this thesis.

1.3.1 Rényi entropy

1.3.1.1 Understanding entropy

Let X be a valued random variable with image $\mathscr{X}$, in order to know how random X is, one needs to compute its entropy or level of randomness. The higher this entropy is, the more random X is. If we call $\mathbf{P}_X$, the probability distribution[4] of X, and $\alpha \in \mathbb{R}_+ \setminus \{1\}$, Rényi defined the entropy of X of order α as:

$$H_\alpha(X) := \frac{1}{1-\alpha} \log\left(\sum_{x \in \mathscr{X}} p(x)^\alpha\right). \tag{1.23}$$

It appears from Equation (1.23) that $H_\alpha(X)$ depends on the probability distribution of X. Since this probability is related to the random variable X, the entropy is commonly considered as a function of X. Note that the definition of entropy is given here using base 2 logarithm. However, it is possible to define it using any base logarithm [103]. In this thesis, we will however restrict to the base 2 logarithm. For this reason, the entropy will be expressed in bits.

[4] The probability distribution, $\mathbf{P}_X$, of the random variable X is defined for any $x \in \mathscr{X}$ as [102, Section 2.1]:

$$\mathbf{P}_X(x) = \mathbf{P}(X = x) = \mathbf{P}(\{\omega \in \Omega : X(\omega) = x\}). \tag{1.22}$$

For the sake of simplicity, we will write $p(x)$ to mean $\mathbf{P}(X = x)$.

The entropy $H_\alpha(X)$ provides a measure of the quantity of unpredictable information contained in the random variable X. This can be understood as the amount of information we obtain by observing the outcome of the experiment involving X [104]. It turns out that the more difficult it is to correctly guess the outcome of X, the more information we get by observing is actual outcome. For example, if X is deterministic, then $\mathscr{X}$ contains only one value, that is $\mathscr{X} = \{x\}$ and thus $p(x) = 1$. One does not need to observe the outcome of X before to know that it will be x. He therefore gains no information by observing the outcome of X. One may note that in this case $H_\alpha(X) = 0$, which is actually a result characterizing any deterministic phenomenon.

From what precedes, one understands that the underlying phenomenon responsible of the TRNG operation must have as high entropy as possible to ensure a high difficulty in guessing the output of the considered TRNG.

1.3.1.2 General properties of Rényi entropy

Equation (1.23) shows that Rényi entropy is actually a parametric family of entropy measures for $\alpha \in \mathbb{R}_+ \setminus \{1\}$. The goal of Rényi when defining his entropy was to have the most general class of information measures that preserve the additivity of statistically independent systems and are compatible with probability axioms [101]. The limit classes, when α approaches 1 and when α approaches infinity, lead to two entropy measures interesting for the field of random number generation.

Continuous extension of Rényi entropy at $\alpha = 1$ From Equation (1.23), Rényi entropy is not defined for $\alpha = 1$. However, for a given X, the function $\alpha \longmapsto H_\alpha(X)$ has a limit as α approaches 1. The proof reveals that:

$$\lim_{\alpha \to 1} H_\alpha(X) = -\sum_{x \in \mathscr{X}} p(x) \log p(x). \tag{1.24}$$

This limit is the Shannon entropy which will be detailed in Section 1.3.2. From the continuous extension theorem, one can then consider Rényi entropy defined for $\alpha \in \mathbb{R}_+$.

Min entropy For a given random variable X, if $\mathscr{X}$ is such that $\max_{x \in \mathscr{X}} p(x)$ exists, then one can prove:

$$\lim_{\alpha \to \infty} H_\alpha(X) = -\log\left(\max_{x \in \mathscr{X}} p(x)\right). \tag{1.25}$$

This limit is called min entropy and denoted $H_\infty(X)$. Knowing that log is a continuous and non-decreasing function, one can write:

$$\log\left(\max_{x \in \mathscr{X}} p(x)\right) = \max_{x \in \mathscr{X}} \log p(x), \tag{1.26}$$

which yields:

$$H_\infty(X) = -\log\left(\max_{x\in\mathscr{X}} p(x)\right) = -\max_{x\in\mathscr{X}} \log p(x) = \min_{x\in\mathscr{X}} [-\log p(x)]. \tag{1.27}$$

Monotony of $\alpha \longmapsto H_\alpha$ For a given random variable X, it can be proved that $\alpha \longmapsto H_\alpha(X)$ is a non-increasing function. Thus, for any $\alpha_1, \alpha_2 \in \mathbb{R}_+$:

$$\alpha_1 < \alpha_2 \quad \Longrightarrow \quad H_{\alpha_1}(X) > H_{\alpha_2}(X). \tag{1.28}$$

As an example, if we consider X being the outcome of a coin flip, Figure 1.9 shows that the curve of Shannon entropy ($\alpha = 1$) is always above that of the min entropy ($\alpha \to \infty$). Actually, min entropy is always lower than any other entropy measure, which justifies its name. It is therefore the smallest entropy measure in the family of Rényi entropies. In this sense, it is the most conservative measure that can be used to evaluate the amount of randomness of a phenomenon.

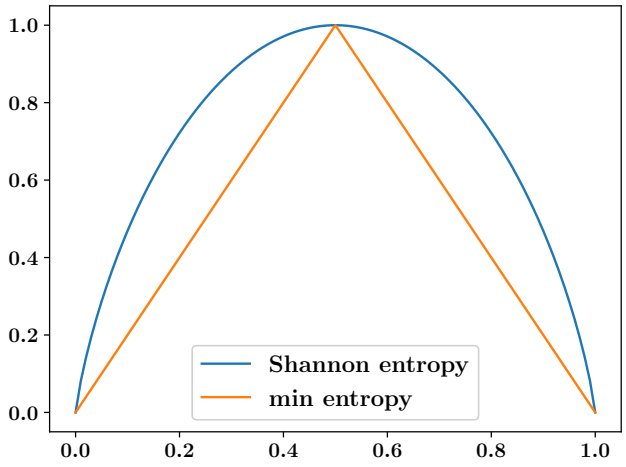


Figure 1.9: Comparison of the Shannon entropy and the min entropy, in the case of a coin flip.

Max entropy For $\alpha = 0$, we have $p_x^\alpha = 1$ for each $x \in \mathscr{X}$. This yields:

$$H_0(X) = \log|\mathscr{X}|, \tag{1.29}$$

where $|\mathscr{X}|$ is the cardinality of $\mathscr{X}$.

Since $\alpha \longmapsto H_\alpha(X)$ is non-increasing, we can say that $H_0(X)$ gives an upper bound to the entropy of X. This motivates the fact that $H_0(X)$ is called max-entropy, which is also called Hartley entropy [105, 104]. It follows that for any $\alpha \in \mathbb{R}_+$:

$$0 \leqslant H_\alpha(X) \leqslant \log|\mathscr{X}|. \tag{1.30}$$

Note that for any $\alpha \in \mathbb{R}_+$, one has $H_\alpha(X) = H_0(X)$ if, and only if, all the outcomes of X have the same probability. This means that the maximum entropy of a phenomenon is reached when its various outcomes are uniformly distributed.

Of these entropy measures, Shannon's entropy is the most commonly used. For example, the most common TRNG assessment procedure in Europe uses it [24]. We will therefore detail its study in the following section.

1.3.2 Shannon entropy

Information theory is concerned with measure of transmitted information content. The basic idea being the more one knows about a system, the less information that system contains. As said earlier, this amount of information is exactly the amount of uncertainty one has of the outcome of a system without observing that outcome. Even though, measures of uncertainty prior to Shannon's existed, introduction of Shannon entropy is widely considered as the birth of information theory.

Let X be a $\mathscr{X}$-valued random variable, Shannon entropy of X is defined as [100]:

$$H(X) := -\sum_{x \in \mathscr{X}} p(x) \log p(x). \tag{1.31}$$

As discussed in Section 1.3.1, one can consider Shannon entropy as a special case of Rényi entropy, as α approaches 1. Shannon was the first to refer to the measure of uncertainty as entropy, due to the analogy of the Boltzmann entropy in thermodynamics [104].

1.3.2.1 Conditional entropy - Mutual information

In a TRNG model, each output bit is considered as a realization of a random variable. Since TRNGs usually output streams of random bits, it is more realistic to consider several random variables than just one. In this setting, one may be interested in evaluating how difficult it would be to guess the next output bit, knowing the current one. This yields the notion of conditional entropy of two random variables, which can be generalized to any number of random variables.

Let X and Y be two random variables defined over the same probability space, with respective images $\mathscr{X}$ and $\mathscr{Y}$. If we consider the outcome of Y prior to the outcome of X, we can assume that this outcome is some value $y \in \mathscr{Y}$. We can then estimate the level of uncertainty on the outcome of X, based on this knowledge as [106, Section 3.6]:

$$H(X|Y=y) := -\sum_{x\in\mathscr{X}} p(x|y)\log p(x|y), \tag{1.32}$$

where $p(x|y)$ is the conditional probability[5] of X being x given $Y=y$. Because y can take any value in $\mathscr{Y}$, one can deduce the average of theses various entropies:

$$H(X|Y) := \sum_{y\in\mathscr{Y}} p(y)H(X|Y=y). \tag{1.34}$$

This average quantity is called conditional entropy of X given Y, and can be rewritten as:

$$H(X|Y) := -\sum_{x\in\mathscr{X}}\sum_{y\in\mathscr{Y}} p(x,y)\log p(x|y), \tag{1.35}$$

where $p(x,y)$ is the joint probability[6] of $X=x$ and $Y=y$.

Equation (1.35) therefore provides the amount of remaining information about X, after observing the outcome of Y. This implies somehow a loss of information regarding X. This loss is the mutual information between X and Y, denoted $I(X,Y)$, and satisfies:

$$I(X,Y) = H(X) - H(X|Y). \tag{1.37}$$

The mutual information is actually the amount of information common to both X and Y, and quantifies the total decrease in uncertainty in X by observing Y. It can equally be expressed as:

$$I(X,Y) := \sum_{x\in\mathscr{X}}\sum_{y\in\mathscr{Y}} p(x,y)\log\frac{p(x,y)}{p(x)p(y)}. \tag{1.38}$$

It characterizes the level of dependence between random variables X and Y. Indeed, it can be shown that this mutual information is null if, and only if, X and Y are independent random variables. This implies that $H(X|Y)=H(X)$ when X and Y are independent random variables.

[5] The conditional probability distribution, $\mathbf{P}_{X|Y}$, of the random variable X given Y with respective images $\mathscr{X}$ and $\mathscr{Y}$, is defined for any $x\in\mathscr{X}$ and $y\in\mathscr{Y}$ as [99, Section 1.4]:

$$\mathbf{P}_{X|Y}(x|y) = \mathbf{P}(X=x|Y=y) = \frac{\mathbf{P}(X=x,Y=y)}{\mathbf{P}(Y=y)}. \tag{1.33}$$

For the sake of simplicity, it is written $p(x|y)$.

[6] The joint probability distribution, $\mathbf{P}_{X,Y}$, of random variables X and Y with respective images $\mathscr{X}$ and $\mathscr{Y}$, is defined for any $x\in\mathscr{X}$ and $y\in\mathscr{Y}$ as [102, Section 4.1]:

$$\mathbf{P}_{X,Y}(x,y) = \mathbf{P}(X=x,Y=y) = \mathbf{P}(\{\omega\in\Omega : X(\omega)=x, Y(\omega)=y\}). \tag{1.36}$$

For the sake of simplicity, it is written $p(x,y)$.

1.3.2.2 Joint entropy

Instead of considering the uncertainty related to only one output bit of the generator, it is interesting to evaluate the uncertainty related to two output bits. This yields the notion of joint entropy of two random variables X and Y, defined as [99, Section 1.4]:

$$H(X,Y) = -\sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} p(x,y)\log p(x,y). \tag{1.39}$$

In particular, when X and Y represent the random variables associated with two consecutive outputs of the TRNG, the joined entropy makes it possible to evaluate the leak of information when observing these outputs. Notions of entropy, joined entropy and mutual information are related to one another as depicted in Figure 1.10.

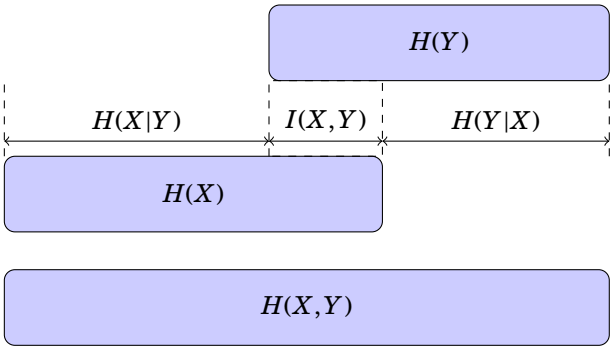


Figure 1.10: Relationship between various measures of information content.

From this discussion, it appears that entropy and its variates are good measures to estimate the level of randomness of a source, but also to study the dependences between its outputs. It is therefore not surprising that entropy and mutual information are used in modern evaluation standards of TRNGs.

1.4 Evaluation of TRNGs

We have seen in Section 1.1 that there are various ways to produce sequences of random numbers. Because of their crucial importance in cryptography, it is mandatory to evaluate their quality. Indeed, L'Ecuyer gave examples of RNGs which produce "bad" sequences of random numbers [43]. The use of such inappropriate generators may weaken the security of the whole cryptographic construction, especially when these numbers are used to generate cryptographic keys. Despite their importance, it is surprising to notice that in early 2000s, neither the Information Technology Security Evaluation Criteria (ITSEC) nor the Common Criteria (CC) specifies any evaluation criteria for random numbers generators [15].

In various cases, the evaluation of a TRNG only consisted in statistical tests applied to the generator's output [107, 108, 109]. We will not detail the theory of statistical tests, but rather give a brief overview of this notion. We invite interested readers to refer to specialized documentation [110]. Broadly speaking, a statistical test checks if a given hypothesis is realistic based on data observation. A statistical hypothesis testing can be split into five steps [111].

- The first step consists in formulating the practical problem in terms of hypotheses H_0 and H_a. The H_0 hypothesis, often called null hypothesis, represents the *status quo*. It is a statement one cannot prove in practice (for example, a given RNG produces actual random numbers). H_a is often called the alternative (or action) hypothesis and should be relatively easy to prove using data. These hypothesis are chosen such that they are mutually exclusive.

- Once the hypothesis have been formulated, one has to compute the test statistic (T) which depends only on the data. Test statistics are required to:

  - behave differently when H_0 is true from when H_a is true,
  - have a probability distribution computable under the assumption that H_0 is true.

- On the basis of the test statistic, a critical region must be defined. This critical region is the set of values of T for which the H_a hypothesis is more likely to be true than H_0. This region can be of three types:

  - right-sided (see Figure 1.11), in which case, one rejects H_0 if T is greater than or equal to some (right) critical value;

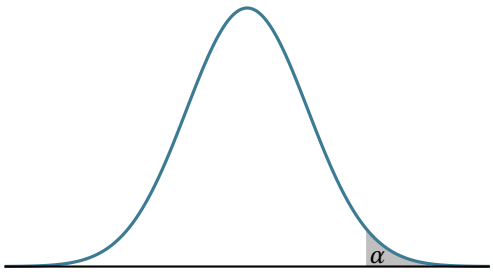


Figure 1.11: Example of a right-sided critical region.

  - left-sided (see Figure 1.12), which is equivalent to the prior case, except that here one rejects H_0 if T is less than or equal to some (left) critical value;
  - both-sided (see Figure 1.13), which can be considered as the logical disjunction of the previous cases, here one rejects H_0 if T is either greater than or equal to the right critical value, or less than or equal to the left critical value.

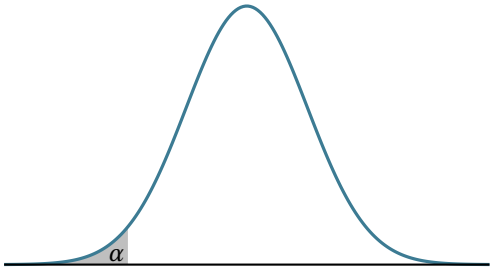


Figure 1.12: Example of a left-sided critical region.

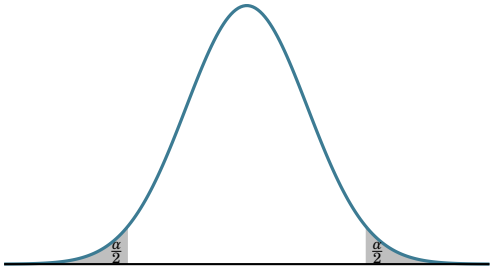


Figure 1.13: Example of a both-sided critical region.

- When performing a hypothesis testing, it is always possible to end with the wrong conclusion. It is thus a good practice to specify the level of risk we accept to run of coming to an incorrect conclusion. For that, statisticians define the significance level or size of the test, denoted α. It is a real number between 0 and 1 which defines the size of the critical region and corresponds to the probability of rejecting H_0, while it is actually true. This is referred to as an error of the first type or a Type I error. The value of α must be set depending on the severity of the consequences of making such an error. Therefore, the fact that a hypothesis testing results in an error of type I is the complete responsibility of the one performing the test, since he is the one choosing value for α.

- The last step of an hypothesis testing consists in concluding if H_0 should be rejected or not. A value of T lying in the critical region will lead to reject H_0 in favor of H_α with significance α. If T lies outside the critical region, one does not reject H_0.

When concluding a statistical test, it is important to note that H_0 is never accepted. Either it is rejected, or we lack enough evidence to reject it. Indeed, it is possible that H_0 is not true, but because of the data (too few, outliers, ...) one fails to reject it. This is called an error of the second type or Type II error. This type of error mainly depends on the quality of the data (sample size, population variance, ...) over which the tester does not always have control. Its probability of occurrence is usually denoted β.

Thus in testing any statistical hypothesis, there are four possible situations which determine whether the decision is correct or not. They are summarized in Table 1.1.

		Situation	
		H_0 is true	H_0 is false
Decision about H_0	fail to reject	correct decision	Type II error
	reject	Type I error	correct decision

Table 1.1: Overview of error types in statistical tests.

Although the rejection of a "good" TRNG may be problematic, it actually results in no security flaw. This is not the case of a Type II error which would lead evaluators to validate an insecure generator. It is therefore important to consider the probability of rejecting H_0 while it is actually false. This probability, computed as:

$$\gamma := 1 - \beta, \tag{1.40}$$

is called the power of the test and tells how likely a statistical test is to detect a "bad" generator. Of course, for security reasons, one wants this power to be as high as possible in order to minimize the risk of validating bad generators.

However, an increase of a test power will results in an increase of the probability of rejecting "good" generators [112, Section 8.9]. It is therefore important to find a compromise between acceptance and rejection rate in order to reject as many "bad" RNGs as possible without rejecting good ones. This is actually not an easy task and mainly depends on the consequences that a wrong decision will have on the target application.

For this reason, statisticians prefer another method which produces more accurate results. Indeed, the method presented above verifies whether the test statistic T is in a critical region or not. It does not specify the distance of this value from the critical threshold α_0. This threshold corresponds to the minimum probability of rejecting H_0 while it is true. In other words, it is the minimum risk to be accepted if we want to reject the null hypothesis H_0. This value is known as the p-value and is obtained by numerical methods using available data. For decision making, this p-value is compared to[7] α:

- if it is smaller than α, the null hypothesis H_0 is rejected,
- otherwise, one fails to reject H_0.

The use of the p-value facilitate the task of finding a good compromise when it comes to rejecting a generator. Indeed, it quantifies how bad a generator is, since it also gives the distance to the

[7] NIST test suites use $\alpha = 10^{-2}$ [113, Section 1.1.5].

pre-defined bound. Note that a p-value close to this bound may indicate insufficient data quality or quantity.

1.4.1 Classical evaluation approach of TRNGs

As stated earlier, the first attempts to assess the quality of a generator mainly focused on the output bitstream. This approach, depicted in Figure 1.14, considered the generator as a black box producing bitstream. To assess the quality of the generator, the only possible way was therefore to evaluate its outputs. Statistical hypothesis testing were performed on generated sequences of bits to check their randomness. In practice, these tests are checking two characteristic properties of random sequences [114]:

- values in the sequence are uniformly distributed,
- each value in the sequence is obtained independently from the previous ones.

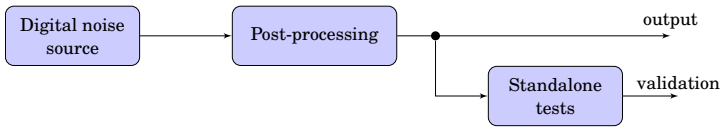


Figure 1.14: TRNG classical evaluation approach.

The goal of a statistical test is to detect a pattern specified in H_a (bias, correlation, etc). The interest resides in the identification of suspicious sequences of bits at the output of the generator. Indeed, "bad" generators tend to produce defective sequences of bits that might be detected by statistical tests. The use of these tests can thus help discard "bad" random number generators. Since a statistical test tries to detect a pattern in the generated sequence, and that several patterns have to be checked in order to conclude on a statistical property, the various randomness tests consist of batteries of statistical tests. Some of these tests (like DIEHARD) are more oriented towards problems related to DRNGs [115], while others are more focused on TRNGs [116]. It is also possible to find general-purpose tests [117, 113].

The success of randomness tests resides in their ability to detect properties which contradict the random nature of a sequence of bits. Knowing that there is an infinite number of statistical properties a truly random sequence must meet, the use of a finite number of tests does not guarantee that the output of a RNG is random. More to the point, it cannot conclude on the quality of the the generator under test. Even though statistical tests can detect some pattern with a given probability, it is possible to construct deterministic sequences that pass all these tests successfully [118]. This shows that statistical tests are necessary, but not sufficient to assess the quality of an RNG.

On the other hand, most RNGs encountered in real-life applications are hybrid. They can be split into two basic blocks: a digital noise source and a post-processor as depicted in Figure 1.15. The digital noise source is actually the block in which takes place the random physical phenomenon. This is the phenomenon responsible of the randomness in the generated sequences of bits. It is the non-deterministic part of the generator. The second block performs an algorithmic post-processing on the sequence of bits generated by the digital noise source. It is usually a DRNG or a cryptographic function (hash function, fixed-keyed AES, *etc*). Its use makes the output of the generator look random even if the post-processor does not use any data compression algorithm, *i.e.* it does not change entropy rate as graphically depicted in Figure 1.16.

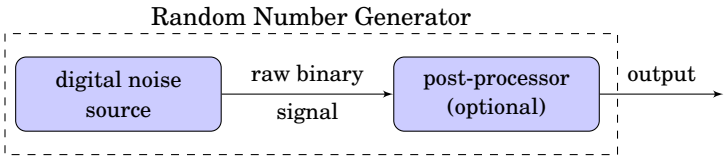


Figure 1.15: Main blocks of a TRNG.

In this figure, one can see that the flawed data plotted before the post-processing look random after being post-processed. This action can lead statistical tests into wrong results: concluding the generator is of good quality while it is actually a bad one. based on this observation, one can say that statistical tests should be applied to the output of the digital noise source instead of the output of the generator.

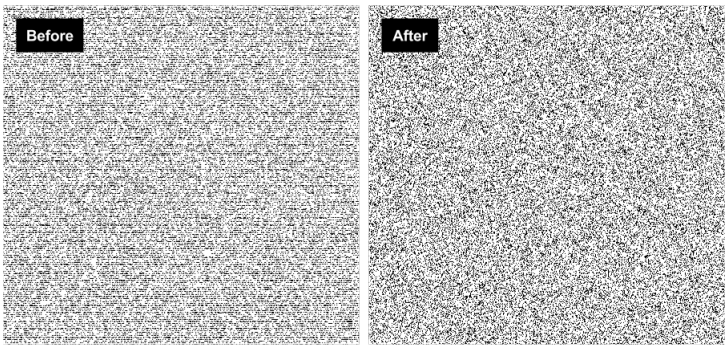


Figure 1.16: Action of a post-processing.

Moreover, caution should be taken while choosing the statistical tests suite, since ad-hoc ones can easily contain errors [119, 120]. Furthermore, assessing a generator must not entirely rely upon statistical tests since they cannot evaluate the entropy, which is actually the quantity that characterizes the level of randomness of the source. Even though some test suites incorporate an entropy estimate, this estimate is done using the generated bit stream, *i.e* on the realizations of

random variables [18]. However, as it was seen in Section 1.3, the entropy is a property related to random variables, not to their realizations [24]. Since randomness comes from the physical phenomenon occurring inside the digital noise source, it is crucial to also evaluate the physical phenomenon which produces the random sequence at the output of the generator. This is the core idea of the approach presented in the next section.

1.4.2 Enhanced evaluation approach of TRNGs

As shown in Figure 1.14, the generator of interest was considered as a black box which outputs sequences of bits. Based on this consideration, RNGs were exclusively evaluated using statistical tests. This practice was mainly due to the lack of common evaluation criteria for RNGs. In order to remedy this situation, the BSI (Bundesamt für Sicherheit in der Informationstechnik) issued in 1999, then in 2001 series of recommendations on how to assess RNGs. The first series, commonly known as AIS-20 addresses the evaluation of DRNGs [29], while the second series deals with evaluation of TRNGs [15, 24]. Through these series of recommendations, BSI adopted an enhanced approach for evaluating RNGs. Indeed, it introduces a classification of RNGs based on their nature (DRNG or TRNG), the intended security level and assessment feasibility. It also recommends a new type of TRNG design scheme which allows assessing internal parts of the TRNG. Even though the standard comprises recommendations on both DRNGs and TRNGs, we will only discuss those related to TRNGs in what follows.

From the AIS-31 perspective [24], internal parts of the TRNG (such as source of randomness, internal signals, etc) should be assessable. This requires the identification of each component of the generator: noise source, harvesting mechanism, post-processor. Based on the intended application and security level, AIS-31 distinguishes three different classes of TRNGs.

- The class PTG.1 defines requirements of TRNGs for cryptographic applications in which unpredictability of generated bits is not required. It therefore allows generated bits to be guessed with a probability higher than 0.5. Generators from this class are only required to produce statistically random bits. To ensure that, they are required to have an embedded total failure test and some non-embedded online tests as depicted in Figure 1.17. The goal of the total failure test is to detect any sudden drop of entropy in the source. AIS-31 recommends this test to be launched both when the generator starts operating and during operation. Online tests are set to detect statistical defects of the output during operation of the TRNG. These latter tests can be either triggered periodically or applied continuously to the TRNG's output. In the event any of these tests triggers an alarm, the generator must stop operating.

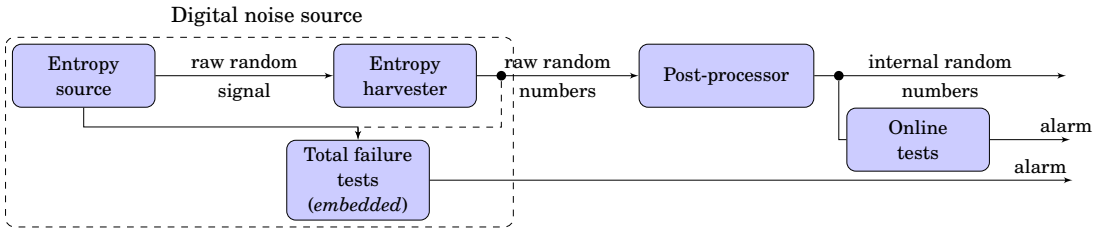


Figure 1.17: Assessment procedure of class PTG.1 TRNG.

- The class PTG.2 concerns TRNGs for which their applications require a high level of unpredictability. AIS-31 considers that such generators can be used for instance to generate random padding bits, seeds for DRNGs, or any other cryptographic application which requires secrecy of the generated numbers. Outputs are not required to be indistinguishable from independent and uniformly distributed random numbers. The most important aspect is that the entropy per bit should be high (0.997 in the case of Shannon entropy) so it could resist guessing attacks. For this reason, PTG.2 class adds to PTG.1 requirements a stochastic model of the source of randomness. However, online tests should rather be applied to the raw random signal instead of the internal random numbers. Because generators of this class can display some bias at the output, BSI recommends the use of a more restrictive class for generating cryptographic keys.

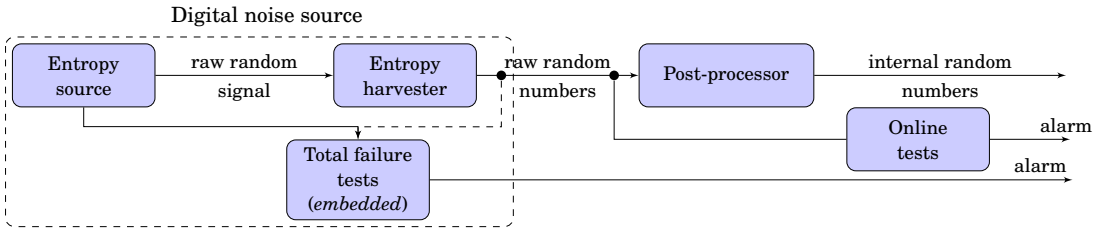


Figure 1.18: Assessment procedure of class PTG.2 TRNG.

- The class PTG.3 is the highest defined by AIS-31 and is intended for very sensitive security applications. It basically consists of PTG.2 compliant generators with cryptographic post-processing. Thus, their security does not only rely on the quality of the source of randomness, but also on the computational security ensured by the cryptographic post-processing algorithm. As a consequence, the generator output must not exhibit any bias nor short term dependencies.

From the above description of various TRNG classes, it follows that a stochastic model of the generator is compulsory to ensure high level of security. This is because the assessment of a TRNG cannot be done without actually identifying and analyzing the characteristics of the system or process which impacts the entropy. Ideally, this should be done through a physical model based

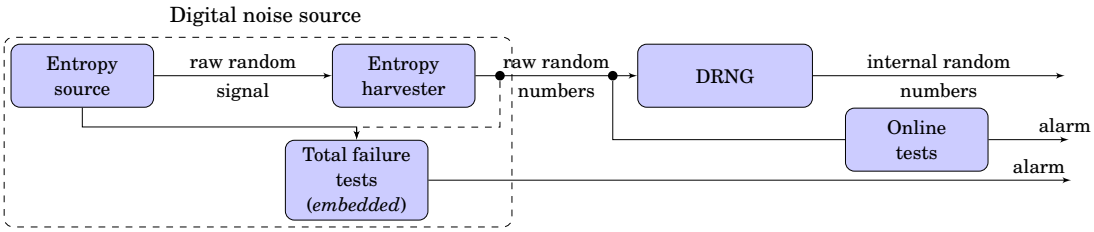


Figure 1.19: Assessment procedure of class PTG.3 TRNG.

on electronic circuits. However, the knowledge required for such model is daunting whereas a stochastic model is less complicated. Hence, the stochastic model is a convenient trade-off that provides a class of probability distributions among which is the true (but unknown) distribution of the TRNG's output [24]. The main goal of the stochastic model is to make the connection between the unpredictability of the source of randomness and the output bit stream of the TRNG. For this purpose, the stochastic model takes various parameters $p_1, p_2, \ldots, p_n$ as inputs. Example of these parameters are jitter variance, jitter drift and clock signal duty cycle. Based on these parameters, it should provide the entropy, or in the worst case a lower bound for the entropy of the source as depicted in Figure 1.20. This expected entropy is to be compared with the one required in the standard to guarantee security of the generator.

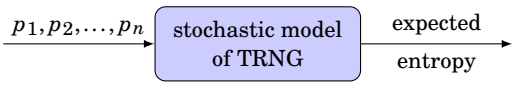
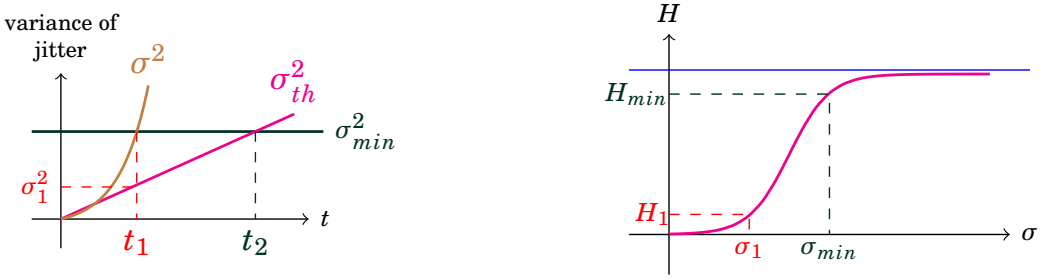


Figure 1.20: Use of a stochastic model ($p_1, p_2, \ldots, p_n$: model parameters).

As it is the case for any modeling procedure, assumptions must be made. Existing models of TRNG assume that the jitter is only due to thermal noise. Section 1.2.3 shows that this assumption is not true, since jitter has several components. Two strategies can then be used: either provide new models that take into account effects of other jitter components, or develop a jitter measurement method which provides the proportion of the jitter variance due to thermal noise. This latter is known as the conservative approach which bases entropy computation only on the jitter caused by the thermal noise, because it is considered to be non manipulable. If neither of the above strategies is adopted, one could overestimate the entropy of the source of randomness, yielding a serious security flaw for cryptographic schemes using that generator. Indeed, standards set a minimum entropy threshold H_{min} which ensures a given security level. Stochastic models can then be used to estimate a minimum jitter variance σ^2_{min} corresponding to this entropy threshold. Under the assumption that jitter contains only the thermal noise, one can consider that jitter variance has a linear progression over time which reaches σ^2_{min} at a instant t_2 as shown in Figure 1.21. Since real-world jitter contains other phenomena in addition to the thermal noise, the evolution of jit-

ter variance over time is not linear and reaches σ^2_{min} at a time instant t_1, yielding a variance $\sigma^2_1 \leqslant \sigma^2_{min}$ due to thermal noise. The variance σ^2_1 therefore results in an entropy H_1 very small compared to H_{min}. Hence, various sources contributing to the randomness used by the generator must be clearly identified to avoid severe security flaws [121].



(a) Evolution of the jitter variance as a function of accumulation time.

(b) Evolution of entropy as a function of the jitter standard deviation.

Figure 1.21: Risk of entropy overestimation.

Although AIS-31 offers a good framework for evaluating TRNGs, it does not take into account the technology on which the generator is implemented (ASIC or FPGA), effects of the power supply, temperature, electromagnetic perturbations and so forth. However, the performance of a TRNG can be altered whenever the operating conditions of a generator change [122]. This study latter demonstrates that the quality of generated random bits highly depends on environmental conditions of the TRNG. It was recommended in the first place that a distinction should be made between the analog noise (which is actually the phenomenon responsible for the random behavior of the generator) and the digital noise (which is a discrete version of the analog noise). Furthermore, a deep study of the source of randomness should be performed, resulting in a stochastic model which will help monitoring the generator in order to detect any potential problem. An improvement of this proposal was developed by DGA-MI (Direction Générale de l'Armement-Maîtrise de l'Information) which requires among others that various physical noises inside the source of randomness should be identified and characterized in addition to the model of the source of randomness, as depicted in Figure 1.22.

This approach was justified and illustrated by David Lubicz using the elementary generator based on ring oscillators [95]. The physical noise source of this generator consists of two ROs (RO_1 and RO_2). RO_2 is considered ideal and produces the clock signal, while the signal produced by RO_1 is the jittered one. This jitter is impacted by the thermal noise which is modeled as a Wiener process of drift μ and volatility σ. These μ and σ are the physical parameters of the noise source. The entropy accumulation is done by using a frequency divider K, which is the only parameter of the

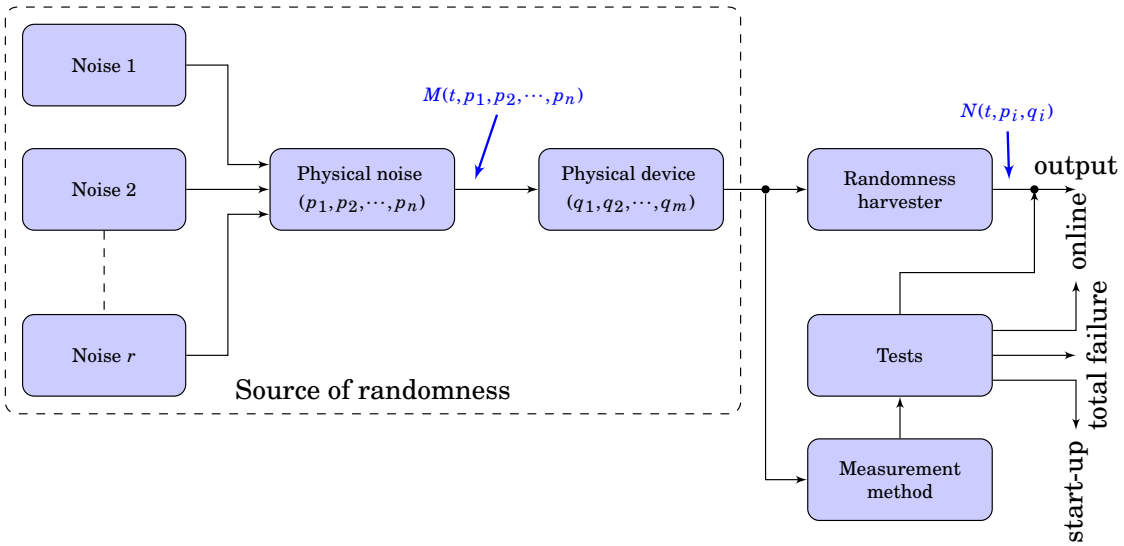


Figure 1.22: TRNG evaluation approach of DGA-MI (DGA-MI: Direction Générale de l'Armement-Maîtrise de l'Information).

generator. Hence, the physical model of this generator can be summarized as:

$$M(t,p_1,p_2) = G\left(t,\mu\Delta t,\sqrt{\Delta t}\sigma\right), \tag{1.41}$$

yielding $p_1 = \mu, p_2 = \sigma$ and $q = K$.

In order not to force TRNG designers to use only one principle, DGA-MI requested that this approach is illustrated by another principle. This request constitutes the core of Chapter 4, where PLL-based TRNGs will be used for this purpose.

The goal of this work is to provide a method to estimate the exact proportion of thermal noise in the jitter, and also study how parameters of a given TRNG affect the randomness quality of the jitter. This would be a progress towards a more accurate estimate of the entropy, yielding a better security evaluation of TRNGs. Moreover, the knowledge of the effects that various parameters could have on the quality of entropy would help optimizing these parameters to ensure maximum entropy at the generator's output.

1.5 Conclusion

Random number generators (RNGs) represent a crucial cryptographic primitive since they impact the security of any construction they are part of. Due to their importance in computer security, the choice of a TRNG should not be made at random. Caution must be taken while designing an RNG since various methods aiming at generating random numbers in the state-of-the-art ended

up with huge security flaws affecting the whole cryptosystem. Based on the method through which they operate, one can distinguish deterministic random number generators (DRNGs) and true random number generators (TRNGs). The latter, including the subclass of generators using physical methods is the one we will focus on, especially hardware ones that can be implemented in logic devices.

In the choice of a suitable generator for cryptographic use, it is necessary to have evaluation methods available. However, there was no standard evaluation criteria of RNGs until recently. Most evaluation consisted in performing statistical tests on the output of RNGs regardless of their constitutions. As a result, several successful attacks were performed on various TRNGs, revealing the need to investigate in depth how each TRNG is built. It was thus highlighted that statistical tests are not sufficient to assess a TRNG, they should always be interpreted with respect to the functionalities of the generator. Hence, the mechanism through which random numbers are obtained is required to be fully understood and mastered.

It happens that TRNGs exploit random physical phenomena to produce series of random numbers. It is admitted that generated numbers inherit randomness from physical phenomena. Thus, when assessing a TRNG, attention must be given to the source of the physical random phenomena. In the case of TRNGs we studied, the source of randomness consists in various electronic noises among which the thermal noise is the only accepted noise for generating random numbers. This is because it is non-manipulable and its modeling is well characterized by a Wiener process. It was shown that most of these noises are not suitable for generating genuine random numbers. A genuine randomness extraction mechanism is therefore compulsory to ensure that only safe noises are exploited for generating random numbers.

Because entropy is a measure of the level of randomness contained in the source, it is important to provide a link between the entropy of the source and measurable physical parameters in order to guarantee the quality of generated random numbers. This requires the availability of a stochastic model describing the source of randomness, and making the bridge between measured variance of the jitter and required entropy for security. This model goes through a better understanding of that source of randomness, the identification and characterization of various random phenomena that will be detailed in Chapter 2.

Résumé

Les nombres aléatoires sont d'une importance capitale pour la cryptographique moderne. Ils sont obtenus par les générateurs de nombres aléatoires, qui peuvent être repartis en deux grandes familles suivant leurs principes de fonctionnement.

- La première famille est celle des générateurs dits déterministes. Ils utilisent des méthodes algorithmiques pour produire des suites de nombres dont les propriétés statistiques ressemblent à celles des nombres véritablement aléatoires. Pour cette raison, tout générateur de cette famille est appelé générateur de nombres pseudo-aléatoire (PRNG) ou générateur déterministe de nombres aléatoires (DRNG).

- La seconde famille est celle des générateurs des nombres véritablement aléatoires (TRNGs). Comme leur nom l'indique, ils génèrent des suites de nombres véritablement aléatoires à partir de phénomènes imprévisibles. Ces phénomènes peuvent être d'origine physique (bruit électronique, radioactivité, etc) ou non physique (mouvements de la souris d'un ordinateur, délai de lecture/écriture sur un disque dur, etc).

Chacune des deux familles ci-dessus présente des avantages qui lui sont propres. Les DRNGs ont des débits très élevés et produisent des suites de nombres ayant de très bonnes propriétés statistiques. Les TRNGs quant à eux ont des sorties réellement aléatoires. Afin de bénéficier des avantages des deux familles, il est courant d'associer un DRNG à un TRNG. On parle alors de générateur hybride de nombres aléatoires. Selon la manière de les combiner, on distingue :

- les générateurs hybrides de nombres véritablement aléatoires (HTRNG), qui génèrent des suites de nombres aléatoire en utilisant un TRNG ; ces nombres passent ensuite par un DRNG pour une étape de post-traitement visant à améliorer leurs propriétés statistiques ;

- les générateurs hybrides de nombres pesudo-aléatoires (HDRNG), dans lesquels se trouvent un TRNG qui réinitialise de manière périodique un DRNG.

Pour les objectifs de cette thèse, nous nous focaliserons sur les TRNGs implémentés dans des circuits logiques utilisant des phénomènes physiques comme source d'aléa. La plus utilisée de ces sources est le jitter d'horloge que nous adoptons également comme source d'aléa pour l'étude faite au cours de cette thèse. Compte tenu de leur importance en cryptographie, des procédures de standardisation, telle que l'AIS-20/31, ont été développées afin d'évaluer la sécurité des générateurs de nombres aléatoires. Pour des applications très sensibles, de niveau militaire par exemple, la DGA recommande en plus de la conformité à l'AIS-20/31, une analyse plus poussée de la source d'aléa afin d'en obtenir une modélisation plus fine. Tout au long de cette thèse, nous nous attarderons donc sur la compréhension et la caractérisation de la source d'aléa, ainsi que des divers

phénomènes qui s'y produisent. Une illustration de cette démarche sera faite avec les TRNGs basés sur les PLLs.

Chapter 2

Characterization of clock jitter as a source of randomness

Contents

Random number generators are security critical cryptographic primitives that need to be assessed as rigorously as possible. This implies having a well defined source of randomness and a thorough

characterization of that source. In this chapter, we present a model of the clock jitter and an estimate of the exact contribution of the thermal noise that affects it. A randomness harvesting method compatible with the measurement of the thermal noise contribution is also developed.

2.1 Random signal

Electronic noises present in digital devices can be considered as parasitic random signals[1]. A time varying signal for which its instantaneous value cannot be determined or predicted [123, Page 3]. Mathematically, they are modeled as random processes and characterized using two approaches, namely probabilistic and statistical [123, Page 209]. The probabilistic approach, although more accurate, cannot be directly applied to real-world random signals. The usual way is to derive a statistical description of the signal, that uses parameters provided by the probabilistic approach and estimated by observations of the random signal.

In this section, we provide a brief description of various concepts related to the study of random signals. This description will then be applied to specify assumptions we will use to characterize the random phenomena occurring in the source of randomness of TRNGs.

2.1.1 Time and ensemble averages

When observing a random physical phenomenon, one has access to a random function of time $x(t)$. In fact, this random function is only one among an infinity of other random functions which could be accessed by observing the random phenomenon as depicted in Figure 2.1. The collection of all possible time functions that might have been observed is called a random process, and denoted $\{x(t)\}_{t\in\mathbb{R}_+}$ [124, Definition 3.1]. When the probability of occurrence of each random function $x(t)$ is defined, the random process is called an ensemble, and any member of that ensemble is called a sample function [123, Page 53], [125, Page 39].

Because the process is random, if t_i is an arbitrary time with $i \in \mathbb{N}$, then the value $x(t_i)$ is a random variable denoted x_i [124, Page 26]. Note that for two different instants t_i and t_j, the random variables x_i and x_j are not necessarily equal. In other words, when dealing with random processes, there is a random variable for each instant of time. Moreover, the random character of interest is the one that exists from one sample function to another in the ensemble. Therefore, the probability description of random variables deduced from a random process is also the probability description of the process.

[1] Other parasitic signals also exist as mentioned in Section 1.2.3. They are deterministic and often denoted interferences. These signals are not taken into consideration since the differential principle mitigates their effects.

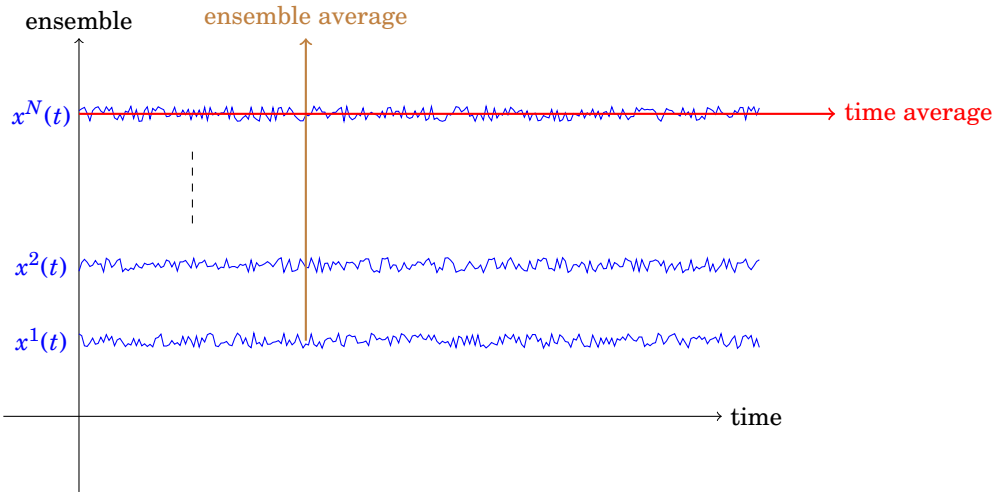


Figure 2.1: Sample functions of an ensemble.

Due to the random nature of the process, it cannot be characterized based on a single event of that process at a certain time. It can only be discussed in terms of averaged quantities, either of a single system (sample function) over a time interval, or of many identical systems at a certain time instance. The former is called time average and the latter ensemble average [126, Section 1.1]. As an example, if $x(t)$ is a sample function of the process, one can define the kth-order time average as:

$$\overline{[x(t)]^k} := \lim_{T\to\infty} \frac{1}{2T} \int_{-T}^{T} [x(t)]^k dt. \tag{2.1}$$

Thus, time averages of a random process are defined for a given sample function along the time axis. The ensemble average, on the other hand, is defined for a given time instant along the ensemble axis as shown in Figure 2.1. The first-order ensemble average, at a given time t_i, is:

$$\langle x(t_i) \rangle := \lim_{N\to\infty} \frac{1}{N} \sum_{k=1}^{N} x^k(t_i) = \int_{-\infty}^{+\infty} \xi f_{x_i}(\xi) d\xi, \tag{2.2}$$

where f_{x_i} is the probability density function of the random variable $x_i := x(t_i)$, and each $x^k(t)$ represents a sample function of the process.

Ensemble averages are convenient since they are related to probability density functions of the random process at each time. This enables a theoretical analysis of the physical system and provides an insight into the different processes taking place in the system. However, computation of these quantities assumes a freeze of the time and the possibility to access all possible values of the process at a specific time. Of course, this is not feasible in experiments. One rather has access to a sample function which yields time averages. However, from the above definitions of time and ensemble averages, one understands that these two concepts are not the same. Actually, they coincide only for a special class of random processes which will be detailed in the next section.

2.1.2 Classification of random processes

Various kinds of random processes arise in engineering problems. These random processes are usually grouped into classes of pairs. According to the kind of problem one is dealing with, the random processes can either be [123, Section 5.1] :

- continuous or discrete,
- deterministic or nondeterministic,
- stationary or nonstationary,
- ergodic or nonergodic.

2.1.2.1 Continuous processes

A random process is said to be either continuous or discrete according to the set of possible values the random variables may have [123, Section 5.2]. A continuous random process is one for which the probability distribution function is continuous. This implies that random variables $x_1, x_2, \cdots$ may have any value within a range of possible values. Thermal noise in electronic devices is an example of such processes. A random process which is not continuous is said to be discrete.

2.1.2.2 Deterministic processes

While discussing random processes, it might seem strange to think some may be deterministic. However, some random processes are such that for a given sample function, it is possible to predict future values from the knowledge of past ones. An example of such processes is the one for which each sample function is defined as :

$$x(t) = a\cos(\omega t + \theta), \tag{2.3}$$

where a and ω are constants and θ is the initial phase corresponding the to realization of a random variable. The choice of the value of θ therefore defines the sample function, since it will not change again. Although one cannot predict which value θ will have, once it is chosen, the behavior of the process becomes deterministic. The interest of this category of random processes is mostly theoretical, since almost all practical random processes are nondeterministic [127, Section 6.3]. Specifically, electronic noises we are dealing with, in the framework of random number generation, are nondeterministic. This means that, for each sample function, future values of the process cannot be predicted from the observed past values [123, Section 5.3]. We shall therefore consider only nondeterministic random processes in the rest of this thesis.

2.1.2.3 Stationarity

Let $t_1, \cdots, t_k$, for any value of $k \in \mathbb{N} \setminus \{0\}$, be different times. Since $\{x(t)\}$ is a random process, each $x_i := x(t_i)$ is a random variable. It is therefore possible to define the joint probability distribution of the random vectors $(x_i)_{1 \leqslant i \leqslant k}$ and $(x_{m+i})_{0 \leqslant i \leqslant k}$ for any $m \in \mathbb{N} \setminus \{0\}$. The process $\{x(t)\}$ is said to be stationary if the random vectors $(x_i)_{1 \leqslant i \leqslant k}$ and $(x_{m+i})_{0 \leqslant i \leqslant k}$ have the same distribution [124, Definition 13.1]. Otherwise, it is said to be nonstationary [123, Section 5.4].

If $k = 1$, we can see that if $x(t)$ is stationary, then all the random variables x_i's for $i \in \mathbb{N}$ have the same probability distribution. This shows that the stationarity of a process implies that the statistics of that process do not change over time. In other words, the behavior of the process does not change with time. This leads to the fact that all ensemble averages are independent from the time origin, and therefore statistics of the process can be defined at any time.

In the real-world, no random process is actually stationary. Indeed, any physical phenomenon starts at a specific time and does not last forever. However, in many physical situations, it may happen that the process does not change a lot during the time it is being studied. In these cases, the stationarity assumption leads to a convenient mathematical model, which closely approximates reality. In general, determining whether the stationarity assumption is reasonable or not, for a given situation, may not be easy. It is however possible to apply statistical tests on the data to provide confidence on the stationarity assumption [128, 129, 130].

Wide sense stationarity In the definition of a stationary process, one requires the property to be valid for any vector of length k. In a rigorous way, the random process is said to be stationary of order k [126, Section 1.2]. From a practical point of view, this definition is too restrictive and not useful. In many practical cases, stationarity of order 2 is more adequate for signal and system analysis [123, Page 196]. For random process that meet this latter, often named wide-sense stationary processes, (x_1, x_2) and $(x_1 + \varepsilon, x_2 + \varepsilon)$ have the same joint probability distribution, for any $\varepsilon \in \mathbb{R}_+ \setminus \{0\}$. This is equivalent to [126, Section 1.2]:

- $\langle x(t) \rangle$ and $\langle x(t)^2 \rangle$ are independent of the time t,
- $\langle x(t_1)x(t_2) \rangle$ only depends on the time difference $\tau = t_2 - t_1$.

In others words, moments of order 1 and 2 are constant, and the autocorrelation function depends only on the time difference $\tau = t_2 - t_1$. Moments of higher orders are subject to no requirements.

2.1.2.4 Ergodicity

Characterization of electronic noises requires to compute the noise statistics. In practice, one has access to a single sample function of the noise process which yields time averages. However, information about the noise process is in the ensemble averages which are not (in general) equal to time averages according to Section 2.1.1. This is where comes in the notion of ergodicity which guarantees that time and ensemble averages coincide [126, Section 1.1]. Thus, whenever one is dealing with an ergodic process, any sample function obtained through experimental measurements represents the entire process.

An important consequence of what was discussed above is that any ergodic process is necessarily stationary as shown in Figure 2.2 [126, Section 1.2]. Indeed, ergodicity states that time averages equal corresponding ensemble averages. The first order time average of a sample function, also called mean, is a number usually denoted μ. On the other hand, the first order ensemble average is obtained by taking the average across all sample functions at a time instant. Thus, if the process was not stationary, one would end up having two different first order ensemble averages μ_1 and μ_2 at instants t_1 and t_2. Because the process is ergodic, we should have $\mu_1 = \mu = \mu_2$, contradicting $\mu_1 \neq \mu_2$.

From the above, it follows that the statistics (mean, variance, etc.) of a non-stationary process are not defined. Thus, it is necessary for the process to be stationary so that its variance is meaningful. Moreover, the measurement instruments used in practice evaluate time averages and not ensemble averages. In order for these time averages to coincide with the ensemble averages used in theory, the process must be ergodic. We will therefore assume that electronic noises we are dealing with are ergodic, therefore stationary, as is done in most state-of-the-art studies [131, 132, 133].

2.2 Mathematical model of the clock jitter

The source of randomness we use to generate random numbers is the clock jitter in logic devices. This clock jitter is caused by electronic noises. We saw in Section 1.4.2 that jitter has several components while existing models of TRNGs assume that only the jitter coming from the thermal noise contributes to entropy. It is therefore of prime importance to evaluate the exact contribution of the thermal noise to the jitter. To solve this problem, we develop in this section a method aimed at estimating the part of jitter due to thermal noise.

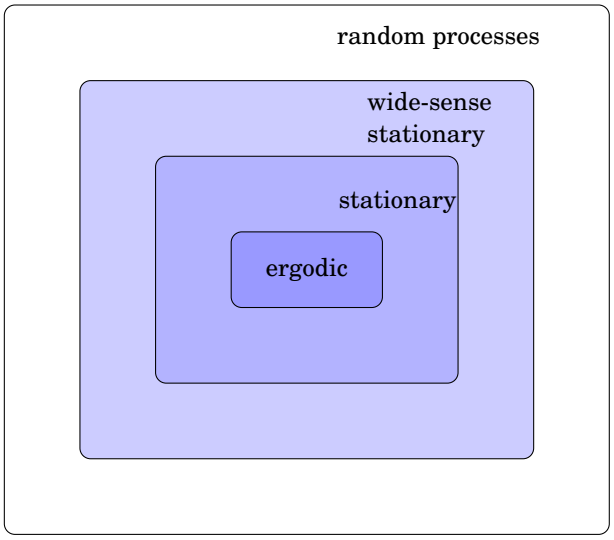


Figure 2.2: Subsets of random processes.

2.2.1 Characterizing noise in time domain

The study of ergodic signals can be made using two complementary approaches: time domain and Fourier frequency domain. Based on the approach used, they are often characterized by either the average power (or variance) or the frequency spectrum. Because of its random nature, the power of a random signal may be different from one time to another. A possible way to get around this difficulty is to measure the average power over a specified time interval [123, Section 2.4]. Thanks to the stationarity of the process, this value should remain the same for any time interval of the same length.

Random fluctuations of the signal of an oscillator being modeled as random processes, the above discussion applies. We will therefore use these considerations to provide a time domain characterization of random fluctuations of an oscillator that will be used as source of randomness in the design of a TRNG.

2.2.1.1 Oscillator output signal

As said in Section 1.1.2, generators of interest in this work are those which are used in oscillator-based TRNGs. The source of randomness for these generators is the clock jitter of the oscillator generating the jittered clock. A real-world oscillator usually outputs a signal, called clock signal, that can be modeled as [134]:

$$v(t) = \big(v_0 + \eta(t)\big)\sin\big(2\pi\nu_0 t + \Phi(t)\big). \tag{2.4}$$

In Equation (2.4), v_0 and ν_0 are respectively the nominal amplitude and frequency of the clock

signal. $\{\eta(t)\}_{t\in\mathbb{R}}$ is a random process denoting amplitude fluctuations around v_0, often known as amplitude noise [81]. In the field of random number generation, we are using only digital devices because we assume discrete levels, therefore we can neglect $\eta(t)$. For this reason, we will assume that there are no amplitude fluctuations in the output signal of oscillators, and therefore focus on the phase jitter.

Equation (2.4) exhibits another term, $\Phi(t)$, which represents phase fluctuations, also called excess phase [81]. Knowing that the instantaneous frequency is the time derivative of the phase divided by 2π, any frequency variation implies a related phase variation. $\Phi(t)$ therefore takes into account any phase fluctuation, but also any frequency fluctuation. Hence, several terms can appear in $\Phi(t)$, among others phase drifts, periodic phenomena, random phenomena, etc [135]. A specific example, shown in Figure 2.3, reads as [134]:

$$\Phi(t) = D_1 t^2 + \Delta\Phi \sin(2\pi f_m t) + \varphi(t), \tag{2.5}$$

where phase drift is modeled as a second-order polynomial (linear frequency drift), periodic phase fluctuations are expressed as a sine function, $\varphi(t)$ denotes random phase fluctuations.

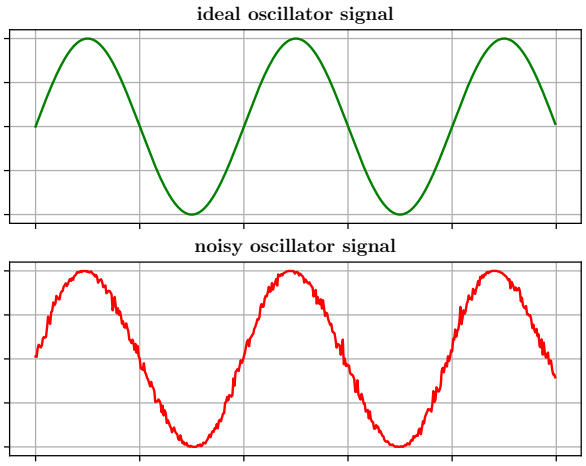


Figure 2.3: Examples of oscillators' output signals (noisy oscillator signal was generated using Equation (2.5)).

Phase fluctuations are due to specific physical mechanisms that make the phase of any real-world signal change continuously over time. Phase drifts are very slow changes, often referred to as *long-term instabilities*. Periodic fluctuations essentially come from the surrounding environment of the oscillator. They are caused by periodic electronic phenomena such as power-supply, environmental temperature, vibrations, pressure, etc. Random fluctuations are due to noise sources such

as thermal, shot and flicker noises encountered in electronic components [134]. Related fluctuations are often referred to as *short-term instabilities* since they are more significant when shorter time intervals are considered. Due to their random nature, statistical treatment is needed for their characterization.

When generating random numbers, random fluctuations of the phase are the only phenomena to consider. These fluctuations behave as a random process, represented by the random function $\varphi(t)$. Moreover, long-term instabilities are filtered out by the differential principle. In the remainder of this thesis, we will therefore deal with the simplified model of a clock signal [131]:

$$v(t) = v_0 \sin\left(2\pi\nu_0 t + \varphi(t)\right). \tag{2.6}$$

We recall that in Section 1.1.2, two oscillators are used to produce random sequences of bits. One is the reference oscillator, producing a reference clock signal, while the other is considered as the instable oscillator that produces a jittered clock signal. Actually, signals of both oscillators are jittered and can be modeled by Equation (2.6). However, to simplify the model, we transfer the jitter of the reference signal to the other signal. Hence, we can consider that random phase fluctuations of the jittered clock signal results from real-world phase fluctuations of both signals, while the reference clock signal is considered ideal. Note that this simplification can be made only under certain conditions which will be discussed in Section 2.4.1.

Moreover, practical applications deal with square waves instead of sine waves. This does not contradict the model of Equation (2.6). Indeed, such signals have two states: a lower state and an upper state. One can think of the change of state as zero crossing of the sine wave presented in Equation (2.6). The rising edge will therefore correspond to the zero crossing from negative to positive values, while a falling edge will correspond to the zero crossing from positive to negative values [81, Section 2.1]. This operation can be described as:

$$v(t) = \frac{1}{2} v_0 \Big(1 + \operatorname{sgn}\left[\sin\left(2\pi\nu_0 t + \varphi(t)\right)\right]\Big), \tag{2.7}$$

in particular to have discrete levels between 0 and v_0.

The model of a clock signal can have a more general form. Indeed, consider a generic periodic waveform $g(2\pi\nu_0 t)$ with exactly one positive zero crossing per period $T_0 = \frac{1}{\nu_0}$. The generalization of a clock signal can be described as [81]:

$$v(t) = v_0 g\left(2\pi\nu_0 t + \varphi(t)\right). \tag{2.8}$$

However, we will stick with the model defined by Equation (2.6) and bear in mind that all considerations about phase and excess phase for a sine wave can be transported to a square wave or any waveform defined by Equation (2.8).

2.2.1.2 Phase and frequency random fluctuations

Phase fluctuations in Equation (2.6) are expressed in radians. However, one can also express them in seconds, with Equation (2.6) being rewritten as:

$$v(t) = v_0 \sin\left(2\pi\nu_0\left[t + \frac{\varphi(t)}{2\pi\nu_0}\right]\right). \tag{2.9}$$

This introduces the instantaneous time error of the clock generated by the oscillator as [131]:

$$x(t) := \frac{\varphi(t)}{2\pi\nu_0}. \tag{2.10}$$

This quantity represents the time difference between a real-world and an ideal clock signal (without fluctuations). It is the excess time needed to reach rising edges of real-world clock signals. The clock jitter of an oscillator can therefore be modeled by Equation (2.10).

Since phase and frequency of any signal are related, one can express the instantaneous frequency of the clock signal in Equation (2.4) as [131]:

$$\nu(t) := \frac{1}{2\pi}\frac{\mathrm{d}}{\mathrm{d}t}\left(2\pi\nu_0 t + \varphi(t)\right), \tag{2.11}$$

which can be rewritten:

$$\nu(t) := \nu_0 + \frac{1}{2\pi}\frac{\mathrm{d}\varphi(t)}{\mathrm{d}t} = \nu_0 + \Delta\nu(t), \tag{2.12}$$

where

$$\Delta\nu(t) := \frac{1}{2\pi}\frac{\mathrm{d}\varphi(t)}{\mathrm{d}t}. \tag{2.13}$$

represents frequency fluctuations of the oscillator.

Equation (2.13) shows that phase and frequency fluctuations are actually two representations of the same phenomenon. Hence, one can gain knowledge about phase fluctuations from the study of frequency fluctuations, and vice-versa. Therefore, one can study and characterize frequency fluctuations, since they are easier to assess.

Even though studying frequency fluctuations seems easier than studying phase fluctuations, the instantaneous frequency $\nu(t)$ is not observable. This comes from the fact that any frequency-measurement technique involves a finite time interval over which the measurement is performed [134]. The basic principle consists in counting, from a time t_k, the number of periods of the

oscillator during a time interval τ defined by the reference oscillator. Hence, one deduces the average value of $v(t)$ during a time interval of length τ:

$$\langle v(t)\rangle_{t_k,\tau} := \frac{1}{\tau}\int_{t_k}^{t_k+\tau} v(\theta)d\theta = \frac{n_k}{\tau}, \tag{2.14}$$

where n_k is the number of periods of the oscillator during the time interval[2] $\tau > \frac{1}{\nu_0}$.

Thanks to Equation (2.12), the average value of $v(t)$ can be rewritten as:

$$\langle v(t)\rangle_{t_k,\tau} := \frac{1}{\tau}\int_{t_k}^{t_k+\tau} [\nu_0 + \Delta v(\theta)]d\theta := \nu_0 + \frac{1}{\tau}\int_{t_k}^{t_k+\tau} \Delta v(\theta)d\theta. \tag{2.15}$$

One can therefore deduce the average frequency fluctuations during the time interval τ:

$$\langle \Delta v(t)\rangle_{t_k,\tau} := \int_{t_k}^{t_k+\tau} \Delta v(\theta)d\theta = \langle v(t)\rangle_{t_k,\tau} - \nu_0. \tag{2.16}$$

In terms of number of clock cycles, it can be written as:

$$\langle \Delta v(t)\rangle_{t_k,\tau} = \frac{n_k}{\tau} - \nu_0. \tag{2.17}$$

Equation (2.17) thus provides an experimental measure of the frequency fluctuations of an oscillator. However, it is more common to express these fluctuations as [131]:

$$y(t) := \frac{\Delta v(t)}{\nu_0} = \frac{\mathrm{d}x(t)}{\mathrm{d}t}. \tag{2.18}$$

Equation (2.18) defines fractional frequency fluctuations which are basically the normalized version of frequency fluctuations. This quantity being dimensionless has the advantage to remain unchanged under frequency multiplications and divisions. Moreover, it allows easier comparisons among oscillators having different nominal frequencies.

2.2.1.3 Average fractional frequency

Similarly to instantaneous frequency $v(t)$, the fractional frequency fluctuations are not observable [134]. Following the same procedure as with $v(t)$, one can have access to its average value over a time interval τ starting at time t_k:

$$\overline{y}_k := \langle y(t)\rangle_{t_k,\tau} = \frac{1}{\tau}\int_{t_k}^{t_k+\tau} y(\theta)d\theta. \tag{2.19}$$

The quantity defined in Equation (2.19) is known as the average fractional frequency during the k-th measurement interval [131]. Based on Equations (2.13) and (2.18), the average fractional frequency can be expressed as:

$$\overline{y}_k = \frac{\varphi(t_k+\tau)-\varphi(t_k)}{2\pi\nu_0\tau} = \frac{x(t_k+\tau)-x(t_k)}{\tau}, \tag{2.20}$$

[2] Since n_k is the number of periods of the oscillator signal during the time interval τ, it is reasonable to consider τ to be greater than the average period $\frac{1}{\nu_0}$ of the signal. Indeed, if it was not the case, it would be impossible to count the number of periods, which would be of no interest.

where the numerator represents phase error accumulated from instant t_k to instant[3] $t_k+\tau$. Thus, the knowledge of the phase error at both ends of the time interval is sufficient to compute the corresponding average fractional frequency. Table 2.1 provides a summary of the relationships between phase and frequency fluctuations.

Phase		Frequency
$\varphi(t)$ [rad]	$\xrightarrow{\frac{1}{2\pi}\frac{d}{dt}}$	$\nu(t)-\nu_0$ [Hz]
$\times\frac{1}{2\pi\nu_0}\downarrow$		$\downarrow\times\frac{1}{\nu_0}$
$x(t)$ [s]	$\xrightarrow{\frac{d}{dt}}$	$y(t)$ [no dimension]

Table 2.1: Relationship between phase and frequency fluctuations (should be read from left to right, and from top to bottom)

The quantity $\overline{y}_k$ can be easily related to experimental results given by counting techniques. Indeed, thanks to Equation (2.14), one can write:

$$\langle \nu(t) \rangle_{t_k,\tau} = \nu_0 \left(1+\overline{y}_k\right) = \frac{n_k}{\tau}, \tag{2.21}$$

which yields:

$$\overline{y}_k = \frac{n_k}{\tau\nu_0} - 1. \tag{2.22}$$

To conclude, the study of frequency (or phase) fluctuations of an oscillator can be conducted using a dataset of M consecutive average fractional frequency measurements $\overline{y}_k$. More precisely, one individual measurement of duration τ provides one sample $\overline{y}_k$. Repeated measurements of a large number M of $\overline{y}_k$ are necessary for statistical treatment that yields meaningful measure of instability over τ.

2.2.1.4 Limitations of the model

As it is always the case, models described in the above sections do not include all phenomena that occur in a real-world oscillator. This is made to avoid dealing with too complex models that would have been not usable in practice. Similarly, assumptions are used to focus, among others, to specificities of interest. Other assumptions may also be used, even though they are not met in the real-world.

[3] Here, we are in the general case as described in Appendix B. However, in the remainder of this thesis we will assume there is no dead time in samples acquisition.

A typical example of such assumptions is the fact that phase fluctuations $\varphi(t)$ are stationary. This assumption is very useful for characterizing random processes, but considerations such as the lifetime of the oscillator show that real-world signals are not necessarily stationary [135]. One therefore needs to specify under which circumstances such assumptions are valid. In regards to the oscillator's lifetime, it is shown that as long as the observation time is negligible with respect to the oscillator's lifetime, one can assume stationarity of $\varphi(t)$. Caution should however be taken while making this assumption, to avoid conflict with other properties or physical considerations of the model [134]. In the context of random number generation, τ is usually of the order of milliseconds, while low frequency phenomena such as temperature, power supply noise take longer to change. They are thus very little integrated during τ. In addition, the differential principle makes it possible to reduce their effects. It is therefore realistic to consider phase fluctuations as stationary random processes.

On the other hand, a random process, even though stationary, may not be differentiable. Based on Equation (2.11), this fact implies that the instantaneous frequency is not defined whenever phase fluctuations are not differentiable. Such situations occur when phase fluctuations include step functions, or when they are modeled as ideal white noise. However, Vernotte imputes this limitations to the model, not the real-world signal [135]. Indeed, the limited bandwidth of any system yields infinitely differentiable processes. Therefore, one can always assume that the instantaneous frequency exists for any oscillator.

2.2.1.5 Autorrelation function

Because of its theoretical nature, a probabilistic model cannot be applied to practical cases dealing with random processes. However, a partial statistical description, in terms of average values, may provide an acceptable substitute for the probabilistic description of the phenomenon. The concept of autocorrelation appears to be a cornerstone in this approach [123, Page 209].

The autocorrelation function estimates how similar $x(t_1)$ and $x(t_2)$ are. It therefore denotes how the shape of process x evolves from t_1 to t_2. Thanks to the stationarity of the phase fluctuations, its autocorrelation function is defined as [123, Chapter 6]:

$$R_x(\tau) := \langle x(t_1) \cdot x(t_1+\tau) \rangle , \tag{2.23}$$

where $\tau := t_2 - t_1$ is the time difference between t_1 and t_2. Because statistics of the process do not change over time, one usually expresses the autocorrelation function as:

$$R_x(\tau) = \langle x(t) \cdot x(t+\tau) \rangle . \tag{2.24}$$

This means that the time origin for computing the autocorrelation function of $x(t)$ does not matter. Hence, one can instead use $t' = t - \tau$ as the time origin, yielding :

$$R_x(\tau) = \langle x(t') \cdot x(t'+\tau) \rangle = \langle x(t-\tau) \cdot x(t) \rangle = R_x(-\tau). \tag{2.25}$$

This shows that the autocorrelation function of a stationary process is an even function.

As mentioned in Section 2.1.1, when dealing with random processes, any ensemble statistic has its time version. This holds for the autocorrelation too for which the time version is defined as [123, Equation 6.3]:

$$\mathscr{R}_x(\tau) := \overline{x(t) \cdot x(t+\tau)} = \lim_{T \to +\infty} \frac{1}{2T} \int_{-T}^{T} x(t)x(t+\tau)dt. \tag{2.26}$$

In general, ensemble and time autocorrelation functions are not equal. However, in the special case of ergodic processes we are dealing with, we have [123, Equation 6.4] :

$$R_x(\tau) = \mathscr{R}_x(\tau) = \lim_{T \to +\infty} \frac{1}{2T} \int_{-T}^{T} x(t)x(t+\tau)dt, \tag{2.27}$$

allowing to evaluate autocorrelation of phase fluctuations from a sample function $x(t)$.

Note that phase fluctuations of the oscillator's output signal is a zero mean process, therefore

$$R_x(0) = \lim_{T \to +\infty} \frac{1}{2T} \int_{-T}^{T} |x(t)|^2 dt \tag{2.28}$$

is the variance (also called energy) of the process.

2.2.2 Characterizing noise in frequency domain

Quantities defined in Section 2.2.1 are all time dependent. This time dependence of the quantities justifies why random processes are usually characterized in time domain. However, it is also possible to characterize them in the Fourier frequency domain. This kind of characterization is complementary to the time domain characterization of random processes. Under the assumption that phase fluctuations are stationary, either representation can be deduced from the other through a Fourier transform. On the other hand, Fourier transform of the above-defined quantities cannot be taken as they are. The first reason is because Fourier transform of a random process does not exist, unless it is considered in the distribution sense [135]. The second reason is that the Fourier transform of a random process is also a random process, and therefore does not bring more insight on the process.

More to the point, random phase (or frequency) fluctuations have theoretically infinite energy, due to the stationarity assumption. Therefore other tools must be introduced, that would enable another way to analyze these random processes.

2.2.2.1 Power spectral density

In the study of deterministic signals, it appears more convenient to deal with them in the frequency domain [136]. Indeed, Laplace and Fourier transforms make computations easier. For instance, convolution in the time domain becomes a multiplication in the frequency domain [137, Section 9.5]. For this reason, one may be tempted to compute Fourier transform of the phase (or frequency) fluctuations in order to use tools developed for the analysis of deterministic signals in the frequency domain.

However, random signals we are dealing with are stationary by assumption. This implies that statistics of the phase fluctuations do not change over time. Thus, for any sample function $x(t)$, the integral

$$\int_{-\infty}^{+\infty} |x(t)|\, dt$$

is infinite, questioning the existence of a Fourier transform [123, Section 7.1]. To work around this, some adjustments are required, yielding the concept of power spectral density.

The problem of existence of the Fourier transform comes from the fact that one computes it over an infinite duration. However, in practical situations, measurements are done during a finite time interval $2T$. This allows to define a truncated version $x_T(t)$ of the sample function $x(t)$ as:

$$x_T(t) = \begin{cases} x(t) & \text{if } |t| \leqslant T \\ 0 & \text{if } |t| > T \end{cases}, \tag{2.29}$$

which is square integrable as T is finite, provided that the process has a finite mean-square value [123, Section 7.2]. The function $x_T(t)$ therefore admits a Fourier Transform X_T given by:

$$X_T(\omega) := \int_{-\infty}^{+\infty} x_T(t) e^{-i\omega t} dt = \int_{-T}^{T} x(t) e^{-i\omega t} dt, \tag{2.30}$$

where $\omega = 2\pi f$ is the radian frequency and f the (Fourier) frequency. From Parseval's theorem, we have [138, Theorem 7.4.2]:

$$\int_{-\infty}^{+\infty} |x_T(t)|^2\, dt = \frac{1}{2\pi} \int_{-\infty}^{+\infty} |X_T(\omega)|^2\, d\omega \tag{2.31}$$

which implies:

$$\frac{1}{2T} \int_{-\infty}^{+\infty} |x_T(t)|^2\, dt = \frac{1}{2\pi} \int_{-\infty}^{+\infty} \frac{|X_T(\omega)|^2}{2T} d\omega. \tag{2.32}$$

It should be remembered that $\{x(t)\}$ is a random process, therefore $\{x_T(t)\}$ and $\{X_T(t)\}$ are also random processes. It is therefore possible to take the ensemble average of both sides to get:

$$\left\langle \frac{1}{2T} \int_{-\infty}^{+\infty} |x_T(t)|^2\, dt \right\rangle = \left\langle \frac{1}{2\pi} \int_{-\infty}^{+\infty} \frac{|X_T(\omega)|^2}{2T} d\omega \right\rangle. \tag{2.33}$$

The ensemble average is actually the probabilistic expectation and can be defined as an integral. The above equation therefore displays an equality between two double integrals. The ensemble average and the integral can therefore be interchanged thanks to Fubini-Toneli's theorem [137, Theorem 1.4.1]. It then follows:

$$\frac{1}{2T}\int_{-\infty}^{+\infty}\left\langle |x_T(t)|^2\right\rangle dt \;=\; \frac{1}{2\pi}\int_{-\infty}^{+\infty}\left\langle \frac{|X_T(\omega)|^2}{2T}\right\rangle d\omega \;=\; \frac{1}{2\pi}\int_{-\infty}^{+\infty}\frac{\left\langle |X_T(\omega)|^2\right\rangle}{2T}d\omega. \tag{2.34}$$

Since $\{x(t)\}$ is stationary, the quantity $\left\langle |x_T(t)|^2\right\rangle$ does not change over time. Therefore, its time average is still equal to itself and one has:

$$\left\langle |x_T(t)|^2\right\rangle \;=\; \frac{1}{2T}\int_{-\infty}^{+\infty}\left\langle |x_T(t)|^2\right\rangle dt \;=\; \frac{1}{2\pi}\int_{-\infty}^{+\infty}\frac{\left\langle |X_T(\omega)|^2\right\rangle}{2T}d\omega. \tag{2.35}$$

Moreover, thanks to the stationarity of $\{x(t)\}$, we have:

$$\lim_{T\to+\infty}\left\langle |x_T(t)|^2\right\rangle \;=\; \left\langle |x_T(t)|^2\right\rangle. \tag{2.36}$$

It then follows that $\lim\limits_{T\to+\infty}\displaystyle\int_{-\infty}^{+\infty}\frac{\left\langle |X_T(\omega)|^2\right\rangle}{2T}d\omega$ exists and is finite, and thus:

$$\lim_{T\to+\infty}\int_{-\infty}^{+\infty}\left\langle \frac{|X_T(\omega)|^2}{2T}\right\rangle d\omega \;=\; \int_{-\infty}^{+\infty}\lim_{T\to+\infty}\frac{\left\langle |X_T(\omega)|^2\right\rangle}{2T}d\omega, \tag{2.37}$$

which allows to write:

$$\left\langle |x_T(t)|^2\right\rangle \;=\; \frac{1}{2\pi}\int_{-\infty}^{+\infty}\lim_{T\to+\infty}\frac{\left\langle |X_T(\omega)|^2\right\rangle}{2T}d\omega. \tag{2.38}$$

The integrand of the right-hand side is called the two-sided power spectral density of x, thus:

$$S_x^{TS}(\omega) \;:=\; \lim_{T\to+\infty}\frac{\left\langle |X_T(\omega)|^2\right\rangle}{2T}. \tag{2.39}$$

It is defined for both positive and negative values of ω. Note however that $\omega \longmapsto |X_T(\omega)|^2$ is an even function and so:

$$\int_{-\infty}^{+\infty}|X_T(\omega)|^2 d\omega \;=\; 2\int_{0}^{+\infty}|X_T(\omega)|^2 d\omega. \tag{2.40}$$

This yields to the unilateral (also called one-sided) power spectral density function defined as:

$$S_x(\omega) \;:=\; \lim_{T\to+\infty}\frac{\left\langle |X_T(\omega)|^2\right\rangle}{T}, \tag{2.41}$$

where only positive frequencies are considered. In practice, we do not have negative frequencies, we will therefore use the one-sided power spectral density for jitter assessment, knowing that it is related to the two-sided power spectral density as follows:

$$S_x(\omega) \;=\; \begin{cases} 2S_x^{TS}(\omega) & \text{for } \omega \geqslant 0 \\ 0 & \text{otherwise} \end{cases}. \tag{2.42}$$

Note that the above discussion reveals the fact that two-sided power spectral density is a real, positive and even function of ω.

Power spectral density as a function of f It is possible to express X_T as a function of f rather than ω. In this case, x_T and X_T are related as:

$$X_T(f) := \int_{-\infty}^{+\infty} x_T(t) e^{-i2\pi f t} dt, \tag{2.43}$$

and

$$x_T(t) := \int_{-\infty}^{+\infty} X_T(f) e^{i2\pi f t} df. \tag{2.44}$$

Power spectral density can therefore be expressed as:

$$S_x(f) := \lim_{T \to +\infty} \frac{\langle |X_T(f)|^2 \rangle}{T}. \tag{2.45}$$

Power spectral density in the Laplace domain Power spectral density has been expressed as a function of radian frequency ω. However, for system analysis, the process needs to be characterized in the Laplace domain, since transfer functions are usually expressed as functions of the Laplace variable $s := \sigma + i\omega$. This can be achieved by replacing ω by $-is$ in the expression of $S_x(\omega)$, with the assumption $\sigma = 0$. The power spectral density will then be denoted[4] $S_x(s)$.

2.2.2.2 Wiener-Khinchin theorem

In most practical cases, the formulas provided in Section 2.2.2.1 are not convenient for evaluating the power spectral density of the random process $\{x(t)\}$. This section aims at providing a more usable formula for determining the power spectral density.

Thanks to Equation (2.30), one can write[5]:

$$|X_T(\omega)|^2 = X_T(\omega) X_T^*(\omega) = \int_{-T}^{T} \int_{-T}^{T} x_T(t_1) x_T(t_2) e^{-i\omega(t_2 - t_1)} dt_1 dt_2. \tag{2.46}$$

Let $\tau = t_2 - t_1$, with fixed t_1, then $d\tau = dt_2$, and one has:

$$|X_T(\omega)|^2 = \int_{-T}^{T} \int_{-T}^{T} x_T(t_1) x_T(t_1 + \tau) e^{-i\omega\tau} dt_1 d\tau. \tag{2.47}$$

It then follows:

$$\begin{aligned} \langle |X_T(\omega)|^2 \rangle &= \left\langle \int_{-T}^{T} \int_{-T}^{T} x_T(t_1) x_T(t_1 + \tau) e^{-i\omega\tau} dt_1 d\tau \right\rangle \\ &= \int_{-T}^{T} \int_{-T}^{T} \langle x_T(t_1) x_T(t_1 + \tau) \rangle e^{-i\omega\tau} dt_1 d\tau \\ &= \int_{-T}^{T} \int_{-T}^{T} \langle x_T(t) x_T(t + \tau) \rangle e^{-i\omega\tau} dt d\tau, \end{aligned} \tag{2.48}$$

[4] Since Laplace domain contains Fourier domain, the spectral density in Laplace domain results in the analytic continuation of the spectral density defined in the Fourier domain. These two functions are therefore not equal, even though they may be deduced from one another. They will however be denoted by the same letter, the argument will help make the difference between them.

[5] $X_T^*(\omega)$ denotes the complex conjugate of $X_T^*(\omega)$, and satisfies:

$$X_T^*(\omega) = X_T(-\omega).$$

because statistics of the process do not depend on the time, due to its stationarity. The power spectral density is therefore computed as :

$$\begin{aligned} S_x(\omega) &= \lim_{T\to+\infty} \frac{1}{2T} \int_{-T}^{T}\int_{-T}^{T} \langle x_T(t)x_T(t+\tau)\rangle\, e^{-i\omega\tau} dt d\tau \\ &= \int_{-\infty}^{+\infty} \left(\lim_{T\to+\infty} \frac{1}{2T} \int_{-T}^{T} R_{x_T}(\tau) dt \right) e^{-i\omega\tau} d\tau \\ &= \int_{-\infty}^{+\infty} R_{x_T}(\tau) e^{-i\omega\tau} d\tau, \end{aligned} \tag{2.49}$$

because $R_{x_T}(\tau)$ does not depend on t. Hence, the power spectral density is expressed as:

$$S_x(\omega) = \int_{-\infty}^{+\infty} R_x(\tau) e^{-i\omega\tau} d\tau. \tag{2.50}$$

Equation (2.50) expresses that the power spectral density $S_x(\omega)$ and the autocorrelation function $R_x(\tau)$ form a Fourier transform pair. This result is known as the Wiener-Khinchin theorem and is only valid for stationary random processes [126, Section 1.5.2].

In the analysis of random signals, this result provides a link between the time domain (correlation function) and the frequency domain (power spectral density) representations. This allows us to assess electronic noises in the frequency domain and deduce their time domain characteristics as:

$$R_x(\tau) = \frac{1}{2\pi} \int_{-\infty}^{+\infty} S_x(\omega) e^{i\omega\tau} d\omega. \tag{2.51}$$

2.2.2.3 Relationships between power spectral densities

Recall that random processes we are dealing with are:

- phase fluctuations $\varphi(t)$ which represent effects of electronic noises on the output phase of the oscillator,
- time error $x(t)$ which basically is a normalized version of phase fluctuations,
- fractional frequency $y(t)$ which represents fluctuations of the output frequency due to noises.

Because each of these processes represent the same phenomenon, their respective power spectral density functions are related. We establish here relationships between these various quantities.

Thanks to ergodicity of these various processes and Wiener-Khinchin theorem, we can write:

$$S_\varphi(f) = \int_{-\infty}^{+\infty} \left(\lim_{T\to+\infty} \frac{1}{2T} \int_{-T}^{T} \varphi(t)\varphi(t+\tau) dt \right) e^{-i2\pi f\tau} d\tau. \tag{2.52}$$

Equation (2.10) helps to related $S_\varphi(f)$ and $S_x(f)$. Indeed:

$$S_x(f) = \int_{-\infty}^{+\infty} \left(\lim_{T\to+\infty} \frac{1}{2T} \int_{-T}^{T} x(t)x(t+\tau)dt \right) e^{-i2\pi f\tau} d\tau \tag{2.53}$$

$$= \frac{1}{(2\pi\nu_0)^2} \int_{-\infty}^{+\infty} \left(\lim_{T\to+\infty} \frac{1}{2T} \int_{-T}^{T} \varphi(t)\varphi(t+\tau)dt \right) e^{-i2\pi f\tau} d\tau \tag{2.54}$$

$$= \frac{1}{(2\pi\nu_0)^2} S_\varphi(f). \tag{2.55}$$

Note that the autocorrelation of x, just like for any random process, can be written as:

$$R_x(\tau) = \left(x \star x^*\right)(\tau), \tag{2.56}$$

where $\star$ is the convolution operator, and x^* is the complex conjugate of x. Since Fourier transform converts convolution into multiplication, it follows that:

$$S_x(f) = \mathscr{F}[x] \cdot \mathscr{F}[x]^*(f). \tag{2.57}$$

This latter is of great help to express the power spectral density of y. Indeed, Equation (2.18) expresses $y(t)$ as the time derivative of $x(t)$. Thus, thanks to the Wiener-Khinchin theorem:

$$S_y(f) = \mathscr{F}[y] \cdot \mathscr{F}[y]^*(f) = 4\pi^2 f^2 \mathscr{F}[x] \cdot \mathscr{F}[x]^*(f) = 4\pi^2 f^2 S_x(f), \tag{2.58}$$

yielding:

$$S_y(f) = \frac{f^2}{\nu_0^2} S_\varphi(f) = 4\pi^2 f^2 S_x(f). \tag{2.59}$$

2.2.3 Noise models

The main interest of power spectral densities comes from the fact that they are theoretically deterministic. That is, any type of noise will always display the same kind of behavior in the frequency domain. Any type noise can therefore be uniquely characterized by its power spectral density. This has great impact on the study of jitter components. Most common types of noise encountered in nature, and especially in electronics, can be modeled either as white noise or power law noises that will be detailed next.

2.2.3.1 White noise

The concept of white noise is of crucial importance in the generation of true random numbers. It is the random process used by existing TRNG models to estimate the entropy of the source of randomness. A perfect comprehension of this process is therefore of highest importance to evaluate physical noise sources.

White noise is defined as a wide-sense stationary, zero-mean and uncorrelated random process [139, Section 3.2]. Hence, the random process $\{w(t)\} = \{w_t\}$ is said to be a white noise process if there is a nonnegative real number S_0 such that the following equations hold:

$$\forall t, \qquad \langle w_t \rangle = 0, \tag{2.60}$$

$$\forall t, \qquad \langle w_t^2 \rangle = S_0, \tag{2.61}$$

$$\forall t, \tau, \quad \langle w_t w_\tau \rangle = S_0 \delta(\tau). \tag{2.62}$$

It is usually characterized as the time derivative of a Wiener process [140, Section 17.2].

The uncorrelatedness of white noise realizations means that there is no coupling between different instants regardless how close they are. This property implies that white noise is completely unpredictable, which explains why it is such a preferable choice for generating random numbers. It is however a purely theoretical concept which is physically unrealizable. Indeed, since white noise is stationary, Wiener-Khinchin theorem can be applied to compute its power spectral density function. From Equation (2.62) and the fact that Fourier transform of a Dirac Delta function is the constant function 1 [138, Example 7.3.9], it follows that:

$$S_w(f) = S_0. \tag{2.63}$$

Equation (2.63), characteristic of white noise processes, means that it has the same power over all frequencies. A straightforward corollary is that white noise has infinite power, which is not possible since no real-world system or phenomenon has infinite power. As a complementary information, the fact that white noise has equal power over all frequencies explains the origin of its name. Indeed, it comes from the analogy with white light which contains an equal mixture of all visible frequencies of light [141, Section 2.9].

However, various random phenomena can be considered as approximation of white noise provided they display a constant spectrum within the bandwidth of the device of interest. Indeed, there is no electronic system which has an infinite bandwidth, it is therefore impossible in practice to access the whole spectrum of a phenomenon. So, once the spectrum of that phenomenon is constant over the observable range of frequencies, one can always assume it is the case outside the bandwidth. Furthermore, computations will be made within the bandwidth of the system, avoiding infinite power [123, Section 7.7].

Note that uncorrelatedness of the noise realizations does not imply their independence. Hence, realizations of the white noise process are not necessarily independent, unless their are jointly normally distributed [102, Theorem 4.5.1]. Such processes are called independent white noise

processes [139, Section 3.2]. In practical cases, one often deals with processes for which realizations are normally distributed. Such processes, like the white noise we are dealing with in our study, are called Gaussian processes. This has the important consequence to imply independence of realizations of Gaussian white noise [142, Section 3.2]. A common example of Gaussian white noise in electronic devices is the thermal noise, which originates from random motion of electrons within the substrate [142, Example 3.7].

2.2.3.2 Power law noise

Electronic noises are not always uncorrelated. Correlated ones are often referred to as colored noises, in opposition to the white noise. Despite thorough investigations of these various types of noises, they are still not fully understood [143, 144, 145, 146, 147, 148] [126, Chapter 9] and their various origins remain unclear. These various noises can be grouped in several classes of noise types[6] based on the shape of their power spectral density. It is admitted that most electronic noises can be grouped in five different noise types which can be characterized by a power law model [149]. Hence, the power spectral density $S_y(f)$ of frequency fluctuations of a real-world oscillator can be written as the sum of the power spectral densities of the five independent noise types as:

$$S_y(f) = \sum_{\alpha=-2}^{\alpha=2} h_\alpha f^\alpha, \tag{2.64}$$

where α is an integer characterizing the noise type. It is possible to find phenomena that yield exponents below $\alpha = -2$ and above $\alpha = 2$. For example, some deterministic phenomena, such as drifts, can introduce different dependencies of $S_y(f)$ ($\alpha = -3$ or $\alpha = -4$). However, the differential principle reduces the impact of the latter types of noise. One can therefore ignore phenomena yielding exponents outside the range $[\![-2,2]\!]$. Moreover, these various noise types are assumed to be Gaussian due to their random nature [135]. Note that the choice for α being an integer depends on this model.

The interest of the power law model, in addition to its simplicity, lies in the fact that each noise type corresponds to a specific physical origin summarized in Table 2.2. However, as mentioned earlier, links between these noises and their cause are not well understood. Some explanations are available for some of these noises, but they only cover a specific aspect of the noise type explained. The case of odd exponents is the most tricky one, since they cannot simply be explained using analog integration or derivation components of oscillators.
This power law noise model was the basis of several algorithms to simulate the different noises as will we see in the next.

[6] *e.g.* white noise is a class of noise which contains among others thermal and shot noises [133, Section 2.1].

α	Noise type	Origin
-2	Random Walk Frequency	environment
-1	Flicker Frequency	resonator
0	White Frequency	white noise internal to oscillator loop
1	Flicker Phase	noisy electronics
2	White Phase	white noise external to oscillator loop

Table 2.2: Origins of the power law noises (summarized from [134]).

2.2.3.3 Noise simulation

It is of great value to have means of generating simulated power law noise having the desired noise type. Applications of such means are numerous, for example they can serve as a simulation tool, or even as a way to validate noise identification techniques. Several algorithms have been developed for this purpose, some simulating white noise [150], some simulating flicker noise [151], while others are of general purpose and try to simulate all the various noise profiles [152, 149]. We adopted the algorithm developed by Kasdin and Walter [149], since it provides a common basis for the generation of the various types of noises, and that it is among noise simulation methods recommended by the NIST time and frequency division [132].

The basic idea of this algorithm is to consider a noise type as the response of a linear time-invariant (LTI) system to white noise (Appendix A). Indeed, simulation of white noise is straightforward. Thanks to its definition, one can simply generate a sequence of independent and identically distributed random values. This can actually be done using any common programming language. For example, to generate 1000 samples of a Gaussian white noise data with standard deviation 0.1, a `Python` syntax goes as follows:

```
import numpy as np
x = np.random.normal(scale=0.1, size=1000)
```

Listing 2.1: Generation of white noise using Python

Thus, to generate the samples of a noise type having a power spectral density of $h_\alpha f^\alpha$, the Kasdin and Walter algorithm is supplied with the value of α, corresponding to the type of noise desired at the output, and a white noise of variance Q^d, for which the samples are distant from τ_0. The whole algorithm consists in finding the appropriate impulse response function to have the desired noise type as output. This is done on the basis of the relation [149, Equation (39)]:

$$Q^d = \frac{h_\alpha}{2(2\pi)^\alpha \tau_0^{\alpha-1}}. \tag{2.65}$$

A `Python` implementation of this algorithm has been made available by Wallin[7]. The use of this

[7] https://github.com/aewallin/allantools.

algorithm therefore allows to simulate each of the five common noise types as shown in Figure 2.4.

2.3 Jitter analysis tools

In order to assess the jitter, we need analysis tools that are compatible with jitter measurement methods. Since this assessment is mainly done through statistical tools such as variance or autocorrelation, measurement methods basically consist in sampling the random signal with a sampling frequency $1/\tau$. Samples of the random signal are then used to compute variance and autocorrelation to deduce the properties of the jitter. The square-root of the variance, known as the standard deviation, is usually preferred for characterizing the jitter.

However, it appears that the phase fluctuations of an oscillator can also be described using its frequency domain representation, namely its power spectral density. In the convenient case of stationary processes, the Wiener-Khinchin theorem draws a bridge between these two representations. Specifically, for a lag 0 in Equation (2.50), we can write [153, Section 3]:

$$I^2(\tau) = \int_0^{+\infty} S_y(f)|H_\tau(f)|^2 df. \tag{2.66}$$

where $I(\tau)$ is the true variance and H_τ is the transfer function of the variance operator.

2.3.1 Limitation of the classical variance

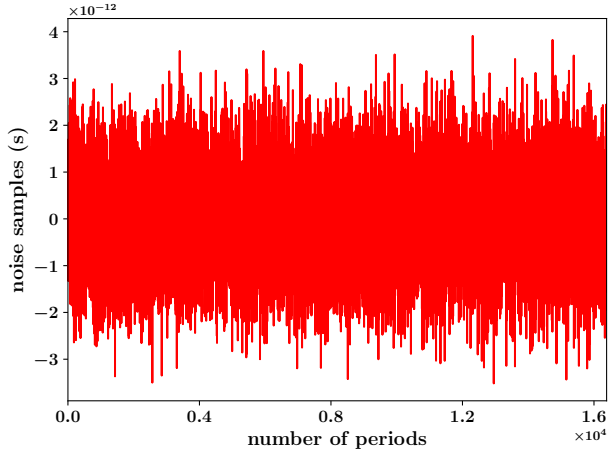
The true variance is actually a theoretical measure. One usually estimate this variance through various means using the available dataset. The transfer function of the variance operator is the Fourier transform of the impulse response function, which indicates how samples are considered to estimate the variance. The most common estimate used is the (unbiased) sample variance defined as:

$$\sigma_y^2 := \frac{1}{n_y - 1} \sum_{i=1}^{n_y} \left(\overline{y}_i - \frac{1}{n_y} \sum_{j=1}^{n_y} \overline{y}_j \right)^2, \tag{2.67}$$

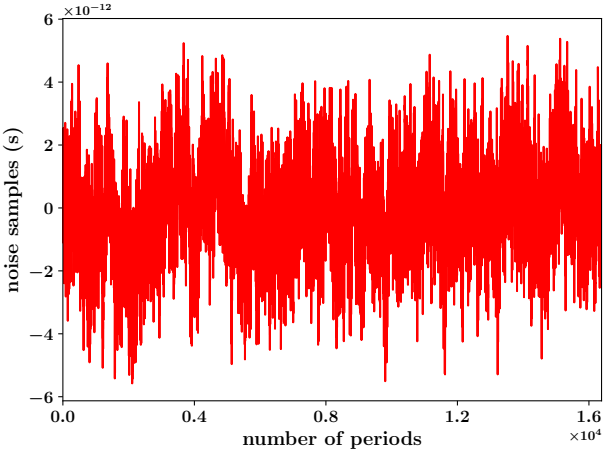
where $n_y := n_k - 1$ is the number of average fractional frequency time series, and n_k is the number of periods of the oscillator that occurred during the observation time τ. Equation (2.67) shows that frequency data are used in a row to estimate the variance. This fact is illustrated by the plot of the impulse response in Figure 2.5.

Hence, the impulse response of the classical variance is a rectangular function. Computing its Fourier transform yields H_τ given by:

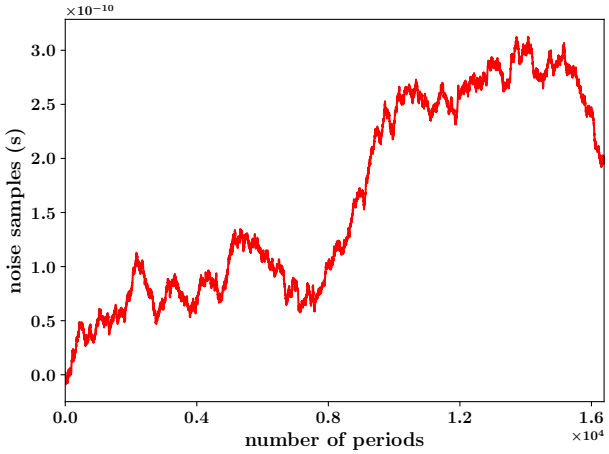
$$H_\tau(f) = \frac{\sin(\pi\tau f)}{\pi\tau f}. \tag{2.68}$$



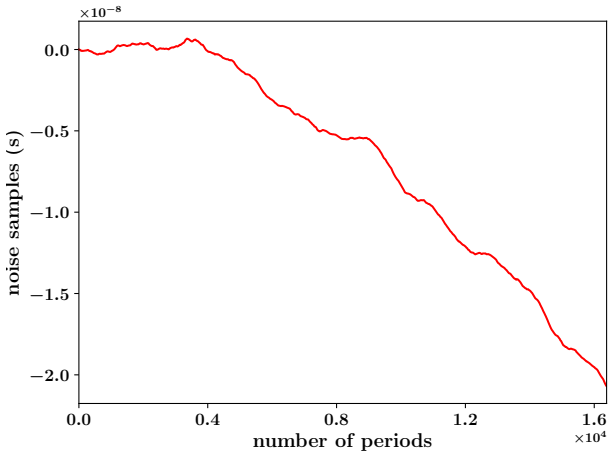
(a) white phase ($\alpha = 2$).



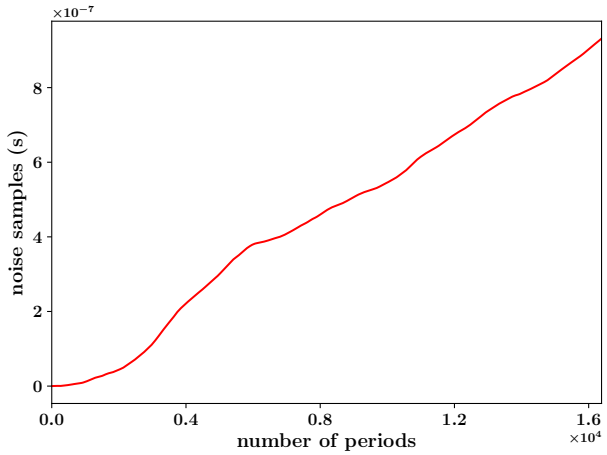
(b) flicker phase ($\alpha = 1$).



(c) white frequency or random walk phase ($\alpha = 0$).



(d) flicker frequency or flicker walk phase ($\alpha = -1$).



(e) random walk frequency ($\alpha = -2$).

Figure 2.4: Different types of noise generated using Kasdin and Walter algorithm (sampling period: $\tau_0 = 8 \cdot 10^{-9}$ s, sample size: 2^{14}, variance of input noise: $Q^d = 10^{-24}$ s).

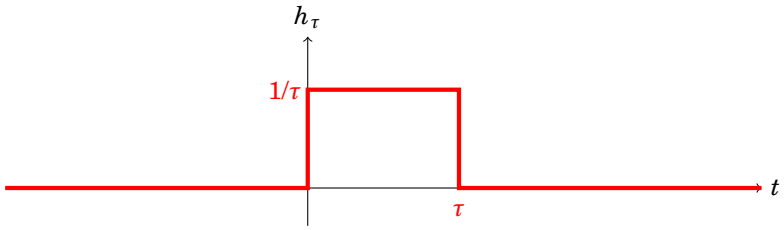


Figure 2.5: Impulse response of the classical variance.

Under the assumption that noises affecting the phase of an oscillator are all described by a power law model, the power spectral density of the phase fluctuations satisfies:

$$S_y(f) = \sum_{\alpha=-2}^{2} h_\alpha f^\alpha, \tag{2.69}$$

where each α represents the noise profile and each h_α represents the corresponding noise level. The variance of the fluctuations can therefore be expressed as:

$$I^2(\tau) = \sum_{\alpha=-2}^{2} \frac{h_\alpha}{(\pi\tau)^2} \int_0^{+\infty} f^{\alpha-2} \sin^2(\pi\tau f) df. \tag{2.70}$$

Convergence at $+\infty$ In any real physical system, there is always a cutoff frequency f_c above which the energy of the signal is reduced to 0 [154]. This guarantees that the integral upper bound is finite, and thus implies its convergence as f approaches $+\infty$.

Convergence at 0 As f approaches 0, one can write $\sin(\pi\tau f) \sim \pi\tau f$, thus:

$$\int_0^{+\infty} \sin^2(\pi\tau f) f^{\alpha-2} df \sim \pi^2\tau^2 \int_0^{+\infty} f^\alpha df \tag{2.71}$$

converges only for $\alpha > -1$ from the Riemann criterion for improper integrals. Thereby, the previous integral does not converge at 0, for $\alpha = -1$ and $\alpha = -2$. So when low-frequency noises such as flicker frequency ($\alpha = -1$) and random walk noise frequency ($\alpha = -2$) are present in the random signal, the classical variance no longer gives stable results [155]. This instability of the results is even greater the longer the observation time τ is. However, as we saw in Section 1.1.2, it is essential to have a large enough observation time to accumulate entropy that guarantee the unpredictability of generated numbers. Thus, the use of classical variance poses a dilemma:

- not to accumulate entropy in order to have the most stable results possible, with a high risk of generating numbers that are not sufficiently random for cryptographic purposes;
- accumulate entropy in order to gain more security, with the certainty of having results that are not interpretable because they are not stable.

Since these types of noises are present in most electronic devices [156], it is therefore necessary to have another variance for which convergence is ensured for $\alpha = -1$ and $\alpha = -2$.

2.3.2 Allan variance

The limitation of the classical variance was first noticed by Barnes [157]. Due to the importance of having a convergent variance estimator, Allan developed and proposed a new type of variance named after him [154]. It has become the most common time domain measure of frequency stability. Being introduced to replace the classical variance, it is a measure of the fractional frequency fluctuations. Unlike the classical variance, it converges for most types of noise.

2.3.2.1 Description of the Allan variance

The Allan variance, as originally defined can be computed using:

$$\sigma_y^2(\tau) = \frac{1}{2}\left\langle \left(\overline{y}_{i+1} - \overline{y}_i\right)^2 \right\rangle, \tag{2.72}$$

where τ is the averaging time. In practice, this quantity is computed as [132, Equation (6)]:

$$\sigma_y^2(\tau) = \frac{1}{2(n_y - 1)} \sum_{i=1}^{n_y - 1} \left(\overline{y}_{i+1} - \overline{y}_i\right)^2. \tag{2.73}$$

Note that Allan variance can also be expressed using phase data. Indeed, thanks to Equation (2.20), one can write:

$$\sigma_y^2(\tau) = \frac{1}{2\tau^2} \left\langle (x_{i+2} - 2x_{i+1} + x_i)^2 \right\rangle. \tag{2.74}$$

An estimate of the Allan variance using phase fluctuations is therefore computed as [132, Equation (7)]:

$$\sigma_y^2(\tau) = \frac{1}{2(n_k - 2)\tau^2} \sum_{i=1}^{n_k - 2} (x_{i+2} - 2x_{i+1} + x_i)^2. \tag{2.75}$$

Results are usually expressed as square roots $\left(\sqrt{\sigma_y^2(\tau)} = \sigma_y(\tau)\right)$ called Allan deviations.

Convergence of the Allan variance The variance of Allan was introduced in order to have a convergent measure, even in the presence of low frequency noises. Equation 2.72 indicates that Allan variance is computed using differences of consecutive samples. This situation yields an impulse response illustrated in Figure 2.6.

The transfer function corresponding to this impulse response is defined as [132, Section 5.27]:

$$H_\tau(f) = \frac{\sqrt{2}\sin^2 \pi\tau f}{\pi\tau f}. \tag{2.76}$$

The signal's variance is therefore :

$$I^2(\tau) = \sum_{\alpha=-2}^{2} \frac{2h_\alpha}{(\pi\tau)^2} \int_0^{+\infty} \sin^4(\pi\tau f) f^{\alpha-2} df. \tag{2.77}$$

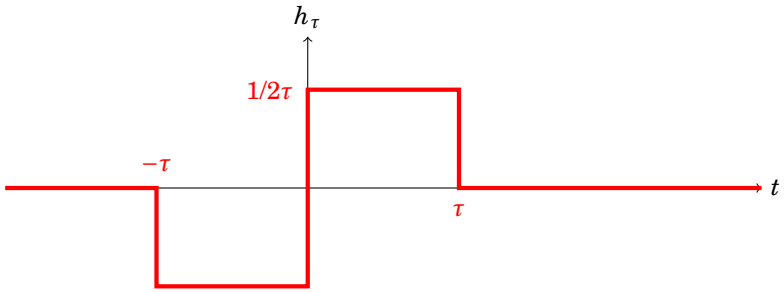


Figure 2.6: Impulse response of the Allan variance.

For the same reasons given in Section 2.3.1, the integral $\int_0^{+\infty} \sin^4(\pi\tau f) f^{\alpha-2} df$ converges at ∞ for any value of α. Let us focus on the convergence as f approaches 0. From $\sin(\pi\tau f) \sim \pi\tau f$, we have:

$$\int_0^{+\infty} \sin^4(\pi\tau f) f^{\alpha-2} df \sim \pi^4\tau^4 \int_0^{+\infty} f^{2+\alpha} df \tag{2.78}$$

Riemann criterion for improper integrals therefore ensures that the above integral converges for $\alpha > -3$. This guarantees that the Allan variance is a stable measure of frequency stability even in presence of low frequency noises. However, it has a poor confidence interval which affects its accuracy [131].

2.3.2.2 Overlapped Allan variance

The confidence interval of the Allan variance can be improved by increasing the number of samples used for the variance computation. However, it is, in practice, not possible to infinitely increase the sample size. Nevertheless, it is possible to circumvent this issue through the use of overlapping samples. In this case, computations are performed by utilizing all possible combinations of the data set, as depicted in Figure 2.7. The use of overlapping samples has the advantage to increase the sample size, thus improve the confidence of the resulting estimate. However, it also increases computational complexity. Due to its better confidence interval, the overlapped Allan variance is usually preferred to its original simple version [132, Section 5.2].

Let τ_0 be the sampling period of the series (x_i). We consider the averaging time τ as an integer multiple of τ_0, thus there exists $m \in \mathbb{N}$ such that $\tau = m\tau_0$. The integer m is called averaging factor, and in practice, the value of m can be arbitrarily chosen among whole numbers less than $\frac{n_k-1}{2}$. The basic idea is to divide the set of x_i's into clusters of finite time duration, in a way that time stride between each consecutive clusters is always equal to τ_0 [158]. This forms all overlapping sample clusters with the averaging time τ, and containing m samples each as shown in Figure 2.7.

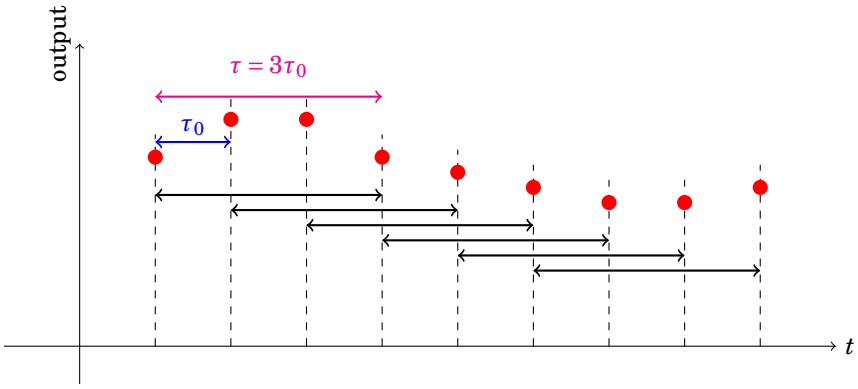


Figure 2.7: Overlapping samples for $m = 3$ (m: number of strides).

This makes a maximum use of the available dataset by forming all possible overlapping samples at each averaging time τ. The overlapped Allan variance can be estimated from a set of n_y frequency measurements for averaging time $\tau = m\tau_0$, by the expression [132, Equation 10]:

$$\sigma_y^2(\tau) := \frac{1}{2m^2(n_y - 2m + 1)} \sum_{i=1}^{n_y-2m+1} \left(\sum_{k=i}^{i+m-1} (\overline{y}_{k+m} - \overline{y}_k) \right)^2 . \tag{2.79}$$

Since phase and frequency data are related, it is possible to express the overlapped Allan variance in terms of phase data [132, Equation 11]:

$$\sigma_y^2(\tau) = \frac{1}{2\tau^2(n_k - 2m)} \sum_{k=0}^{N-2n-1} \Big((x_{k+2n} - 2x_{k+n} + x_k)^2 \Big). \tag{2.80}$$

In most cases, when referring to Allan variance, it is the overlapped version which is computed. This preference comes from the fact that it improves the confidence interval. But it also does not introduce bias in computations, even though samples are not independent [159].

2.3.2.3 Application to noise identification

Assessing the clock jitter of an oscillator comprises several aspects, such as the identification various noise types that affect the jitter. This step is of crucial importance, since it allows to either validate or reject assumptions used in the entropy estimation models. Therefore consequences of a wrong identification can be as severe the reduction of security in cryptographic constructions. In addition to converging in the presence of low-frequency noise, the Allan variance also happens to identify the dominant noise type profile in the signal.

Indeed, Equation (2.72) explicitly shows that Allan variance is a function of the observation time τ. Since these various noise types behave differently from one another based on the value of observation time[8] τ, and that Allan variance depends on τ, it can be expected that Allan variance

[8] High frequency noises will be dominant over low frequency ones as τ goes to 0, while low frequency noises will be dominant for large values of τ.

also behaves differently depending on the nature of the dominant noise. In order to exhibit this behavior of the Allan variance, one needs to expand Equation (2.77) and obtain [160]:

$$\sigma_y^2(\tau) = h_2\frac{3f_c}{4\pi^2}\tau^{-2} + h_1\frac{1.038+3\ln(2\pi f_c\tau)}{4\pi^2}\tau^{-2} + h_0\frac{1}{2}\tau^{-1} + 2h_{-1}\ln 2 + h_{-2}\frac{2\pi^2}{3}\tau. \tag{2.81}$$

Equation (2.81) helps to consider Allan variance as a sum of terms $A_\mu\tau^\mu$. Each value of μ, *i.e* term of this function is a characteristic of a specific noise type, yielding the possibility to identify the noise profile which affects the phase of the oscillator. Figure 2.8 depicts the response of Allan deviation response to various noise types.

Dashed lines correspond to the theoretical responses while dotted lines represent responses to simulated noises. One can therefore identify the noise types affecting phase fluctuations data based on the Allan deviation response. We can see that the Allan deviation has a $\tau^{-1/2}$ dependence for white frequency noise (see Figure 2.8c), a τ^0 dependence for flicker frequency noise (see Figure 2.8d), and a $\tau^{1/2}$ dependence for random walk frequency noise (see Figure 2.8e). However, Allan deviation is not able to discriminate white phase noise from flicker phase noise as illustrated in Figures 2.8a and 2.8b. This is what actually motivated the development of the modified Allan variance that will be discussed next.

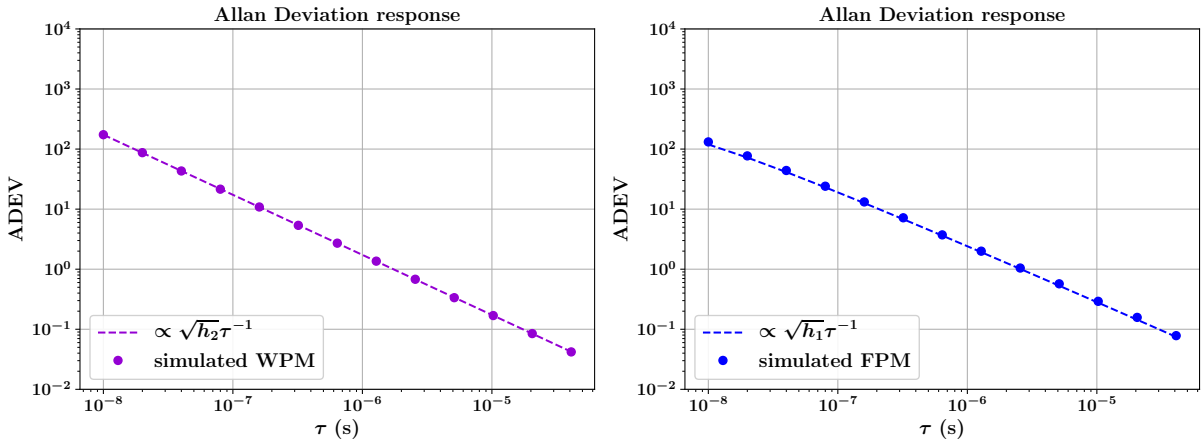
2.3.3 Modified and time versions of the Allan variance

The Allan variance has two major advantages: the convergence of the measurement in the presence of low-frequency noises, and the structural description of the signal over time. The second advantage helps to identify of the dominant noise type present in the signal as a function of the accumulation time τ. However, this method based on Allan variance has difficulties to discriminate between flicker phase noise and white phase noise. Since jitter is an instability of the signal phase, the white phase noise is the one of interest. Therefore, we need a metric that can reveal the presence of the white phase noise, which is precisely why the modified Allan variance was introduced [114].

2.3.3.1 Modified Allan variance

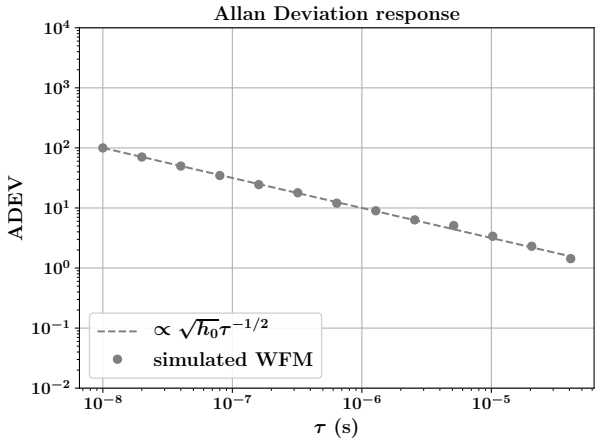
Therein, we still assume that the averaging time τ is an integer multiple of the sampling period τ_0. It then follows that there exists an averaging factor $m \in \mathbb{N}$ such that $\tau = m\tau_0$. From a set of n_y frequency measurements, the modified Allan variance can be estimated as [132, Equation 13]:

$$\mathrm{Mod}\,\sigma_y^2(\tau) := \frac{1}{2m^4(n_y-3m+2)}\sum_{i=1}^{n_y-3m+2}\left[\sum_{j=i}^{i+m-1}\left(\sum_{k=j}^{j+m-1}\left(\overline{y}_{k+m}-\overline{y}_k\right)\right)\right]^2. \tag{2.82}$$

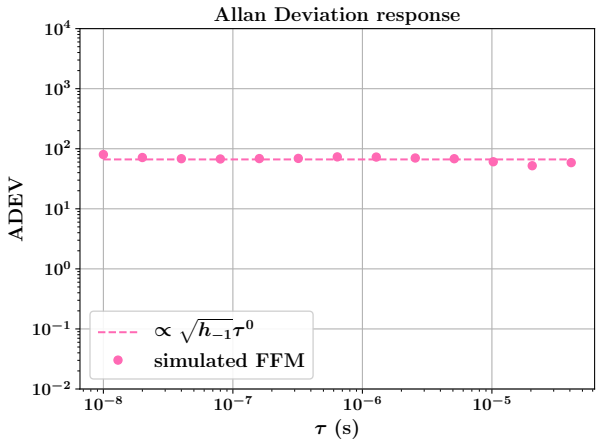


(a) white phase ($\alpha = 2$).

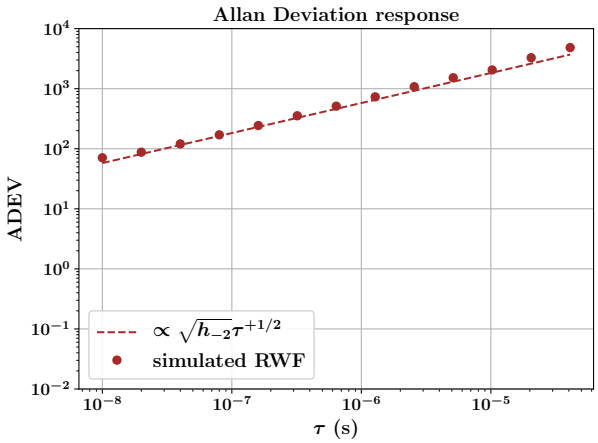
(b) flicker phase ($\alpha = 1$).



(c) white frequency or random walk phase ($\alpha = 0$).



(d) flicker frequency or flicker walk phase ($\alpha = -1$).



(e) random walk frequency ($\alpha = -2$).

Figure 2.8: Allan deviation response to various noise types (dashed lines are theoretical responses and dots are simulated ones, ADEV: Allan deviation).

In terms of phase data, the modified Allan variance can be estimated as [132, Equation 14]:

$$\mathrm{Mod}\,\sigma_y^2(\tau) := \frac{1}{2m^2\tau^2(n_k-3m+1)} \sum_{i=1}^{n_k-3m+1} \left[\sum_{j=i}^{i+m-1} \left(x_{j+2m} - 2x_{j+m} + x_j\right) \right]^2. \tag{2.83}$$

The modified Allan variance response to phase fluctuations due to the various noise types can be expressed [160]:

$$\mathrm{Mod}\,\sigma_y^2(\tau) = h_2\frac{3}{8\pi^2}\tau^{-3} + h_1\frac{24\ln 2 - 9\ln 3}{8\pi^2}\tau^{-2} + h_0\frac{1}{4}\tau^{-1} + h_{-1}\frac{27\ln 3 - 32\ln 2}{8} + h_{-2}\frac{11\pi^2}{20}\tau, \tag{2.84}$$

yielding Figure 2.9.

We see that, as in the case of the Allan deviation, the modified Allan deviation response has a $\tau^{-1/2}$ dependence for white frequency noise (see Figure 2.9c), a τ^0 dependence for flicker frequency noise (see Figure 2.9d), and a $\tau^{1/2}$ dependence for random walk frequency noise (see Figure 2.9e). In contrast to the Allan deviation, the modified Allan deviation has a $\tau^{-3/2}$ dependence for white phase noise (see Figure 2.9a), and a τ^{-1} dependence for flicker phase noise (see Figure 2.9b). This makes it possible to unambiguously distinguish between white phase noise and flicker phase noise.

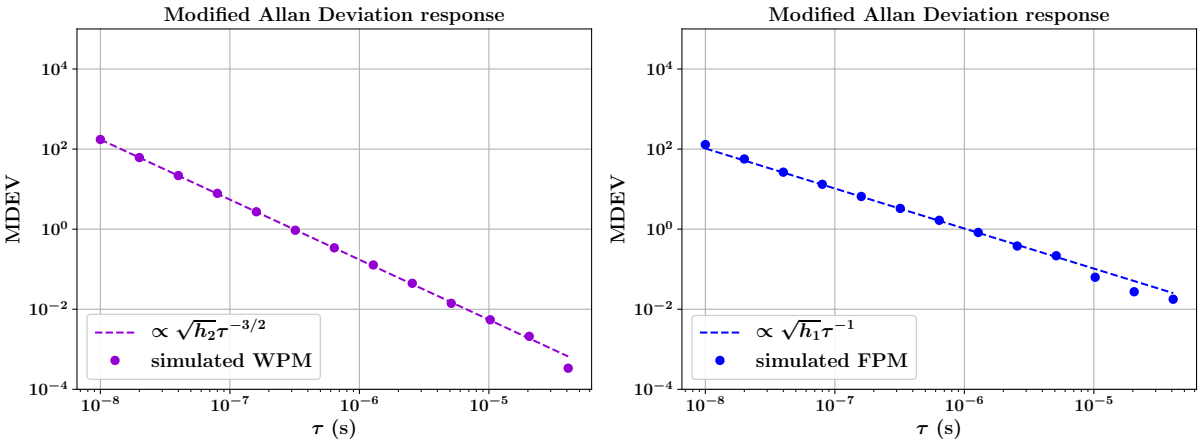
This method is actually the most commonly used in the timekeeping field [132]. However, it requires to output data and process them externally. Moreover, being a graphical method, it cannot be implemented in hardware. For online tests of TRNGs, it is necessary to have some means to automatically identify the power law noise type. One of these means could be the autocorrelation function.

2.3.3.2 Time Allan variance

The Allan variance serves as a measure of frequency stability. However, jitter is the time instability of the signal. It is therefore needed to have a measure of the time stability of the oscillator's signal. This is where comes into play the time Allan variance defined [132, Equation 15]:

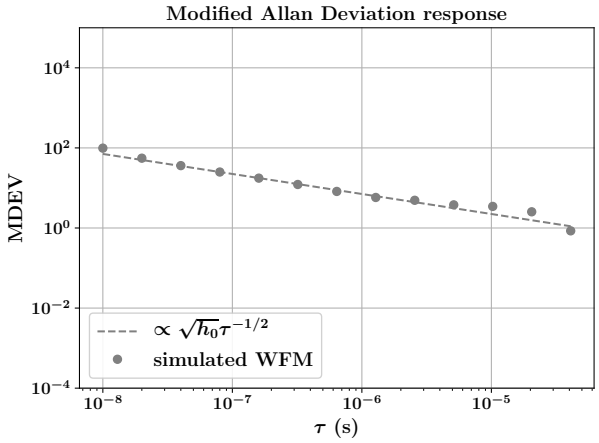
$$\sigma_x^2(\tau) = \frac{\tau^2}{3}\mathrm{Mod}\,\sigma_y^2(\tau). \tag{2.85}$$

The time Allan variance (simply called time variance and denoted TVAR) corresponds to the time stability of phase x over the observation interval τ. It is therefore the preferred measure to estimate the jitter. Equation (2.85) shows that the time variance is actually a normalized version of the modified Allan variance. This normalization makes TVAR equal to the classical variance when the fluctuations in x are random and uncorrelated [132, Section 5.2.6]. Note that the square root of the time variance is often called time deviation.

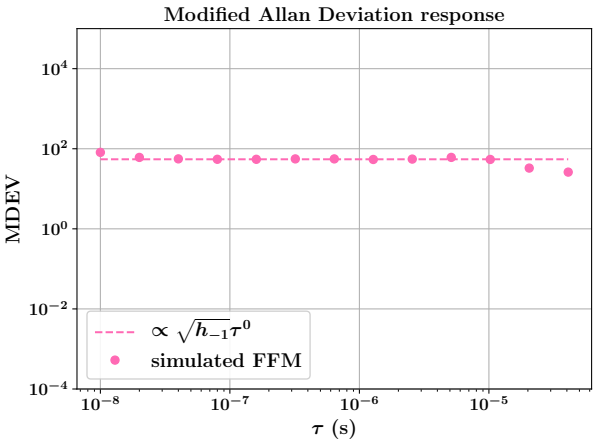


(a) white phase ($\alpha = 2$).

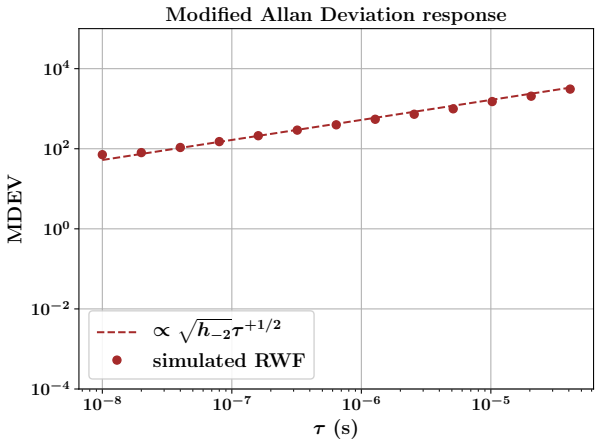
(b) flicker phase ($\alpha = 1$).



(c) white frequency or random walk phase ($\alpha = 0$).



(d) flicker frequency or flicker walk phase ($\alpha = -1$).



(e) random walk frequency ($\alpha = -2$).

Figure 2.9: Modified Allan deviation response to various noise types (dashed lines are theoretical responses and dots are simulated ones, MDEV: modified Allan deviation).

2.3.4 Noise identification using autocorrelation function

The autocorrelation function was already introduced in Section 2.2.1.5. It is a fundamental tool to describe a time series. In this way, it evaluates the degree by which its value at one time is similar to its value at a certain later time. For example, one knows that white noise is such that its value at a specific time is uncorrelated with its value at any other delayed time. This intuition reveals the fact that autocorrelation can be used to identify various noise types.

Indeed, when dealing with frequency data, its appears that autocorrelation can actually be used to identify power law noises. The method is based on the properties of discrete-time fractionally integrated noises having spectral densities of the form $\frac{1}{(2\sin\pi f)^{2\delta}}$. When δ satisfies $\delta < 1/2$, the noise process is stationary and has a lag 1 autocorrelation equal to $r_1 = \frac{\delta}{1-\delta}$. Note that r_1 stands for the value of the lag 1 autocorrelation of the noise process. Thus, its type can be estimated from $\delta = \frac{r_1}{1+r_1}$. Under these circumstances, white phase noise has $r_1 = -1/2$, flicker phase noise has $r_1 = -1/3$, and white frequency noise has $r_1 = 0$. A more general study of this subject, which includes more noise types like flicker frequency or random walk frequency has been made by Riley [132, Section 5.5.5].

This method has been simulated to confirm results. As one knows, the autocorrelation is a theoretical tool that can only be estimated from real data. For this reason, one needs to perform several trials and then compute the average value of r_1. Moreover, for better confidence, it is recommended to perform as many trials as possible. We have therefore repeated the following process 10^4 times:

- generate phase samples of a given noise profile,
- convert phase samples into frequency data,
- compute r_1 on frequency data.

For each trial, we generated 2^{14} phase samples for the considered noise type, yielding $2^{14} - 1$ frequency data.

Table 2.3 presents results of our simulations over one thousand trials for each noise type. Figures are rounded to 3 decimal places, and as one can notice, the mean value of r_1 equals its theoretical value. This gives confidence on the fact that lag 1 autocorrelation can be used to identify noise types we are interested in, namely white phase and flicker whenever they are the dominant noise types. Since Riley ensures acceptable results for at least 34 data points [132, Section 5.5.6], we

can have confidence in the results presented in Table 2.3.

Noise type	α	Expected r_1	Minimum r_1	Maximum r_1	Average r_1	Std r_1
White Phase	2	$-1/2$	-0.591	-0.413	-0.499	0.022
Flicker Phase	1	$-1/3$	-0.426	-0.215	-0.333	0.026
White Frequency	0	0	-0.134	0.106	-0.001	0.032

Table 2.3: Lag 1 autocorrelation of some noise types (Std r_1: standard deviation computed on values of r_1).

Since these different types of noise coexist in the real world, it is more realistic to consider a situation in which different types of noise would impact the phase data. As an example, we can assume that the phase data are impacted by thermal noise and flicker noise. Using the Kasdin and Walter simulator, it is possible to adjust the proportion of each type of noise by acting on the respective variances of white phase noise and flicker phase noise. This allowed us to calculate the lag 1 autoccorelation of the total noise impacting phase data for each proportion of the flicker noise (ranging from 0 to 100%) as depicted in Figure 2.10.

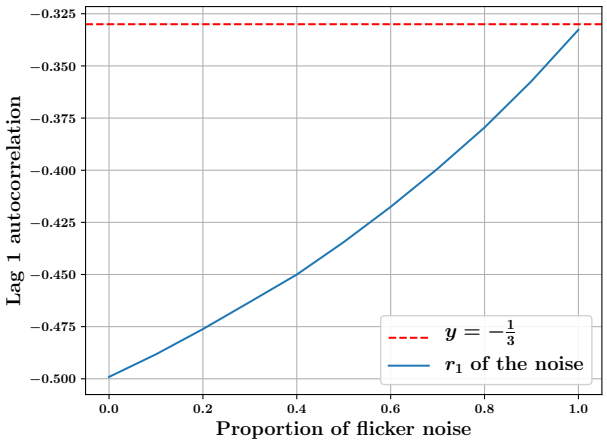


Figure 2.10: Evolution of the lag 1 autoccorelation as a function of the proportion of flicker noise (proportions given in the range $[0,1]$, r_1: lag 1 autoccorelation).

Knowing the value of the lag 1 autocorrelation, Figure 2.10 allows us to make reasonable assumptions concerning the components of the noise that cause phase fluctuations. Thus, thanks to Allan variance and lag 1 autocorrelation function, we are able to identify the dominant noise type. From this knowledge, one should be able to estimate the thermal noise level and thus its proportion in the jitter.

2.4 Jitter measurement method

2.4.1 Counter based method for jitter measurement

In order to assess phase jitter of oscillator signals, Allan variance and its variants focus on frequency fluctuations. This strategy can be applied in the field of random number generation. Indeed, it is possible to design measurement methods that provide frequency data. In hardware, TRNGs are usually made of two oscillators respectively outputting signals s_1 and s_2 (see Section 1.1.2). As already explained, both signals are jittered, however, to simplify the computation, it is common practice to include the jitter of the reference signal in that of the other signal and consider one of the clock signals as jitter-free [95].

In most cases, s_2 is considered as the reference signal and is used to sample s_1 at rising edges. This method has the disadvantage of being sensitive to the duty cycle of the sampled signal. The impact of this dependence is the introduction of some bias if the duty cycle of s_1 is not 50%, which is the case in general. It is however possible to remove this dependency by using frequency data instead of time data. This can be done by counting the number of periods of s_1 within a time interval τ defined as being a constant number of periods of s_2 as depicted in Figure 2.11. From the number N of the periods of s_1, we can deduce its frequency as $\frac{N}{\tau}$. Because s_1 and s_2 are not ideal, the number N is likely to change from one measurement to another. As expressed in Equation (2.14), this change of value is related to the frequency fluctuations which can be linked to the jitter. Furthermore, this jitter measurement method happens to be the only one that can be embedded to date [161].

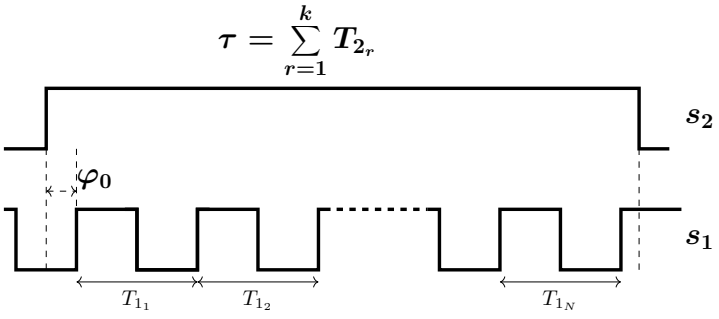


Figure 2.11: Timings in counting the periods of signal s_1.

In order to provide a link between the values of the counter and the jitter, we use the fact that s_1 is jitter-free and that its jitter is included in that of the signal s_2. Consequently, the period T_2 of

signal s_2 can be considered as a random variable of standard deviation (see [95, Appendix C]):

$$\sigma_{eq} \simeq \sqrt{\sigma_2^2 + \frac{T_2}{T_1}\sigma_1^2}, \tag{2.86}$$

and the period T_1 of signal s_1 as a constant. The measurement time $\tau = \sum_{r=1}^{k} T_{2_r}$ is thus a random variable. This time defines only the duration of the time period, not the initial phase φ_0 of the signal s_1 when the counting starts (see Figure 2.11). However, to measure the jitter more accurately, the initial phase φ_0 must be taken into account. This initial phase is independent of τ, since its value does not depend on τ.

Because T_1 is constant, the counter value N is a random variable defined as:

$$N := \max\left\{k \in \mathbb{N},\ \varphi_0 + \sum_{r=1}^{k} T_{1_r} \leqslant \tau\right\} = \max\left\{k \in \mathbb{N},\ \varphi_0 + kT_1 \leqslant \tau\right\}. \tag{2.87}$$

The value N thus satisfies the inequality:

$$\varphi_0 + N \times T_1 \leqslant \tau < \varphi_0 + (N+1) \times T_1, \tag{2.88}$$

which is equivalent to:

$$N \leqslant \frac{\tau - \varphi_0}{T_1} < N + 1. \tag{2.89}$$

It then follows that N can be written as:

$$N = \left\lfloor \frac{\tau - \varphi_0}{T_1} \right\rfloor. \tag{2.90}$$

It thus exists $0 \leqslant \varepsilon < 1$ such that:

$$N = \frac{\tau - \varphi_0}{T_1} - \varepsilon \tag{2.91}$$

According to Sheppard's correction [162], ε is a random variable that is uniformly distributed over $[0,1)$. Since ε is independent of $\frac{\tau-\varphi_0}{T_1}$ and that properties of the classical variance can the extended to the Allan variance (see Appendix C), the following equations hold:

$$\mathrm{avar}(N) = \mathrm{avar}\left(\frac{\tau - \varphi_0}{T_1}\right) + \mathrm{avar}(\varepsilon) = \frac{\mathrm{avar}(\tau) + \mathrm{avar}(\varphi_0)}{T_1^2} + \frac{1}{12}, \tag{2.92}$$

where avar stands for the Allan variance operator.

It is important to note that the Allan variance of counter values always overestimates the Allan variance of the jitter per unit of time (e.g the signal period). The correction must be applied by subtracting $\mathrm{avar}(\varepsilon) = \frac{1}{12}$ and $\frac{\mathrm{avar}(\varphi_0)}{T_1^2}$.

As $\frac{\text{avar}(\varphi_0)}{T_1^2} \in \left[0, \frac{1}{12}\right]$ (the maximum is obtained if φ_0 is uniformly distributed over $[0, T_1)$), according to Equation (2.92):

$$\text{avar}(\tau) \;=\; T_1^2\,\text{avar}(N) - \frac{T_1^2}{12} - \text{avar}(\varphi_0) \;\geqslant\; T_1^2\,\text{avar}(N) - \frac{T_1^2}{12} - \frac{T_1^2}{12}. \tag{2.93}$$

As we do not want to overestimate the jitter, a conservative approach is to take the minimum value for $\text{avar}(\tau)$ that is:

$$\text{avar}(\tau) \;=\; T_1^2 \cdot \text{avar}(N) - \frac{T_1^2}{6}. \tag{2.94}$$

Using Equation (2.94), the variance of the accumulated jitter can be computed from the variance of counter values. Hence, it is possible to use counter values to assess the oscillator phase jitter.

2.4.2 Jitter measurement in hardware

Since AIS-31 recommends that the source of randomness is monitored continuously using dedicated embedded test(s), it is important that the variance measurement method can be implemented in hardware. The adopted method should require minimum area and power resources. In our experiments, we compared the area requirements of the Allan variance (based on differences of counter values) with that of two existing methods respectively proposed by Haddad *et al.* [163], and Fischer and Lubicz [76]. For consistency in comparison results, they were implemented in the same device, an Intel Cyclone V FPGA.

As one can see in Figure 2.12, the circuitry corresponding to hardware implementation of the Allan variance requires:

- one multiplier to square data;
- one subtracter to compute the difference of the consecutive samples;
- one adder, with associated register, used as accumulator.

The circuitry corresponding to hardware the implementation of the variance computation used by Haddad *et al.*, as depicted in Figure 2.13, requires:

- two multipliers (one of 12 bits and the other of 24 bits) to square data,
- one subtracter,
- two adders and associated registers (one of 24 bits and the other of 12 bits) as accumulators,
- four additional data registers to store intermediate data.

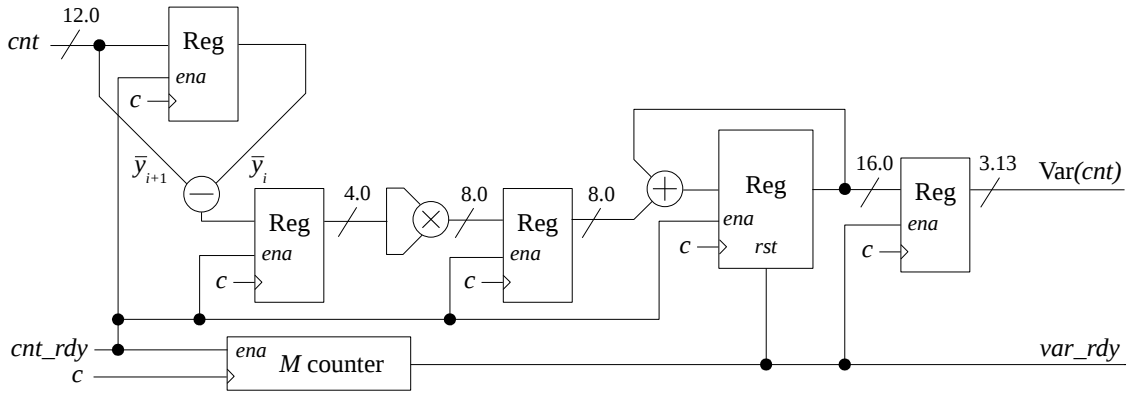


Figure 2.12: Allan variance measurement circuitry based on Equation (2.73) (Numbers before and after the radix point indicate the number of bits of the integer and fractional part of the given value, respectively).

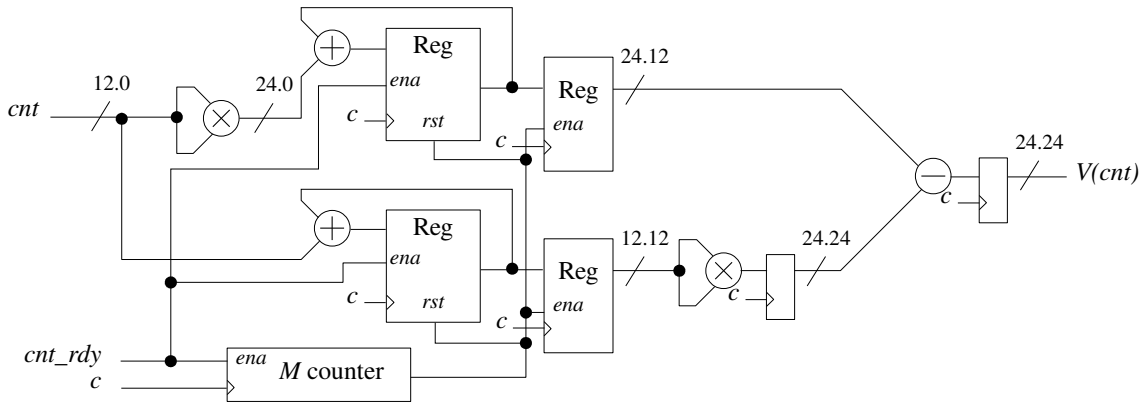


Figure 2.13: Implementation of the counter variance measurement circuitry for the method proposed by Haddad *et al.* in [163] (Numbers before and after the radix point indicate the number of bits of the integer and fractional part of the given value, respectively).

The circuitry corresponding to the hardware implementation of the variance computation of the third method is similar to that presented by Fischer and Lubicz [76, Figure 6].

After implementing these three variance measurement methods, we compared them by evaluating their respective design parameters: area, speed and power consumption. Area and speed values were obtained from `Quartus II, version 16.1`, while the power consumption was measured using a dedicated hardware evaluation platform [164].

Method	Area		f_{max}	Power
	ALM/Regs	DSPs	[MHz]	[mW]
Proposed method, Equation (2.73)	49/117	1	238.5	4 – 5
Haddad *et al.* [163]	119/160	2	178.3	6 – 7
Fischer and Lubicz [76]	169/200	4	187.7	7 – 8

Table 2.4: Summary of implementation results of the variance measurement method based on counter differences (Allan variance) compared to other state-of-the-art methods implemented in the dedicated evaluation board featuring Intel Cyclone V FPGA device 5CEBA4F17C8N.

From this evaluation, presented in Table 2.4, one can deduce that the Allan variance measurement circuitry is smaller, faster and consumes slightly less power than the circuitry required by the other two methods. This can be explained by the fact that the Allan variance computes the variance of the successive differences, which form a time series with zero mean. Thus, the Allan variance eliminates the need to compute the mean as opposed to other methods. It therefore requires less resources (only one subtracter, one adder and one DSP block used) than others.

Experimental results presented here confirm the advantages of Allan variance over classical variance for characterization of counter value distribution. To increase precision the overlapped Allan variance should be used. However, to characterize the jitter distribution (and not the counter values), the above study should be made with the time Allan variance, yielding a more complicated circuit than that of Figure 2.12. This will be made in the future.

2.5 Estimation of the thermal noise contribution

Knowing that existing TRNG models use the thermal noise as the only accepted source of entropy, it is of crucial importance to know the proportion of thermal noise contained in the phase jitter. This step aims at preventing any overestimation of the entropy, thus preventing any security flaw in TRNG used in cryptographic constructions. Thermal noise being white phase noise [133, Chapter 2], its level in the signal is given by its power spectral density coefficient h_2. Finding the value of h_2 will therefore yield the knowledge of the thermal noise proportion in the jitter.

Since the phase jitter is the one considered for the generation of random numbers, the use of the Allan time variance is more relevant than the Allan variance. From Equations (2.85) and (2.84), one can write:

$$\sigma_x^2(\tau) = h_2 \frac{1}{8\pi^2}\tau^{-1} + h_1 \frac{8\ln 2 - 3\ln 3}{8\pi^2} + h_0 \frac{1}{12}\tau + h_{-1}\frac{27\ln 3 - 32\ln 2}{24}\tau^2 + h_{-2}\frac{11\pi^2}{60}\tau^3. \tag{2.95}$$

Thus, knowing the observation time τ, it follows that the part of the variance due to the thermal

noise is:

$$\sigma_{th}^2(\tau) = h_2 \frac{1}{8\pi^2} \tau^{-1}. \tag{2.96}$$

The problem with Equation (2.96) is that h_2 is unknown in most practical cases. The value of h_2 represents the level of white noise in the source, which differs from one source to another. However, under certain assumptions, it is possible to find this value, and thus deduce the variance due to thermal noise.

Due to modern photolithography methods, it is possible to produce even smaller transistors. While this technological advance has some advantages, it has a negative effect on the generation of random numbers. Indeed, this miniaturization of transistors leads to a reduction of the level of the thermal noise. This makes it possible for flicker noise to be is easily integrated by slightly increasing the observation time as shown in Figure 2.14. As technological advance follow this trend, flicker (phase) noise will increasingly impact phase data. It therefore seems reasonable to assume for this approach that data are mainly impacted by the thermal noise and phase flicker noise.

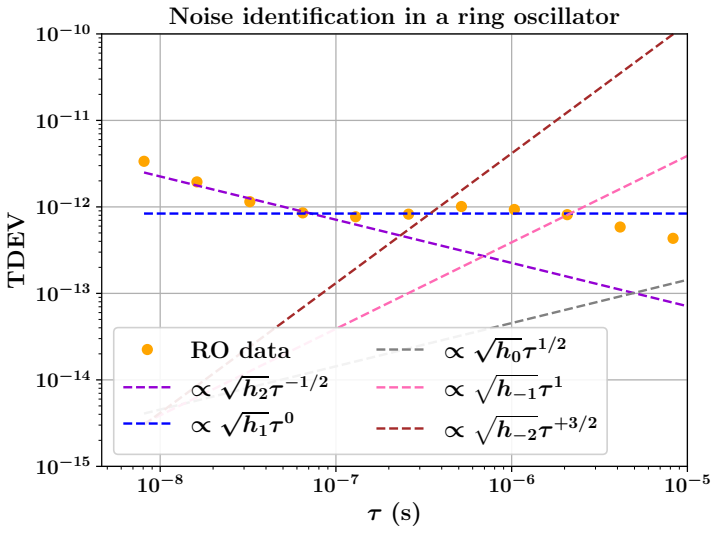


Figure 2.14: Noise identification in a ring oscillator (Obtained by applying time deviation to successive periods of the output signal of an oscillator).

In this case, Equation (2.95) becomes:

$$\sigma_x^2(\tau) = h_2 \frac{1}{8\pi^2} \tau^{-1} + h_1 \frac{8\ln 2 - 3\ln 3}{8\pi^2}, \tag{2.97}$$

where h_1 and h_2 are respective (unknown) levels of flicker and thermal noises. These values can be found using simultaneous equations. Indeed, for two different observation times τ_1 and τ_2, one

can write:

$$\begin{cases} \sigma_x^2(\tau_1) = h_2 \dfrac{1}{8\pi^2}\tau_1^{-1} + h_1 \dfrac{8\ln 2 - 3\ln 3}{8\pi^2} \\ \sigma_x^2(\tau_2) = h_2 \dfrac{1}{8\pi^2}\tau_2^{-1} + h_1 \dfrac{8\ln 2 - 3\ln 3}{8\pi^2} \end{cases} . \tag{2.98}$$

Solving these equations yields:

$$h_2 = \frac{8\pi^2\tau_1\tau_2\left(\sigma_x^2(\tau_1) - \sigma_x^2(\tau_2)\right)}{\tau_2 - \tau_1} \tag{2.99}$$

which is the level of thermal noise in the source.

In order to evaluate this method, we implemented it using Kasdin and Walter's noise generator [149]. To do this, we generated 2^{14} data samples affected by white phase noise and phase flicker noise. By its operation, the Kasdin and Walter generator takes as argument the variance Q^d of the white input noise of the LTI system, as well as the sampling period τ_0. In the simulations for which some results are presented in Table 2.5, we took $Q^d = 10^{-12}$ for both the thermal noise and the flicker noise. The sampling period was set to $\tau_0 = 8 \cdot 10^{-9}$ s. We have chosen these values for the noise generator input parameters because they lead to standard deviation values that we often find in experiments.

values of h_2		values of $\sigma_s(\tau_0)$	
found	error	found	error
$6.5357 \cdot 10^{-31}$	0.0347	$1.0172 \cdot 10^{-12}$	0.0172
$6.3614 \cdot 10^{-31}$	0.0071	$1.0035 \cdot 10^{-12}$	0.0035
$6.1955 \cdot 10^{-31}$	0.0191	$9.9037 \cdot 10^{-13}$	0.0096
$6.4449 \cdot 10^{-31}$	0.0203	$1.0101 \cdot 10^{-12}$	0.0101
$6.3425 \cdot 10^{-31}$	0.0041	$1.0020 \cdot 10^{-12}$	0.0020

(a) data affected only with the thermal noise.

values of h_2		values of $\sigma_s(\tau_0)$	
found	error	found	error
$5.5127 \cdot 10^{-31}$	0.1272	$9.3421 \cdot 10^{-13}$	0.0657
$5.7789 \cdot 10^{-31}$	0.0851	$9.5649 \cdot 10^{-13}$	0.0435
$7.5961 \cdot 10^{-31}$	0.2025	$1.0966 \cdot 10^{-12}$	0.0966
$7.0014 \cdot 10^{-31}$	0.1084	$1.0528 \cdot 10^{-12}$	0.0528
$7.9125 \cdot 10^{-31}$	0.2526	$1.1192 \cdot 10^{-12}$	0.1192

(b) data affected with thermal and flicker noises.

Table 2.5: Thermal noise contribution to the standard deviation of simulated noises (h_2: level of the thermal noise, τ_0: sampling period, mean-value of the jittered clock period).

The interest of Table 2.5a is to have an idea of the precision of the method. Indeed, if the data are only affected by the thermal noise, we could just use the classical variance. And as we can see, the standard deviation due to thermal noise seems very close to the expected value. In fact, during our various simulations, we obtained relative errors that revolve around 0.2%.

The error made when the flicker is added is greater, as shown in Table 2.5b. Indeed, when Q^d is the same for thermal and flicker noises, we obtain a relative error of about 12%. This major error is due to a very high level of the flicker noise. Because of this level, the thermal noise is practically drowned out by the flicker noise as shown in Figure 2.15, making it difficult to determine the contribution of thermal noise with high accuracy. This explanation is confirmed by Table 2.6 which shows the decrease in absolute error as the proportion of flicker noise decreases in favor of thermal noise. These results show importance of ensuring a relatively low flicker noise level in order to improve the accuracy of the jitter estimation due to thermal noise.

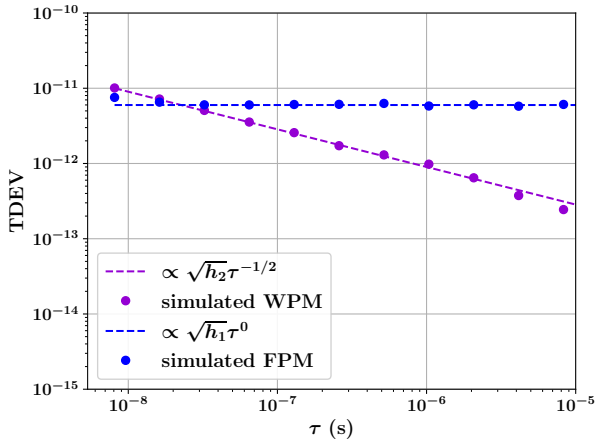


Figure 2.15: Illustration of the thermal noise drowned out by the flicker noise.

Q_1^d/Q_0^d	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
relative error	0.014	0.026	0.039	0.051	0.063	0.076	0.088	0.100	0.112	0.124

Table 2.6: Relative error as a function of the flicker noise proportion.

Since both thermal and flicker noises coexist in the data, Figure 2.15 shows the existence of a critical observation (or accumulation) time τ_c during which the jitter variances due to thermal and flicker noises are the same. From Equation (2.97), one can deduce this critical time, which is then expressed as follows:

$$\tau_c := \frac{h_2}{h_1(8\ln 2 - 3\ln 3)}. \tag{2.100}$$

Thus, for an accumulation time τ lower than τ_c, thermal noise dominates over flicker noise. It can then be assumed that data is mostly affected by thermal noise. In this case, the variance of the jitter can be assimilated to that of the thermal noise. However, for an accumulation time τ greater than τ_c, the influence of the flicker noise can no longer be ignored. An accurate estimate

of the proportion of the thermal noise in the jitter is required.

Suppose that the accumulation time τ is greater than τ_c, then the influence of the flicker noise cannot be neglected. It is therefore necessary to estimate the contribution of the thermal noise to the jitter. For this purpose, Equation (2.99) recommends calculating the Allan time variance for two different accumulation times τ_1 and τ_2. If the choice of τ_1 and τ_2 can be made arbitrarily, it is preferable to choose them in such a way as to reduce as much as possible the error made in estimating the thermal noise level. Intuitively, the best way to proceed would be to choose τ_1 and τ_2 both lower than τ_c, making sure they are as far apart as possible.

However, the different tests performed, for different values of τ_1 and τ_2 lead to errors of the same order of magnitude. This implies that the choice of positions[9] of τ_1 and τ_2 does not have a significant influence on the accuracy of the method. Thus, the error made would be inherent to the method of simulating the different types of noise. In order to verify this conjecture, we have carried out a curve fitting from the simulated data as shown in Figure 2.16. This latter yields errors of the same order of magnitude as the method presented in this section.

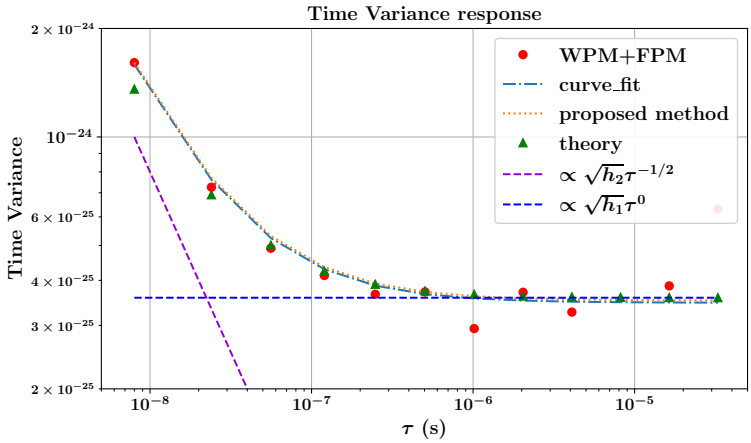


Figure 2.16: Comparison of the proposed method with a curve fitting (WPM: white phase modulation noise, FPM: flicker phase modulation noise, input white noise yielding WPM and FPM have the same variance, curve fit made using non-linear least squares).

These findings tend to confirm that an error inherent in the noise generation method influences the accuracy of the estimate of the contribution of thermal noise. This is not really surprising since the noise generator of Kasdin and Walter produces data for which the behavior approximates that of the noise being simulated, under the assumption of stationarity. However, it turns out that flicker noise is not stationary. It is therefore understandable that approximation errors

[9]The different positions of τ_1 and τ_2 are $\tau_1 < \tau_2 < \tau_c$, $\tau_1 < \tau_c < \tau_2$ and $\tau_c < \tau_1 < \tau_2$.

inherent to the method are inevitable. Error propagation can therefore explain the differences between simulated and expected noise levels.

The method presented here makes it possible to estimate the contribution of thermal noise to the variance of the jitter. This will allow for a better estimation of the entropy, and therefore prevent an overestimation of the entropy. However, there is still the problem of the error that remains to be solved. Although it is true that the noise generation method explains some of these errors, it is necessary to investigate further in order to reduce estimation errors as much as possible. A conservative approach, taking into account the maximum error of 12%, can be used.

2.6 Conclusion

In this chapter, we presented mathematical tools for jitter assessment. A good understanding of these tools is crucial since they are related to the nature of phase fluctuations of the oscillator used as source of randomness. These tools provide a mathematical framework used to model the output signal of an oscillator. From that model, we were able to characterize random fluctuations in the time domain, but also in the frequency domain. Attention has been drawn to the issues related to the widely used classical variance. Presence of low frequency noises in the oscillator signal yields a divergence of this variance, resulting in an overestimation of entropy. Allan variance, already used in the timekeeping field, was introduced as a solution to this problem. Its properties as well as its variants have proved to be of a great interest, especially when it comes to identify the dominant noise type in real-world signals, or estimating the thermal noise proportion in the jitter.

We published part of the results presented in this chapter was published in the IACR TCHES 2018[10]. In the same article, higher-order Markov chains was suggested by Skórski to model dependencies between subsequent bits of the generator. Hardware implementation and analysis were done by Oto Petura. Other results like the estimation of thermal noise proportion is still under investigation and have not been published yet. These results are used in Chapter 3 to assess the quality of the jitter used in PLL-based TRNGs.

[10] 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, 2018(3), 214-242. https://doi.org/10.13154/tches.v2018.i3.214-242

Résumé

Dans ce chapitre, nous avons présenté le cadre mathématique permettant d'étudier et de caractériser les fluctuations de phase des signaux des oscillateurs. Ces fluctuations de phases, qui sont à l'origine du jitter d'horloge, se modélisent en processus stochastiques. Dans le cadre de la génération des nombres aléatoires, ces processus ont souvent été caractérisés dans le domaine temporel par le biais de la variance. En nous basant sur des résultats de la théorie du traitement du signal, nous proposons de caractériser ces fluctuations dans le domaine fréquentiel. Cette caractérisation, complémentaire à celle faite dans le domaine temporelle, apporte de nouvelles informations concernant les phénomènes physiques en jeu, notamment les types de bruits en présence.

La variance étant la mesure communément utilisée pour évaluer le jitter, il est impératif qu'elle soit stable et précise. Cependant, il a été prouvé que la variance classique ne converge pas lorsque le jitter comporte des bruits basse fréquence, tel que le flicker. Vu que ce bruit est présent dans les fluctuations de phase d'un oscillateur, l'utilisation de la variance classique pour évaluer le jitter est fortement déconseillée. Ce problème a déjà été détecté et étudié par les spécialistes des horloges et oscillateurs ultra stables. Suite à leurs études, la variance d'Allan a été proposée comme substitut à la variance classique pour l'étude des fluctuations de fréquences. Nous avons étudié cette variance afin de voir comment l'adapter à la problématique de la génération des nombres aléatoires.

Cette étude de la variance d'Allan nous a permis de constater la stabilité, même en présence de bruits basse fréquence. Ceci en fait un candidat de choix pour mesurer le jitter lors de la génération des nombres aléatoires. De plus, sa dépendance au temps d'observation du phénomène permet de déduire une structure temporelle du phénomène. En effet, il a été illustré que la variance d'Allan, notamment sa variante modifiée permet de distinguer les différents types de bruits admettant une modélisation en loi de puissance. Ces différents atouts de la variance d'Allan ainsi que de ses variantes nous exhorte à l'adopter non seulement pour avoir une mesure du jitter, mais également pour en analyser la structure au cours du temps.

Puisque la plupart des modèles estimant l'entropie d'une source d'aléa supposent uniquement la présence du bruit thermique, il est impératif d'estimer avec précision la part du bruit thermique dans la taille du jitter. Pour cela, nous avons proposé une estimation basée sur la variance temporelle d'Allan. Puisque les simulations réalisées montrent une erreur maximale de 12%, nous recommandons d'adopter une approche conservative afin de tenir compte de l'erreur maximale. Ceci permettra de ne pas surestimer la variance, et donc l'entropie, due au bruit thermique.

Chapter 3

Phase-locked loops as sources of randomness

Contents

Phase-locked loops are electronic circuits used in logic devices mainly to generate clock signals. They have many applications such as frequency synthesis in wireless applications [165], clock and data recovery in communication systems [166], clock generation and distribution in microprocessors [167]. They are interesting for the generation of random numbers, particularly because of their robustness and ability to maintain a well-defined relationship between frequencies of their

input and output signals. In this chapter, we will examine some of the underlying properties of PLLs, with a particular focus on noise properties.

3.1 Phase-locked loops

Phase-locked loops (PLLs) belong to a larger set of regulation systems [168]. Research and design studies related to PLLs go back to the early 50s [169]. Since then, this field has gained major practical applications and PLLs are found in communication equipments such as mobile phones and computers. This section describes the fundamentals of PLLs.

3.1.1 Basic PLL overview

A phase-locked loop (PLL) is a circuit (as depicted in Figure 3.1) that uses an input signal to synchronize a signal from an embedded oscillator on it. The objective is to maintain a frequency relationship between input and output signals as:

$$f_{out} = \frac{M}{N \times C} \times f_{in}, \tag{3.1}$$

where f_{in} is the input frequency, and f_{out} the output frequency. The embedded oscillator is usually a voltage-controlled oscillator (VCO), which is synchronized with the input signal (sometimes called the reference signal).

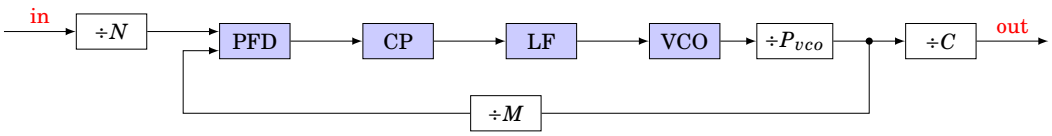


Figure 3.1: Phase-locked loop block diagram (PFD: phase-frequency detector, CP: charge pump, LF: loop filter, VCO: voltage-controlled oscillator).

In order to synchronize the VCO, the PLL has a feedback loop with a phase-frequency detector (PFD) in it. Its goal is to evaluate the phase difference between the reference signal and the feedback signal in order to produce a voltage proportional to their phase difference. This voltage, known as the error voltage, is converted into a current by the charge pump (CP). A low-pass filter (LF) then eliminates high-frequency fluctuations in order to have a clean control voltage at the input of the VCO. Thanks to this process, the frequency of the VCO will be adjusted to maintain the frequency relationship between the input and output signals of the PLL. During this operation, the PLL is said to be locked. When the VCO fails to synchronize, the PLL is unlocked.

3.1.2 Basic equations of the PLL

Equations that govern the behavior of a PLL can be established either in the time or frequency domain. The description in the frequency domain requires linear approximations which only apply when the PLL is locked. Indeed, in the locked state, the phase error is small[1] [170, Page 143], and the behavior of the PLL can be studied through transfer functions [171, Section 2.4]. Whenever the PLL is unlocked, this description does not apply. A time domain description is therefore required. In this case, the analysis of the PLL becomes far more challenging [172]. Therefore, in the remainder of this thesis, we will always assume that the PLL is in a locked state, allowing us to use transfer functions. Indeed, the TRNG is not allowed to output random bitstream before the PLL is locked.

3.1.2.1 Basic PLL transfer functions

The transfer function of an electrical circuit relates voltages or currents of the input and output signals [171, Section 2.2]. However, when dealing with the PLL, we are more interested by the phases of both the input and output signals, not the voltages or currents [172, Page 6]. Therefore, the transfer functions we consider in the study of PLLs relate the phase of a signal applied at a specific location in the PLL to the phase response at the output of the PLL.

Let us thus consider a PLL as in Figure 3.2 having an input signal for which the phase is $\theta_{in}(t)$. The PLL response yields an output signal with a phase denoted $\theta_{out}(t)$, expressed in radian just like $\theta_{in}(t)$. The operation of the phase-frequency detector (combined with the charge pump[2]) results in a phase error $\theta_e(t)$ which corresponds to the phase difference between the input and feedback signals. We thus have:

$$\theta_e(t) = \theta'_{in}(t) - \theta_{fb}(t) = \frac{\theta_{in}(t)}{N} - \frac{C\theta_{out}(t)}{M}. \tag{3.2}$$

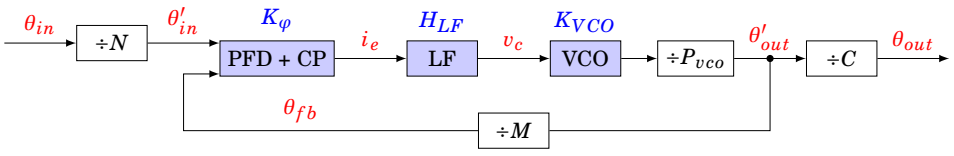


Figure 3.2: Phase-locked loop block diagram including internal PLL parameters (PFD: phase-frequency detector, CP: charge pump, LF: loop filter, VCO: voltage-controlled oscillator, K_φ: gain of the PFD combined with the CP, H_{LF}: transfer function of the loop filter, K_{VCO}: gain of the VCO).

[1] The notion of smallness is relative, but one can consider the phase small if it does not exceed $\frac{\pi}{2}$ [168, Section 1.4].

[2] This is a common practice in PLL modeling [172, Chapter 10]. It is justified by the fact that there is no phase change between the PFD and the CP. This practice helps to simplify the equations in the PLL models.

Since the PLL is locked, the PFD can be considered as a linear component [170, Page 4], and thus the PFD output is a current which can be written as:

$$i_e(t) = K_\varphi \theta_e(t) = K_\varphi \left(\theta'_{in}(t) - \theta_{fb}(t)\right) = K_\varphi \left(\frac{\theta_{in}(t)}{N} - \frac{C\theta_{out}(t)}{M}\right), \tag{3.3}$$

where K_φ is the gain of the PFD, expressed in[3] Amperes per radian [172, Section 12.1].

The current i_e described in Equation (3.3) is the error current. It is processed by the filter, whose purpose is to establish the performance of the loop [173, Page 592]. This filter is also used to attenuate noise and high-frequency components of the input signal [171, Section 2.3.2].

Although a study in the time domain is possible, it is often more convenient to carry out this study in the Laplace domain. Thanks to the linearity of the Laplace transform, Equations (3.2) and (3.3) remain unchanged. The only changes are that the time functions are replaced by their respective Laplace transforms, and that the time variable t is also replaced by the Laplace variable s. In order to remain as general as possible, the transfer function of the loop filter is not specified, rather it is denoted as $H_{LF}(s)$. Depending on the kind of filter used, $H_{LF}(s)$ can easily be replaced by the appropriate expression in final equations.

The loop filter delivers a voltage $v_c(t)$ which controls the frequency of the VCO. In the Laplace domain, the action of the filter is described by:

$$V_c(s) = H_{LF}(s)V_e(s) = K_\varphi H_{LF}(s)\theta_e(s), \tag{3.4}$$

where $V_c(s)$ and $V_e(s)$ are respectively the Laplace transform of $v_c(t)$ and $v_e(t)$.

Notation In the remainder of this chapter, the phases will be designated by the lowercase Greek letter θ and a subscript indicating its origin such as:

- $\theta_{in}(t)$ for the input phase,
- $\theta_{out}(t)$ for the output phase,
- $\theta_{vco}(t)$ for the phase of the VCO.

The presence of the time variable t indicates that the phases are expressed in the time domain. Their respective Laplace transforms will be designated by the same letters, replacing the time variable t with the Laplace variable s. Hence, the Laplace transform of:

[3] Here, the unit is in Amperes because we are dealing with current, due to the presence of the charge pump. Without a charge pump, the phase detector output would be a voltage, Volts would then be used instead of amperes. So the unit depends on the situation and should therefore be chosen as applicable [172, Section 2.1.1].

- $\theta_{in}(t)$ will be denoted as $\theta_{in}(s)$,
- $\theta_{out}(t)$ will be denoted as $\theta_{out}(s)$,
- $\theta_{vco}(t)$ will be denoted as $\theta_{vco}(s)$.

The argument of the phase therefore indicates whether it is considered in the time or Laplace domain.

The deviation of the VCO from its free-running frequency is $\Delta\omega = K_{vco}v_c$ in $\text{rad}\cdot\text{s}^{-1}$, where K_{vco} is the VCO gain factor expressed in $\text{rad}\cdot\text{s}^{-1}\cdot V^{-1}$ [172, Page 8] [171, Page 30]. Taking into account that the radian frequency is the derivative of the phase, and that the VCO produces an output phase [171, Equation 2.33a], the VCO operation may be described as:

$$\frac{\mathrm{d}\theta_{vco}(t)}{\mathrm{d}t} = K_{vco}v_c(t), \tag{3.5}$$

which, in the Laplace domain becomes:

$$\theta_{vco}(s) = \frac{K_{vco}V_c(s)}{s}. \tag{3.6}$$

Since $1/s$ is the Laplace transform of an integration, it can be said that the output phase of the VCO is proportional to the integral of the control voltage. This highlights the integrative nature of the VCO.

The transfer functions of the previous individual elements can be combined to obtain more elaborate transfer functions for finer analyses of the PLL's behaviour. This approach and the resulting equations generally applies to any PLL.

3.2 Transfer functions of an analog PLL

3.2.1 Open loop transfer function

In open loop, the PLL behaves as if the connection between the VCO and PFD would not exist. This situation, illustrated in Figure 3.3, avoids taking into account the feedback signal.

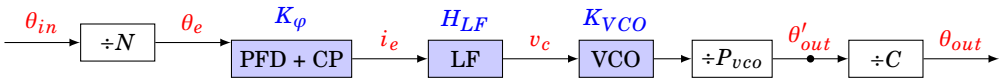


Figure 3.3: PLL in an open loop mode (PFD: phase frequency detector, LF: loop filter, VCO: voltage-controlled oscillator).

From Equations (3.4) and (3.6), one can write:

$$s\theta_{vco}(s) = K_\varphi K_{VCO} H_{LF}(s)\theta_e(s). \tag{3.7}$$

Due to the actions of the post-divider of the VCO and the divider by C, the output phase θ_{out} is:

$$\theta_{out} = \frac{\theta_{vco}}{CP_{vco}}, \tag{3.8}$$

which results in:

$$sCP_{vco}\theta_{out}(s) = s\theta_{vco}(s) = K_\varphi K_{VCO} H_{LF}(s)\theta_e(s). \tag{3.9}$$

Since the PLL is in open loop, we can write $\theta_e = \theta_{in}/N$ and therefore conclude that the open loop transfer function of the PLL is given by:

$$G(s) = \frac{\theta_{out}(s)}{\theta_{in}(s)} = \frac{K_\varphi K_{VCO} H_{LF}(s)}{sNCP_{vco}}. \tag{3.10}$$

In our consideration, the open loop PLL comprises PLL blocks between the input and the output signal. Hence, the feedback signal is not taken into account. Moreover, most existing PLL models do not consider actions of the frequency dividers N, P_{vco}, C. The absence of these dividers corresponds to taking $N = C = P_{vco} = 1$, making Equation (3.10) a generalization of existing models.

3.2.2 Closed loop transfer function

From Section 3.2.1, it is possible to write $\theta_{out}(s) = NG(s)\theta_e(s)$. When the feedback loop is closed, the $\theta_e = \theta_{in}$ assumption is no longer valid. The expression of θ_e is obtained in this case by Equation (3.2). By replacing θ_e by its expression, it follows:

$$\left(1 + \frac{NC}{M}G(s)\right)\theta_{out}(s) = G(s)\theta_{in}(s), \tag{3.11}$$

thus:

$$H_T(s) = \frac{\theta_{out}(s)}{\theta_{in}(s)} = \frac{G(s)}{1 + \frac{NC}{M}G(s)} = \frac{M}{NC} \times \frac{K_\varphi K_{VCO} H_{LF}(s)}{sMP_{vco} + K_\varphi K_{VCO} H_{LF}(s)}, \tag{3.12}$$

which is the closed loop transfer function of the PLL, also known as the jitter transfer function of the PLL [81, Section 7.2.2].

It should be noted that the transfer function of the PLL is dimensionless. Thus, it is possible to consider its input as a frequency signal instead of a phase signal, without any modification.

3.2.3 PLL in presence of disturbing signals

In Sections 3.2.1 and 3.2.2, we have established transfer functions for PLL assuming that it is ideal. In other words, blocks of the PLL do not generate electronic noises. Of course, in practice, this is not true. Hence, in order to take into account the effects of disturbing signals (*e.g* noises)

which could come from various blocks that make up the PLL, we express the output phase of the PLL as a function of the input phase and additional disturbances as shown in Figure 3.4.

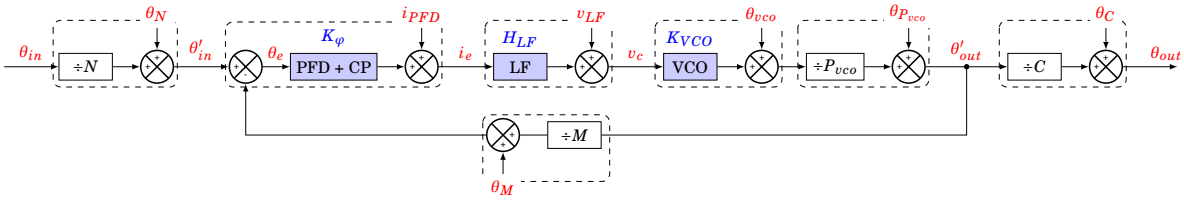


Figure 3.4: PLL loop with disturbance addition (PFD: phase frequency detector, LF: loop filter, VCO: voltage-controlled oscillator).

In Figure 3.4, the considered disturbances are:

- the phase disturbance θ_N produced by the divider N,
- the current disturbance I_{PFD} produced by the PFD,
- the voltage disturbance V_{LF} produced by the filter,
- the phase disturbance θ_{vco} produced by the VCO,
- the phase disturbance $\theta_{P_{vco}}$ produced by the divider P_{vco},
- the phase disturbance θ_C produced by the divider C,
- the phase disturbance θ_M produced by the divider M.

These disturbances can be of various kinds: pulses, noises, etc, even if in practice, we assume that they are noises (zero mean random signals) produced by the different blocks of the PLL. In order to establish the expression of the output phase θ_{out} in this condition, we can start by writing:

$$\theta_{out}(s) = \frac{\theta'_{out}(s)}{C} + \theta_C(s), \tag{3.13}$$

where:

$$\theta'_{out}(s) = \frac{K_{VCO}V_c(s)}{sP_{vco}} + \frac{\theta_{vco}(s)}{P_{vco}} + \theta_{P_{vco}}(s). \tag{3.14}$$

As we have:

$$\begin{aligned}
V_c(s) &= H_{LF}(s)V_e(s)+V_{LF}(s)\\
&= H_{LF}(s)\left[K_\varphi\theta_e(s)+V_{PFD}(s)\right]+V_{LF}(s)\\
&= K_\varphi H_{LF}(s)\theta_e(s)+H_{LF}(s)V_{PFD}(s)+V_{LF}(s)\\
&= K_\varphi H_{LF}(s)\left[\left(\frac{\theta_{in}(s)}{N}+\theta_N(s)\right)-\left(\frac{\theta'_{out}(s)}{M}+\theta_M(s)\right)\right]+H_{LF}(s)V_{PFD}(s)+V_{LF}(s)\\
&= \frac{K_\varphi H_{LF}(s)}{N}\theta_{in}(s)+K_\varphi H_{LF}(s)\theta_N(s)-\frac{K_\varphi H_{LF}(s)}{M}\theta'_{out}(s)-K_\varphi H_{LF}(s)\theta_M(s)\\
&\quad +H_{LF}(s)V_{PFD}(s)+V_{LF}(s),
\end{aligned}$$

it follows that:

$$\begin{aligned}
\left(1+\frac{K_{VCO}K_\varphi H_{LF}(s)}{sMP_{vco}}\right)\theta'_{out}(s) &= \frac{K_{VCO}K_\varphi H_{LF}(s)}{sNP_{vco}}\theta_{in}(s)+\frac{K_{VCO}K_\varphi H_{LF}(s)}{sP_{vco}}\theta_N(s)\\
&\quad +\frac{K_{VCO}H_{LF}(s)}{sP_{vco}}V_{PFD}(s)+\frac{K_{VCO}}{sP_{vco}}V_{LF}(s)\\
&\quad -\frac{K_{VCO}K_\varphi H_{LF}(s)}{sP_{vco}}\theta_M(s)+\frac{1}{P_{vco}}\theta_{vco}(s)+\theta_{P_{vco}}(s).
\end{aligned}$$

Which gives:

$$\begin{aligned}
\theta'_{out}(s) &= \frac{M}{N}\times\frac{K_\varphi K_{VCO}H_{LF}(s)}{sMP_{vco}+K_\varphi K_{VCO}H_{LF}(s)}\theta_{in}(s)+M\times\frac{K_\varphi K_{VCO}H_{LF}(s)}{sMP_{vco}+K_\varphi K_{VCO}H_{LF}(s)}\theta_N(s)\\
&\quad +M\times\frac{K_{VCO}H_{LF}(s)}{sMP_{vco}+K_\varphi K_{VCO}H_{LF}(s)}V_{PFD}(s)+M\times\frac{K_{VCO}}{sMP_{vco}+K_\varphi K_{VCO}H_{LF}(s)}V_{LF}(s)\\
&\quad -M\times\frac{K_\varphi K_{VCO}H_{LF}(s)}{sMP_{vco}+K_\varphi K_{VCO}H_{LF}(s)}\theta_M(s)+M\times\frac{s}{sMP_{vco}+K_\varphi K_{VCO}H_{LF}(s)}\theta_{vco}(s)\\
&\quad +M\times\frac{sP_{vco}}{sMP_{vco}+K_\varphi K_{VCO}H_{LF}(s)}\theta_{P_{vco}}(s).
\end{aligned}$$

In order to deduce the transfer functions related to the various solicitations of the PLL, it is assumed that all solicitations are null except the one of interest. This makes it possible to write:

$$H_T(s) = \frac{M}{NC}\times\frac{K_\varphi K_{VCO}H_{LF}(s)}{sMP_{vco}+K_\varphi K_{VCO}H_{LF}(s)}, \tag{3.15}$$

$$H_N(s) = \frac{M}{C}\times\frac{K_\varphi K_{VCO}H_{LF}(s)}{sMP_{vco}+K_\varphi K_{VCO}H_{LF}(s)} = N\times H_T(s), \tag{3.16}$$

$$H_{PFD}(s) = \frac{M}{C}\times\frac{K_{VCO}H_{LF}(s)}{sMP_{vco}+K_\varphi K_{VCO}H_{LF}(s)} \tag{3.17}$$

$$H_F(s) = \frac{M}{C}\times\frac{K_{VCO}}{sMP_{vco}+K_\varphi K_{VCO}H_{LF}(s)} \tag{3.18}$$

$$H_M(s) = -\frac{M}{C}\times\frac{K_\varphi K_{VCO}H_{LF}(s)}{sMP_{vco}+K_\varphi K_{VCO}H_{LF}(s)} = -N\times H_T(s), \tag{3.19}$$

$$H_{VCO}(s) = \frac{M}{C}\times\frac{s}{sMP_{vco}+K_\varphi K_{VCO}H_{LF}(s)} \tag{3.20}$$

$$H_{P_{vco}}(s) = \frac{M}{C} \times \frac{sP_{vco}}{sMP_{vco} + K_\varphi K_{VCO} H_{LF}(s)} = P_{vco} \times H_{VCO}(s), \tag{3.21}$$

$$H_C(s) = 1. \tag{3.22}$$

Note that H_{VCO} is also known as the PLL jitter generation function, and denoted H_G [81, Section 7.2.1]. This is because it relates the jitter of the VCO to the output of the PLL, and because the jitter in the output essentially comes from the VCO as we will see in Section 3.4.1.

The model of the PLL in terms of the transfer function introduced in this section will be used to study the PLL and have a better understanding of its operation. It will also be used to identify the origin of phase fluctuations at the output of PLL. This model can be described as:

- the open loop transfer function of the PLL is given by Equation (3.10),
- the output phase of the PLL (in closed loop) is defined by:

$$\begin{aligned}\theta_{out}(s) = {} & H_T(s)\cdot\theta_{in}(s) + H_{PFD}(s)\cdot V_{PFD}(s) + H_F(s)\cdot V_{LF}(s) + H_{vco}(s)\cdot\theta_{vco}(s) \\ & + N\cdot H_T(s)\cdot(\theta_N(s) - \theta_M(s)) + P_{vco}\cdot H_{vco}(s)\cdot\theta_{P_{vco}}(s) + H_C(s)\theta_C(s),\end{aligned}$$

  where $H_T, H_{PFD}, H_F, H_{vco}$ and H_C are respectively defined by Equations (3.15), (3.17), (3.18), (3.20) and (3.22).

3.3 Physical parameters of the PLL model

The PLL model in transfer functions presented in Section 3.2 is intended to be used to simulate the behavior of the PLL. In particular, it allows to study the PLL's response to a noise generated in one of its blocks. In order to be able to use this model, it is necessary to evaluate it and to ensure that it produces realistic results.

3.3.1 Comparison with existing models

The validation of a model requires a comparison between the results predicted by the model and real-world experiments. In the case of a PLL, this requires access to the component. However, the PLLs we use are embedded in an FPGA. Therefore, we do not have access to the internal signals of the PLL. For this reason, we compare the model presented in Section 3.2 with existing models. The main difference between the above model and those existing in the state-of-the-art is the consideration of frequency division blocks. Indeed, most models consider a PLL without a frequency divider [174, 175, 172, 176, 81], others consider a PLL with only one frequency divider present in the feedback loop [177, 171, 170, 178]. However, PLLs embedded in the FPGAs are as presented in Figure 3.1.

The model of the PLL in terms of transfer function is justified by the fact that it is a feedback system. It can therefore be represented as a block diagram [179, Section 8.3]. By replacing M, N, C and P_{vco} with 1 in the model in Section 3.2, the PLL open loop transfer function is expressed as:

$$G(s) = \frac{K_\varphi K_{VCO} H_{LF}(s)}{s}, \tag{3.23}$$

which is consistent with existing models [172, Equation 2.5] [168, Equation 1.36]. Likewise, the closed loop transfer function is expressed as:

$$H_T(s) = \frac{K_\varphi K_{VCO} H_{LF}(s)}{s + K_\varphi K_{VCO} H_{LF}(s)}, \tag{3.24}$$

as stated in the state-of-the-art [172, Equation 2.6] [168, Equation 1.35] [174, Section 3.5].

This proves that the model in Section 3.2 makes it possible to find the open-loop and closed-loop transfer functions of the PLL present in the state-of-the-art. This model also makes it possible to observe the response of PLL to signals (especially random signals) that occur in various blocks of the PLL. In particular, we can simulate the behavior of the PLL as a result of the noise that occurs in the VCO. The corresponding transfer function, when M, C and P_{vco} are set to 1, is written:

$$H_G(s) = \frac{s}{s + K_\varphi K_{VCO} H_{LF}(s)} \tag{3.25}$$

and is identical to the one found by Da Dalt and Sheikholeslami [81, Section 7.2.1].

We can therefore see that the model presented in Section 3.2 allows us to find the PLL transfer functions available in the state-of-the-art. Moreover, it allows to study the influence of noise generated in various blocks of the PLL, which is not available in the state-of-the-art. We can therefore say that this model generalizes those present in the state-of-the-art. Since the intended end use of the model is to be used for simulations, it is important to choose appropriate physical parameters.

3.3.2 Choice of physical parameters

To continue our study, we need to specify the type of filter used. However, we do not have detailed description of the PLLs used in FPGAs for two reasons. First because this information is (at least partially) confidential, and second because the PLL structure (architecture) may be different for each FPGA family. Nevertheless, based on the state-of-the-art, we can say that the most commonly-used filters are first and second order low-pass filters. They can be either passive or active. In the case of passive filters, we distinguish between lag filters (which have no zero) [168, Section 2.2.1] [176, Section 13.5.1], and lead-lag filters (transfer functions with one zero) [171, Section 2.3.2] [168, Section 2.2.2] [180, Section 3.3.1] [176, Section 13.5.2]. Active filters used in

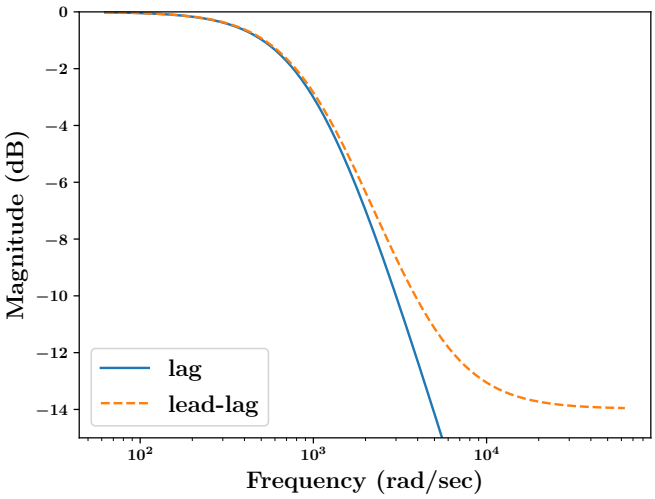


Figure 3.5: Comparison of a lag filter to a lead-lag filter (Bode magnitude plot of the two first order passive low-pass filters).

PLLs are often lead-lag filters [171, Section 2.3.2] or PI filters which incorporate an integrator [171, Section 2.3.2] [176, Section 13.5.3]. The use of a second or higher orders low-pass filter is less common and dedicated to specific applications [168, Section 3.1.3].

For simplicity, we opted for a first-order passive low-pass filter. Specifically, we will use a first order passive lag filter. This choice is justified by Figure 3.5, where we can see that lead-lag filters do not filter high frequencies as well as lag filters do. It may be noted that the study done in this thesis can be done with other types of filters. For the rest of this thesis, the transfer function of the loop filter will therefore be written:

$$H_{LF}(s) = \frac{1}{\tau s + 1}, \tag{3.26}$$

where $\tau := R_{LF} \times C_{LF}$ is the time constant of the low-pass filter which consists of a resistor (R_{LF}) and a capacitor (C_{LF}). The PLL jitter transfer function can then be expressed as:

$$H_T(s) = \frac{M}{NC} \times \frac{K_\varphi K_{VCO}}{s^2 M P_{vco} \tau + s M P_{vco} + K_\varphi K_{VCO}}. \tag{3.27}$$

Likewise, the PLL jitter generation function can be expressed as:

$$H_G(s) = \frac{M}{C} \times \frac{s^2 \tau + s}{s^2 M P_{vco} \tau + s M P_{vco} + K_\varphi K_{VCO}}. \tag{3.28}$$

In order to simulate the PLL, it is necessary to specify the values of the physical parameters, namely the PFD gain K_φ, the VCO gain K_{VCO}, as well as the filter time constant $\tau = R_{LF} \times C_{LF}$. As noted above, these values are unknown to us, and mostly kept secret by designers. In order to

choose values for these parameters, we have taken the values available in the state-of-the-art as a starting point [174, Chapter 2]. Based on this available values of the parameters, we compared the responses of the simulations with the responses of the PLL implemented in an Intel Cyclone V FPGA device 5CEBA4F17C8N mounted on the Evariste chip [181]. Thanks to this, we corrected values of the parameters until there was a match between responses of the simulations and the real-world experiments. Physical parameters we obtained through this procedure are:

- $K_\varphi = 29.39\,\mathrm{A}\cdot\mathrm{rad}^{-1}$,
- $K_{VCO} = 2\,\mathrm{Mrad}\cdot\mathrm{s}^{-1}\cdot\mathrm{V}^{-1}$,
- $\tau = 1\,\mu\mathrm{s}$.

3.4 Noise properties

In our application framework, the PLL is intended to be used as a source of randomness for the generation of true random numbers. Since the quality of the source of randomness strongly impacts the quality of the generated numbers, it is important to consider the properties of the noise at the output of the PLL. This will allow the output noise to be evaluated according to the physical parameters of the PLL.

3.4.1 Origin of the output noise

Most PLL designers admit that the noise at the output of the PLL comes mainly from the VCO [172, Section 15.4.1]. However, we did not find any argument to support this fact in the state-of-the-art. In this section, we propose elements of confidence to reinforce the fact that the output noise comes from the noise present in the VCO. For this analysis, we will focus on the analog blocks of the PLL and ignore digital ones. The reason is twofold:

- first because Section 3.2.3 shows that the transfer functions of the PLL's digital blocks can be derived from the transfer functions of analog blocks by scalar multiplication. Since scalar multiplication does not modify the spectral properties, but only the magnitude, we can deduce that these blocks will behave in a similar manner to the analog blocks from which the transfer functions are deduced.
- the second reason is that digital blocks are frequency dividers implemented by counters. It is admitted that these counters produce negligible amount of noise compared to analog components of the PLL [182, Section E.1.3].

Based on this reasoning, we assume in this section that only analog blocks produce significant amount of noise.

By using the corresponding transfer functions, described in Section 3.2.3, we can study the response of the PLL to noise generated in the various blocks of the PLL. We therefore simulated the generation of white phase noise coming from the input signal, the PFD, the loop filter, and the VCO. To find out which of these noises is dominant at the output of the PLL, we compared the PLL responses to these various noises to experimental data coming from a real-world PLL.

For this, we collected 250,000 periods of the PLL with a sampling period of about 8 ns (which corresponds to the maximum capacity of the oscilloscope used[4]). For consistency reasons, we used these same values to simulate white noise using Kasdin and Walter algorithm. The time Allan variance response to the real-world PLL output noise and simulated ones yields Figure 3.6, where simulated data are in red and green, and experimental ones in cyan.

We see that the experimental data behaves in the same way as the simulated data in the only case where the noise comes from the VCO. This shows that the noise contained in the experimental data cannot come from neither the PLL input nor from an internal block other than the VCO. On the other hand, by comparing the time Allan variance responses to theoretical models (dashed lines), we find that noises from the input signal and the PLL blocks except the VCO translates into noise with a phase power spectral density $S_x(f)$ proportional to $1/f^4$ (slope proportional to $\sqrt{h_{-2}}\tau^{3/2}$ in Figure 3.6). This type of noise is of very low frequency. Thus, the impact of other PLL noises, except that of VCO, on the PLL output are much smaller and can be neglected. This reinforces the assumption that the noise at the output of the PLL coming from the VCO dominates. For this reason, we will consider that the noise at the output of the PLL comes from the VCO.

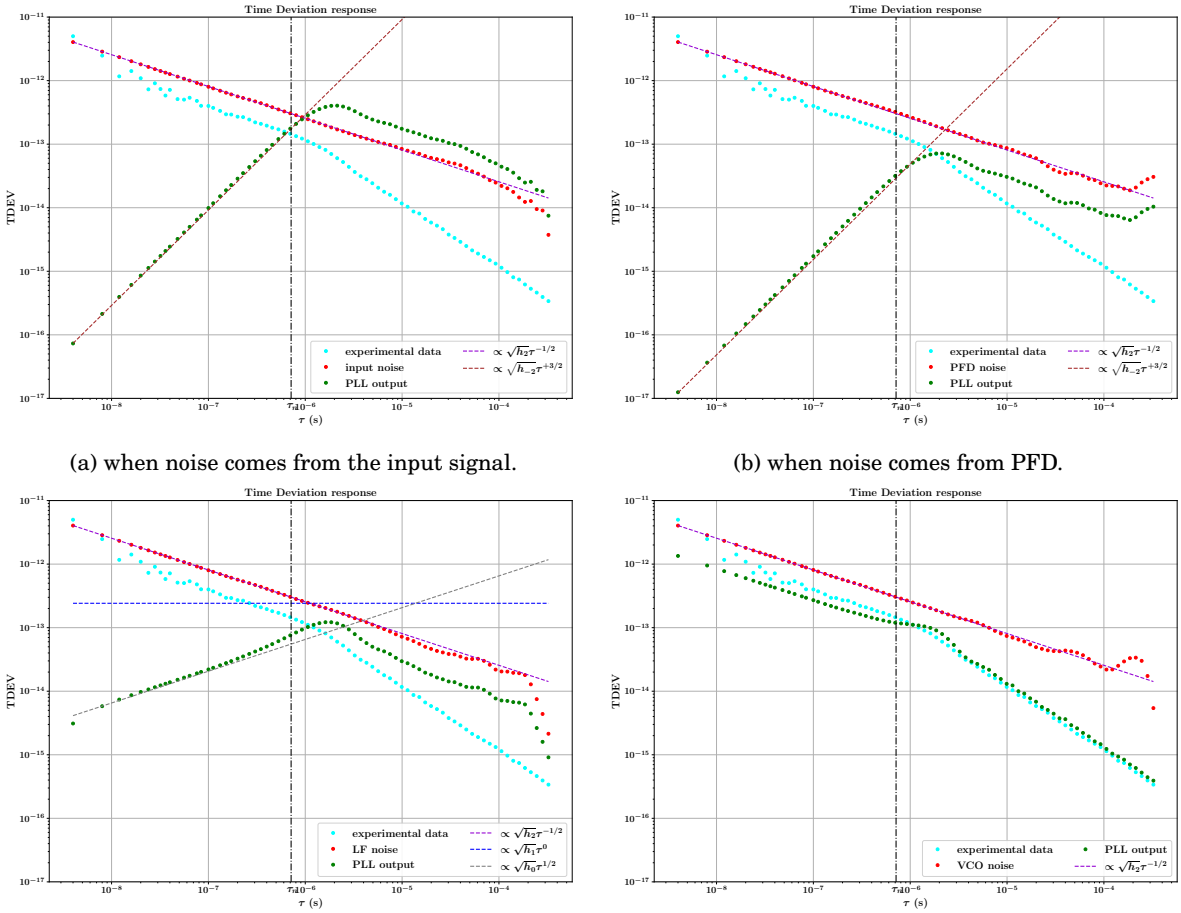
Since the output noise of the PLL comes mainly from the VCO, it is possible to ignore (in a first approach) the noise coming from the other blocks of the PLL. The model described in Section 3.2.3 can therefore be simplified to take into account only the impact of the input noise and the VCO noise. The output noise of the PLL can then be described by the relationship:

$$\theta_{out}(s) = H_T(s)\theta_{in}(s) + H_{vco}(s)\theta_{vco}(s), \tag{3.29}$$

or in terms of power spectral densities:

$$S_{out}(s) = H_T(s)H_T(-s)S_{in}(s) + H_{vco}(s)H_{P_{vco}}(-s)S_{vco}(s). \tag{3.30}$$

[4] LeCroy WavePro 7 Zi-A Series [183].



(a) when noise comes from the input signal.

(b) when noise comes from PFD.

(c) when noise comes from the loop filter.

(d) when noise comes from the VCO.

Figure 3.6: Noise type at the output of a PLL ($M = 35, N = 5, C = 3$, white phase noise generated using Kasdin and Walter algorithm).

Remembering that s is the Laplace variable, if one takes $s = j2\pi f$, it follows that:

$$S_{out}(f) = |H_T(f)|^2 S_{in}(f) + |H_{vco}(f)|^2 S_{vco}(f), \tag{3.31}$$

where $H_T(f)$ and $H_{vco}(f)$ are respectively the PLL jitter transfer and jitter generation functions.

3.4.2 Noise filtering and jitter overshoot

An interesting fact is that the PLL tends to act like a low-pass filter with respect to the input signal and high-pass filter with respect to the VCO signal, as shown in Figure 3.7. Thus, the PLL will let low-frequency fluctuations present in the input signal pass through, while it will attenuate high-frequency input fluctuations. In a similar manner, the PLL will attenuate low-frequency fluctuations present in the VCO signal, while allowing high-frequency fluctuations to pass through.

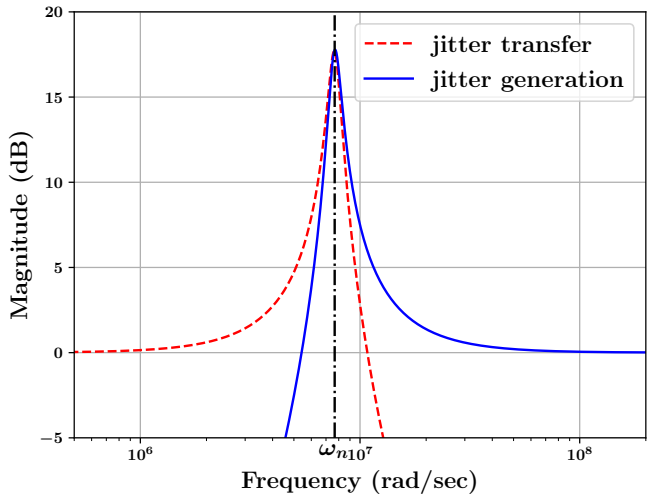


Figure 3.7: Bode magnitude plot of the jitter transfer and jitter generation functions of a PLL ($M = 1$, $N = 1$, $C = 1$ and $P_{vco} = 1$, leading $\zeta = 0.0652$ and $\omega_n = 7666811.5928\,\text{rad}\cdot\text{s}^{-1}$).

3.4.2.1 Jitter peaking

It should be noted that the notion of low or high-frequency is related to a reference frequency. In this study, the reference frequency is the natural frequency ω_n of the PLL visible in Figure 3.7. It is the frequency at which the response of the PLL has a peak, and can be found by writing the normalized form of the PLL jitter transfer function:

$$H_T(s) = \frac{M}{NC} \times \frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2}, \tag{3.32}$$

where ζ is the PLL damping factor [174, Section 3.5]. By identifying with Equation (3.27), we can deduce that:

$$\omega_n := \sqrt{\frac{K_\varphi K_{VCO}}{M P_{vco} \tau}}, \tag{3.33}$$

and

$$\zeta := \frac{1}{2\omega_n \tau}. \tag{3.34}$$

ω_n is the frequency at which the gain (the module) of both transfer functions is at its maximum. This appears as a peak, as shown in Figure 3.7, called jitter peaking [81, Section 7.2.2].

3.4.2.2 PLL response to input noise

Since the PLL behaves as a low-pass filter with respect to its input signal, it follows that the PLL is able to filter out high-frequency noises impacting its input signal. This results in an attenuation of these noises for accumulation times lower than $\tau_n := \frac{1}{\omega_n}$, as we can see in Figure 3.6a. They will therefore have little influence on the PLL output signal, unless their amplitude in the input

signal is large enough to drown out the useful information. In this case, the PLL will lose its lock [184], and the generation of random numbers will then have to be stopped. Hence, high-frequency noises from the PLL input signal have very little influence on the generation of random numbers. Their influences can therefore be neglected.

Note that for accumulation times lower than τ_n, the noises that dominate are those with high amplitudes for frequencies higher than ω_n. Thus, low-frequency noises, such as flicker noise, which have low amplitudes for frequencies above ω_n accumulate very little for times below τ_n. Their amplitudes at the output of the PLL are therefore very small compared to the amplitude of the input noise as we can see in Figure 3.8. Conversely, for accumulation times greater than τ_n, these noises have enough time to accumulate and their amplitudes are no longer negligible. As we might expect, the simulations show that the PLL then locks onto them. It is therefore important to identify the low-frequency noises that impact the input signal, and to characterize them when we plan to accumulate randomness for times greater than τ_n.

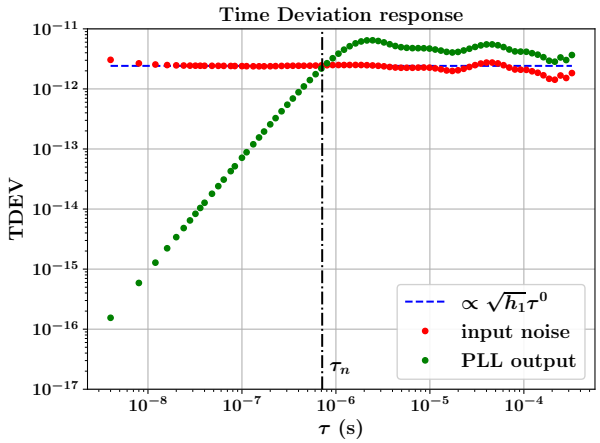


Figure 3.8: PLL response to flicker phase noise in the input signal.

We have therefore seen that high-frequency phase fluctuations play very little role in the generation of random numbers. However, it is necessary to provide a test to detect a loss of PLL lock due to the action of these noises in order to stop the generation of random numbers. The noises at the output of the PLL are the low-frequency noises that need to be characterized. It is however possible to ignore them if the accumulation times is less than τ_n. Another solution would be to use an input signal that is very little influenced by low-frequency noises. This can be done by using output signals of a crystal oscillator [185]. This last option allows us to assume that there is no jitter transfer. According to Equation (3.31), this means that phase fluctuations in the output signal come only from the VCO, making it possible to consider the VCO as the only source of

randomness for TRNGs.

3.4.2.3 PLL response to VCO noise

As we saw at the beginning of Section 3.4.2, the PLL behaves like a high-pass filter with respect to the signal coming from the VCO. It can then be deduced that at low frequencies, the PLL strongly attenuates the different signals present in the VCO. Knowing that the low frequencies correspond to frequencies lower than ω_n, it follows that for accumulation times higher than τ_n, the signals are strongly attenuated as we can see in Figures 3.6d and 3.9. Since flicker noise is a low frequency noise, its amplitude can be considered negligible for times below τ_n. For times above τ_n (corresponding to the region where flicker noise is likely to dominate), Figure 3.9 shows a high attenuation of the signal, leading to a low level of flicker noise at the output of the PLL.

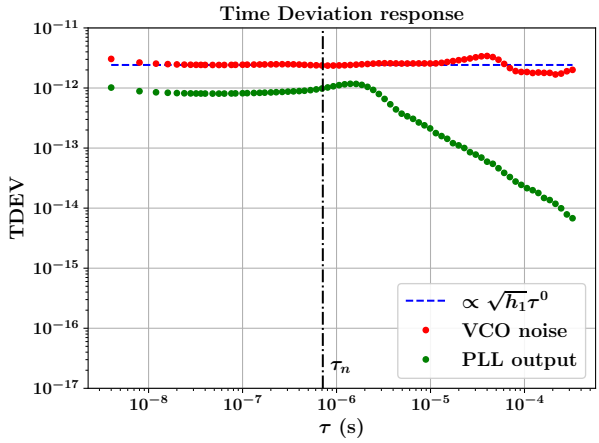
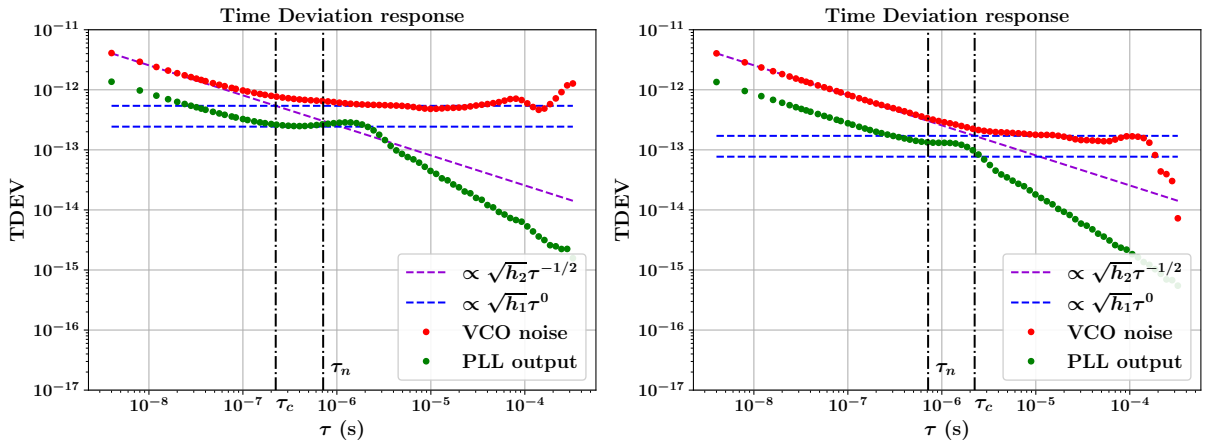


Figure 3.9: PLL response to flicker phase noise in the VCO.

In the low frequencies, for $\tau > \tau_n$, the thermal noise will suffer the same fate as the flicker noise, namely a magnitude attenuation. This is due to the fact that the PLL attenuates any signal present in the low frequencies. In addition, as shown in Figure 3.6d, the PLL denatures thermal noise in the region $\tau > \tau_n$. Indeed, for such large accumulation times, the feedback loop and bandwidth of the PLL induces deterministic actions that prevent a free accumulation of thermal noise. A characterization of these actions is therefore required for a proper evaluation of the quality of the generated numbers. Otherwise, the accumulation time should be limited to the region $\tau < \tau_n$, in order to avoid that the nature of the noise at the output of the PLL is altered.

We have thus seen that the ideal to generate random numbers would be to limit entropy accumulation to times $\tau < \tau_n$. These times correspond to the region where the noises generated in the VCO are not altered by the PLL at the output. This is also the region in which high-frequency



(a) PLL response for $\tau_c < \tau_n$ ($h_1 = 1.03 \cdot 10^{-23}\,\mathrm{rad}^2 \cdot \mathrm{s}^2$, $h_2 = 5.16 \cdot 10^{-30}\,\mathrm{rad}^2 \cdot \mathrm{s}^3$, $\tau_c = 2.23 \cdot 10^{-7}\,\mathrm{s}$).

(b) PLL response for $\tau_c > \tau_n$ ($h_1 = 1.03 \cdot 10^{-24}\,\mathrm{rad}^2 \cdot \mathrm{s}^2$, $h_2 = 5.16 \cdot 10^{-30}\,\mathrm{rad}^2 \cdot \mathrm{s}^3$, $\tau_c = 2.23 \cdot 10^{-6}\,\mathrm{s}$).

Figure 3.10: PLL response to VCO noise consisting of thermal and flicker noises ($\tau_n = 7.14 \cdot 10^{-7}\mathrm{s}$ is the inverse of the PLL natural frequency).

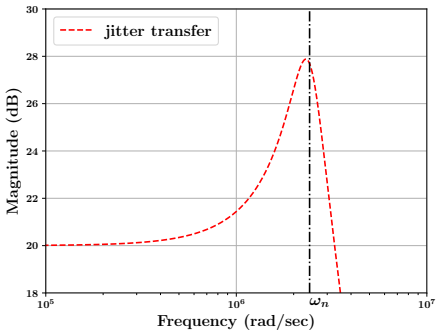
noises are most likely to dominate over low-frequency ones. Limiting to this region therefore ensures that only thermal noise is present at the output of the PLL, provided that the critical time τ_c is greater than τ_n. Indeed, as can be seen in Figure 3.10b, the flicker generated by the VCO cannot dominate the output if τ_c is greater than τ_n. This is not the case if τ_c is smaller than τ_n as can be seen in Figure 3.10a. Resulting in a significant presence of the flicker which might lead to overestimating the entropy of the source. As expressed in Equation (2.100), the critical time τ_c depends on the noise levels h_1 and h_2 which are specific to the oscillator that constitutes the VCO. It is therefore important that PLL designers bear this in mind in order to ensure, as far as possible, the relationship $\tau_c > \tau_n$.

3.4.2.4 Lowering the jitter peaking

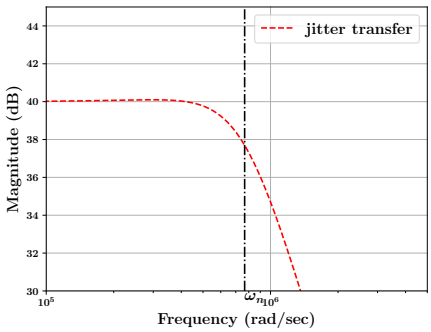
Section 3.4.2.1 highlighted a peak of the jitter in the neighborhood of the natural frequency ω_n of the PLL. This peak, slightly higher than the loop gain, implies that the jitter of any signal present in this frequency range will be amplified, whether it comes from the input signal or from the VCO. This will definitely affect the quality of the PLL output and thus yields some overestimation of the entropy. To avoid this effect, it is preferable to lower the peaking as much as possible.

Although it is not easy to express the amplitude of this peaking, the literature on PLLs reveals that this amplitude depends on the value of the damping factor ζ[81, Section 7.2.2] [172, Section 2.2.4]. These theoretical results were confirmed by simulations using the model presented in Section 3.2. Indeed, as can be seen in Figure 3.11, the smaller the damping factor, the greater the amplitude of the peak. A logical deduction to reduce the amplitude of this peak as much as

possible would therefore be to increase the value of the depreciation factor as much as possible.



(a) Bode magnitude plot for $M = 10$, leading to $\zeta = 0.2062$.



(b) Bode magnitude plot for $M = 100$, leading to $\zeta = 0.6521$.

Figure 3.11: Effect of the damping factor on the jitter peaking.

The simulations carried out have shown that this peak is almost non-existent for $\zeta = 1$. Thus, a damping factor greater than 1 would already be acceptable. As illustrated in Equations (3.33) and (3.34), this damping factor depends on several physical parameters to which only PLL designers have access. On the design level of a TRNG, as a user of the PLL, the only accessible parameter is the multiplication factor M. We can therefore assume the physical parameters set once and for all, and look for the values of M for which we have a large damping factor.

An analysis of Equations (3.33) and (3.34) leads to the conclusion that large values of M would be the most appropriate to avoid this jitter peaking. Using the values of the physical parameters in Section 3.3, we obtain a near-zero peak for $M \geqslant 138$.

It should however be noted that a high value of M increases the response time of the PLL, and therefore increases the risk of integrating a larger amount of flicker noise. Indeed, Equation 3.33 shows that an increase in the value of M leads to a decrease in ω_n, and thus to an increase in τ_n. If this increase of τ_n is not controlled, there is a significant risk of finding ourselves in the situation $\tau_n > \tau_c$ as shown in Figure 3.10a. We would then have an output signal impacted by flicker noise as explained in Section 3.4.2.3.

Even if the jitter at the output of the PLL is bounded, as we will see in Section 3.4.4, it is imperative to take into account this integration of the flicker noise in data at the output of the PLL. Since it is a type of noise that we are trying to avoid, care should be taken to find a trade-off between the value of M and the relative position of τ_n and τ_c. However, this can be facilitated when

designing PLLs. Indeed, by choosing appropriate physical parameters, it is possible to reduce this jitter peak, even with small values of M.

3.4.3 Types of noise at the output of the PLL

According to Section 3.4.1, the VCO is the main contributor to the noise at the output of the PLL. Section 3.4.2 shows that when the noise comes from the VCO, only the fluctuations present for frequencies higher than ω_n are passed through. Hence, if we assume the condition $\tau_n < \tau_c$ verified, and we limit the entropy accumulation to times $\tau < \tau_n$, we can reasonably assume that the output of the PLL is mainly composed of thermal noise. This conjecture is confirmed by the results presented in Figure 3.6d, where simulated data shows that for any accumulation time below τ_n, the noise at the output of the PLL can be considered as white noise (slope proportional to $\sqrt{h_2}\tau^{-1/2}$ in Figure 3.6). Similar observations are also visible in the experimental data.

Beyond τ_n, we observe that the fluctuations at the output of the PLL follow a new slope, different from that of white noise. This allows us to think of a modification of the nature of the white noise under the action of the PLL for frequencies lower than ω_n. Thus, in addition to attenuating the signals present in the low frequencies, this fact suggests that the PLL changes their spectra, and thus their natures. This observation does not contradict the theory, since Equation (A.26) shows that the spectrum of the output signal from a LTI is not necessarily the same as that of the corresponding input signal. This observation reinforces the conviction to confine oneself to accumulation times lower than τ_n. If security requirements demand a longer accumulation time, then ω_n should be lowered so that the accumulation time is always smaller than τ_n.

However, it should be noted[5] that with a small ω_n, there is a significant risk of letting low-frequency noise such as flicker noise pass through. Since we want to avoid integrating these types of noise, a trade-off is needed, in order to have ω_n small enough to accumulate enough thermal noise at the output of the PLL, but high enough not to integrate flicker noise.

3.4.4 Bounded nature of the PLL noise

For generating random numbers, we are interested in the jitter, which is a fluctuation of the period of a given signal from its nominal value. We know that this fluctuation in the period is due to the influence of various electronic noises present in the device. Since, in reality, signals are observed over time, we actually have access to an accumulation of these noises. This accumulation results in a random walk if only the thermal noise is present. By its nature, random walk is not

[5] By comparing it with the Bode diagram in Figure 3.7.

stationary and can therefore take unbounded values.

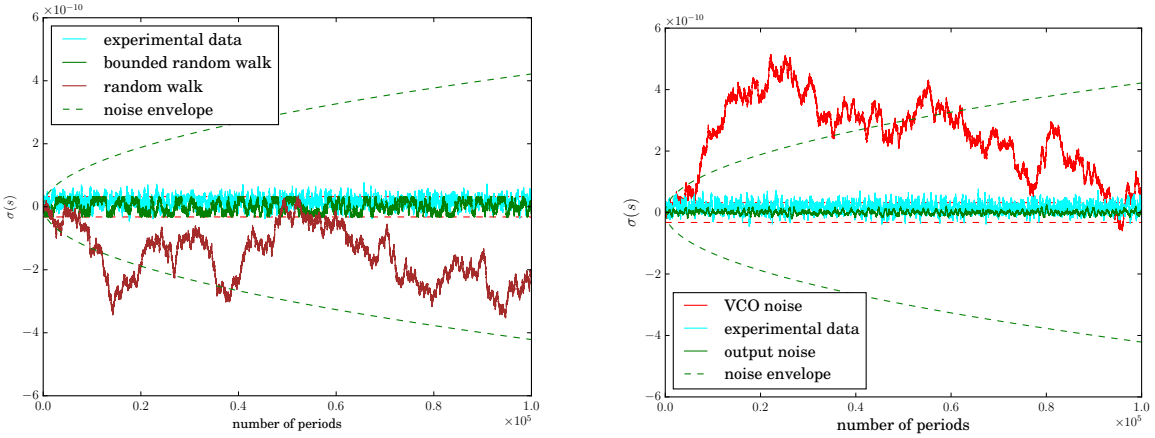
However, if we consider a random walk in the VCO, because of the feedback from the PLL, we can assume that the output noise will be bounded. Indeed, if it was not the case, the phase error would increase indefinitely. This would be in contradiction with the principle of operation of the PLL. This is demonstrated by Figure 3.12.

To obtain Figure 3.12a, we have retrieved a set of successive periods from the output signal of a PLL. By making a cumulative sum to simulate the accumulation of the noise, we can see that the noise at the output of the PLL looks like a bounded random walk. To confirm this observation, we simulated a bounded random walk, as described in Listing 3.1. This bounded random walk appears in green in Figure 3.12a. We can notice that the simulated noise behaves in a similar way to the noise at the output of the PLL. This gives confidence that the noise at the output of the PLL is bounded, otherwise it would behave similarly to the random walk represented in brown.

```
import numpy as np

def bounded_random_walk(sigma, bound, nbr):
    """
    Function generating bounded random walk noise

    Parameters
    ----------
    sigma: float
         Standard deviation (spread or "width") of the normal distribution.
         Must be non-negative.
    bound: float
         Threshold of the bounded random walk.
         Must be non-negative.
    nbr: int
       Number of steps of the bounded random walk.
       Must be non-negative.

    Returns
    -------
    noise : ndarray
          An array of the steps of the bounded random walk.
    """

    noise = [0]
    for i in range(nbr):
```

```
    step = np.random.normal(scale=sigma)
    while abs(noise[-1] + step) > bound:
      step = np.random.normal(scale=sigma)
    noise.append(noise[-1] + step)
  return np.array(noise)
```

Listing 3.1: Generation of a bounded random walk using Python



(a) Simulated bounded random walk versus PLL experimental data.

(b) PLL model versus PLL experimental data.

Figure 3.12: Bounded accumulation of the jitter at the output of the PLL (obtained by making cumulative periods).

These observations are also confirmed by the model of the PLL presented in Section 3.2.3. Indeed, by simulating a random walk in the VCO, we can observe in Figure 3.12b that the output of the PLL is bounded. It could thus be assumed that a random walk taking place within the VCO would manifest itself as a bounded random walk at the output of the PLL.

3.5 Conclusion

In this chapter, we described the PLL in terms of transfer functions. This description allowed us to study the behavior of the PLL as a source of randomness. By injecting phase fluctuations into the different blocks of the PLL, the analysis of the output with the use of the time Allan variance allowed us to confirm that the noise at the output of the PLL comes essentially from the VCO. These simulation results are in agreement with the experiments conducted on the PLLs embedded in FPGAs. These simulations show that if the entropy accumulation time is limited by the inverse of the natural frequency of the PLL, it is reasonable to assume that the fluctuations at the output of the PLL are white noise.

Hence, the quality of the phase fluctuations depends on the natural frequency of the PLL. The lower this frequency, the longer the time during which we have a good quality of the jitter. A smaller natural frequency allows the jitter to accumulate longer during the generation of random numbers. We also observed that the jitter at the output of the PLL is bounded, thus limiting the maximum level of entropy available at the output of the PLL. A thorough study of this character is therefore recommended in order to deduce the values of these bounds. Finding them will make it possible to evaluate the maximum entropy that the PLL can provide. This will allow designers of PLL-based TRNGs to focus on parameters that ensure an optimal entropy rate as we will see in Chapter 4.

Résumé

Les boucles à verrouillage de phase (abrégé en PLL, de l'anglais Phase-Locked Loop) sont des composants électroniques utilisés dans les circuit logiques principalement pour générer des signaux d'horloge. Afin de générer des nombres aléatoires propres à être utilisés en cryptographie, il est exigé entre autre que la source d'aléa soit robuste face aux tentatives de manipulations. Or il se trouve que les PLLs sont isolées, d'un point de vue physique et électrique, des autres composants des circuit logiques. Cette isolation de la PLL la rend moins sensible aux interférences que pourraient causer les autres composants. Ceci en fait un composant dont le fonctionnement est difficile à perturber. De plus, son système de contrôle lui permet de se déverrouiller en présence de fluctuations trop importantes pouvant survenir au niveau de son signal d'entrée. À cela s'ajoute la capacité de la PLL d'assurer une relation rationnelle préalablement définie entre les signaux d'entrée et de sortie.

Dans ce chapitre, nous nous proposons d'étudier de la PLL et d'évaluer ses propriétes relatives au bruit. Pour cela, nous établissons un modèle en fonctions de transfert des PLLs présents dans les FPGAs que nous utilisons. A partir de ce modèle, nous confirmons que le bruit en sortie de la PLL provient de son oscillateur commandé en tension (VCO). Nous déterminons également le type dominant du bruit en sortie de la PLL en fonction du temps d'accumulation du jitter. Ceci a permis de fixer un temps maximal d'accumulation garantissant la domination du bruit blanc en sortie de la PLL. Ces résultats ont été obtenus grâce à l'utilisation de la variance temporelle d'Allan présentée dans le chapitre 2. Nous mettons également en évidence le caractère borné du jitter en sortie de la PLL, dû en partie à la contre-réaction de la PLL. Ce fait pourrait servir à estimer l'entropie maximale à laquelle on peut s'attendre en sortie de la PLL. Toutes ces propriétés de la PLL en font un bon candidat pour être une source d'aléa lors de la génération des nombres aléatoires. A cet effet, nous verrons dans le chapitre 4 qu'un TRNG basé sur les PLLs répond aux exigences de la DGA-MI.

Chapter 4

Design of a certifiable PLL-based TRNG

Contents

True random number generators that use phase-locked loops (PLLs) as source of randomness have proved to be of great interest [82]. Construction of such TRNGs can be formalized using stochastic models [92], making it certifiable by certification bodies. Because of the availability of PLLs in most FPGAs, the implementation of PLL-based TRNGs can be considered as low cost compared to TRNGs based on other sources of randomness. Moreover, PLLs are physically isolated from the rest of the FPGA, making them less sensitive to the logic implemented in the FPGA and cross-talks. All these illustrates the suitability of the implementation of PLL-based TRNGs. However, stochastic models use assumptions, some of which could not be verified, particularly because the PLLs are considered as black boxes. In this chapter, we will present the general concept of a TRNG based on PLLs. We will present the certification approach recommended by DGA-MI and illustrate it with the principle based on PLLs. For this purpose, we will use the results of

Chapter 3. This illustration will lead us to a method aimed at optimizing parameters of the PLL in the context of random number generation.

4.1 Principle of a PLL-based TRNG

In FPGAs, free-running oscillators are implemented in FPGA fabric, while PLLs are hardwired in a physically isolated region, in which the impact of processes running in the FPGA fabric is significantly reduced. This is considered to be the main advantage of implementing PLL-TRNG in FPGAs. Its simple and comprehensive design based on the use of one [82, 186, 85] or two PLLs [71], the availability of the stochastic model [92] and dedicated tests [187] are other important advantages. Moreover, the use of PLL ensures good stability and repeatability of results in different devices in time, and in large ranges of temperatures and power supply voltages.

The generation of random numbers using the jitter of the PLL was first suggested by Fischer and Drutarovsky [82]. In this type of generators, the source of randomness used is the phase difference between the reference clock (ideally not jittered) and the (jittered) clock signal produced by the PLL. As we have seen in Section 3.4.4, the operation of the PLL ensures that this tracking jitter is bounded and depends on the parameters of the PLL. The functionality of such generators is based on coherent sampling through which it produces sequences of random bits [64]. In this type of generators, the PLL is expected to ensure the followings:

- having a random jitter exclusively coming from internal electronic components of the PLL, especially the VCO;
- having a rational relationship between input and output frequencies, that is guaranteed by the locking property of the PLL in such way that:

$$f_{clj} = \frac{K_M}{K_D} f_{clk}, \tag{4.1}$$

  where K_M and K_D are respectively multiplication and divisor factors of the PLL[1].

As depicted in Figure 4.1, a general PLL-based TRNG is made of a PLL, a D-flip flop (DFF) and a 1-bit counter. To generate random numbers, the output clock signal clj of the PLL is sampled via the D input of DFF, using a reference clock signal clk. The output of the flip-flop is then directed to the ena input of the 1-bit counter. Its role, as its name indicates, is to count samples for which the logic value is 1 for a specified time period[2] $T_Q := K_D \times T_{clk}$, where T_{clk} is the period of the

[1] Note that the design of a PLL-based TRNG does not consider the fractional mode of PLL operation, in which multiplication and division factors can take real values, since signals generated in this mode can feature a deterministic jitter caused by the operation of the Delta sigma modulator [188, 189].

[2] T_Q is the period of the regular pattern, which would appear at the sampler's output without the presence of noises.

signal clk. The PLL property expressed in Equation (4.1) yields:

$$T_Q = K_D \times T_{clk} = K_M \times T_{clj}. \tag{4.2}$$

The final value of the counter is the random output bit of the generator. The counter value is then reset to 0 for a new count cycle of K_D reference clock periods. One should pay attention to the fact that a counting period equal to integer multiples of K_D reference clock periods causes the appearance of a repetitive pattern that should be avoided.

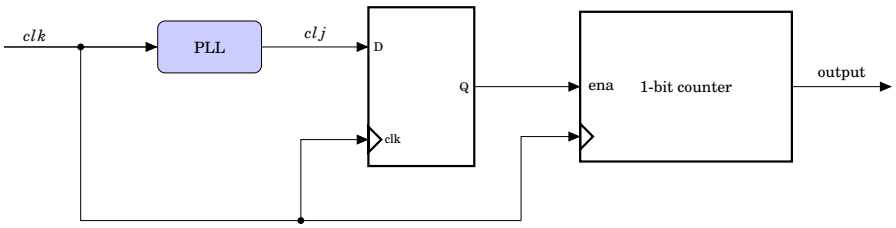


Figure 4.1: PLL-based TRNG.

An illustration of this situation is shown in Figure 4.2. There, one can see an example of input/output timing diagrams of the PLL-based generator in which the multiplication factor is $K_M = 5$ and the division factor is $K_D = 7$. The consequence of such values for K_M and K_D is that the input clock clk displays 7 rising edges while the output clock signal has 5 for a time period $T_Q = K_D \times T_{clk}$. Analysis of these diagrams yields the followings:

- two of the rising edges of clk occur when clj is in the high state (samples 3 and 6),
- two other rising edges of clk occur when clj is in a low state (samples 1 and 4),
- the three other rising edges occur when clj is in an instable region, where there is a nonzero probability of it being either at a high or low state (samples 0, 2 and 5).

The positions of the seven samples is repeated for each new period T_Q causing the appearance of a repetitive pattern with few unstable bits at the output of the D flip-flop. This pattern can be removed by XOR-ing K_D samples in the decimator, which is equivalent to taking the least significant bit of the counter value [82]. Through this operation, one can guarantee a pattern-free random output when using a PLL-based TRNG.

Thanks to this principle and the operation of the PLL, one can easily perform a coherent sampling of the signal clj, enabling to observe the tracking jitter through waveform reconstruction as illustrated in Figure 4.3. In the case where K_M and K_D relatively prime, this reconstruction is done by reordering the K_D samples obtained during the time period T_Q, based on the following

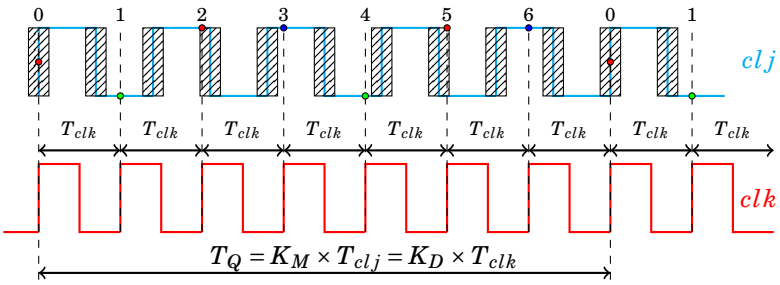


Figure 4.2: Example of input/output signals diagram in a PLL ($K_M = 5$ and $K_D = 7$).

equation [92]:

$$j \;=\; (i \times K_M) \mod K_D, \tag{4.3}$$

where i is time index of the sample, and j is its index in the reconstructed waveform.

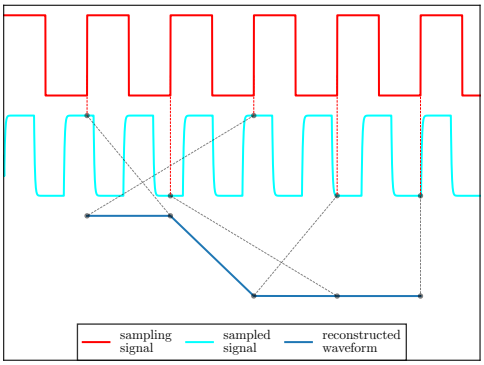


Figure 4.3: Waveform reconstruction by coherent sampling ($K_M = 5$ and $K_D = 7$).

When K_D is odd, in addition of being relatively prime with K_M, it is possible to define the generator bit rate as [82]:

$$R \;:=\; T_Q^{-1} \;=\; \frac{f_{clk}}{K_D}, \tag{4.4}$$

and its sensitivity to jitter as:

$$S \;:=\; f_{clj} \times K_D \;=\; f_{clk} \times K_M. \tag{4.5}$$

The sensitivity to jitter expresses how likely a sample will be affected by the jitter. It characterizes the jitter and parameters of the generator. It therefore has a direct impact on the entropy rate per bit at the generator output [66]. Since the PLL-based generator works with the principle of consistent sampling, the samples are evenly spaced. So we can define the distance between two consecutive samples as:

$$\Delta \;:=\; \frac{T_{clj}}{K_D} \;=\; \frac{1}{f_{clj} \times K_D}. \tag{4.6}$$

From equations (4.5) and (4.6), one can deduce that this distance is inversely proportional to the sensitivity to the jitter.

In order to obtain random numbers at the output of a PLL-TRNG, it is necessary that at least one sample is affected by the jitter. This requires that the distance between any edge of clk and its corresponding edge on clj is less than Δ. This condition is met if the following condition holds [71]:

$$\sigma_{jit} > \max(\Delta T_{min}), \tag{4.7}$$

where σ_{jit} is the standard deviation of the jitter at the output of the PLL, and $\max(\Delta T_{min})$ is the largest distance between the two closest edges of both clk and clj. This can be computed as [71]:

$$\max(\Delta T_{min}) = \frac{T_{clk}}{4K_M}\gcd(2K_M, K_D) = \frac{T_{clj}}{4K_D}\gcd(2K_M, K_D), \tag{4.8}$$

where gcd is the greatest common divisor of two integers.

When designing a TRNG based on PLLs, it is therefore necessary to take into consideration the requirement expressed by Equation 4.7. Indeed, it sets the condition under which one can expect that each generated bit is random. However, some technologies make this condition hard to meet [71], thus compromising the security of cryptographic constructions. For this reason, the principle was adapted to the use of two PLLs rather than just one.

In order to design a generator using two PLLs, it is possible to connect these PLLs in series (as shown in Figure 4.4) or in parallel (as shown in Figure 4.5). The use of this design allows Equation (4.7) to be verified on most hardware technologies. In addition, it significantly increases the throughput and sensitivity to the jitter of the generator. It offers more flexibility to designers for the choice of the multiplication and division factors of the PLLs.

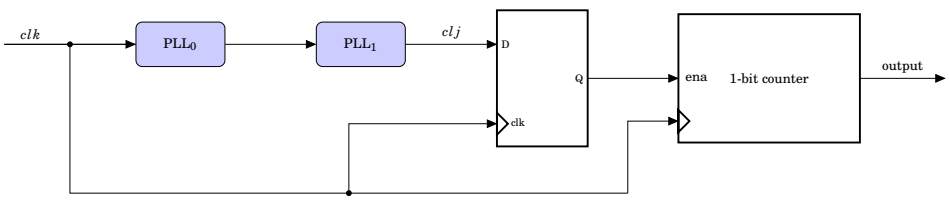


Figure 4.4: PLL-TRNG with two PLLs in series.

As in the case of TRNG based on one PLL, multiplication and division coefficients can be defined for TRNGs based on two PLLs. For consistency, they are respectively denoted as K_M and K_D. These coefficients depend on individual coefficients K_{M_i} and K_{D_i} of each PLL, but also on the

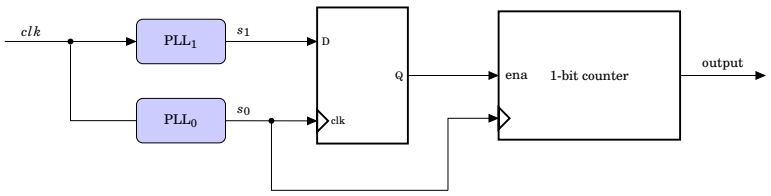


Figure 4.5: PLL-TRNG with two PLLs in parallel.

arrangement of the PLLs. Thus, for a series arrangement as in Figure 4.4, we have:

$$K_M = K_{M_0} \cdot K_{M_1}, \tag{4.9}$$

and

$$K_D = K_{D_0} \cdot K_{D_1}, \tag{4.10}$$

whereas for a parallel arrangement as in Figure 4.5, we have:

$$K_M = K_{M_1} \cdot K_{D_0}, \tag{4.11}$$

and

$$K_D = K_{M_0} \cdot K_{D_1}. \tag{4.12}$$

Although both cases (PLLs in series or in parallel) yield an increase of the K_M and K_D coefficients, they greatly differ in terms of the variance of the resulting jitter [71]. For a serial connection, the jitter introduced by PLL_0 is filtered by PLL_1, which is clearly not the case in the connection of the two PLLs in parallel. Therefore, the parallel configuration produces a larger relative jitter between the output signal of the two PLLs. For this reason, the parallel configuration of a PLL-based generator is preferred to its serial version. Therefore, we will only consider the design of PLL-based TRNGs that use two parallel PLLs.

4.2 Illustration of the DGA-MI approach on PLL-based TRNG

Within the framework of the randomness task group[3], DGA-MI proposed an approach for the evaluation of TRNGs. This approach is essentially based on the AIS-31, but aims to take the analysis of the generator components much further than AIS-31 does. The goal of this approach is not to replace AIS-31, but rather to complement it for the evaluation of systems with high security requirements, such as military and governmental systems.

In order to ensure the feasibility of this approach, David Lubicz illustrated the principle of the elementary generator based on ring oscillators. However, not all state-of-the-art generators use

[3]Group working on the security of true random number generators. It includes the DGA-MI and the Hubert Curien Laboratory among its members.

ring oscillators. To avoid imposing a particular type of generator, and thus limiting the choice of designers, it is therefore necessary to show that this approach can also be applied to another generator principle. In this section, we will illustrate the approach by the generator based on PLLs.

4.2.1 Entities of a generator

From the point of view of the DGA-MI, a TRNG is a physical device characterized by the internal state E of its noise source (which is a function of time with values in phase space V) and which produces binary sequences whose term values are determined by the knowledge of the internal state E. This definition of the TRNG implies that any change in its internal state affects the overall behavior of the generator.

In the case of the PLL-based generator, the phase space can be considered as $V = [0, T_1)$ [92]. The internal state is then the function of the time variable t, defined by:

$$E(t) \;=\; t + \varphi_0 \mod T_1, \tag{4.13}$$

where φ_0 is the initial phase between the signals s_0 and s_1 in Figure 4.5.

In this approach, an TRNG can be decomposed into four parts called generator entities. Each of these entities will have to be evaluated on the basis of requirements provided by this method. It is important to remember that this evaluation can only be done by a competent institution, approved by a certification body, ANSSI for example. Note that it is possible to end up in a situation where each entity of the TRNG is respectively evaluated by a different institution. In order to avoid language discrepancies, the DGA-MI has proposed a vocabulary to be used by the different institutions involved in the evaluation of the generator. We briefly present these four entities while remaining faithful to the vocabulary used by DGA-MI.

4.2.1.1 Physical noise source

Any device producing a noisy signal based on a random physical phenomenon is called a noise sub-source. The physical noise source of a TRNG is the set of all noise sub-sources present in that generator. Since the purpose of a TRNG is to produce unpredictable sequence of bits, it follows that the operating principle of a random number generator must be based on the signal produced by the analog noise source. Therefore, an incorrect identification of noise sub-sources would prevent the signal produced by the noise source from being properly characterized. This can weaken the security of the generator, hence the importance of identifying the different secondary noise

sources, as well as the signals produced by each of its secondary sources, as required by the DGA-MI.

In the case of a generator based on PLLs as shown in Figure 4.5, each PLL is a noise sub-source. Although it is possible to consider the reference signal as coming from a crystal oscillator, the oscillator producing the reference signal is also a noise sub-source. The physical noise source therefore consists of the two PLLs and the oscillator producing the reference signal. Remember that the use of a single reference signal for the two PLLs is imposed by the differential principle in order to minimize the influence of global phenomena on generated bits.

4.2.1.2 Randomness harvester

The randomness harvester, as we saw in Section 1.1.2.2, is the generator entity responsible for transforming the analog signal of the physical noise source into a binary signal with the maximum entropy rate per bit. Its principle of operation is deterministic and should not contain any shadow area. Therefore, the DGA-MI approach requires that the components of the randomness harvester are identified.

In the case of the TRNG based on PLLs, the randomness harvester consists of the D flip-flop and the 1-bit counter.

4.2.1.3 Post-processing block

The post-processing consists in the implementation of an algorithm that does not reduce the entropy rate per bit. It is located after the randomness harvester and aims at improving the statistical quality of the bits produced by the generator. Its presence is not mandatory, because if the statistical properties of the bitstream are satisfactory, the post-processing block can be ignored. However, if it is present, it is mandatory to identify it.

The PLL-based TRNG does not use any post-processing block.

4.2.1.4 Embedded tests

Embedded tests are algorithms implemented in the hardware, whose purpose is to ensure the proper functioning of the generator. These tests can be divided into two categories:

- start-up tests, which are carried out when the generator is started;
- online tests, which continuously monitor the behavior of the generator during its operation.

These tests are compulsory, and it is required that they are identified by the designers of the TRNGs.

It is possible to design several embedded tests (at startup, as well as online) for the PLL-based generator. We will detail them in Sections 4.2.2 and 4.2.3.

4.2.2 Evaluation of the physical noise source

In order to assess the physical noise source, it is necessary to identify the physical phenomena that cause the random behavior of the generated sequences of bits. A wrong identification, or unverified hypothesis on the composition of these phenomena could have bad consequences on the security of the generator, and thus on the security of cryptographic constructions using this generator. Given this observation, the DGA-MI requires that the various physical phenomena responsible for the random nature of the output of TRNGs are identified.

In the case of the generator based on PLLs, we need to use a very stable reference signal, such as that from a crystal oscillator. This will avoid low-frequency fluctuations of the input signal to be reflected in the PLL output signal as detailed in Section 3.4.2.3. With this condition, we can ensure that phase fluctuations in the output signal of the PLL come from the VCO. The accumulation time τ should not exceed τ_n in order to avoid having at the output of the PLL a denatured signal that does not necessarily respect the hypotheses and is difficult to characterize (see Section 3.4.2). When designing the PLL, its parameters should be chosen such that the condition $\tau_n < \tau_c$ is met. Proceeding that way, Section 3.4.2.3 shows that flicker noise present in the VCO will not appear at the output of the PLL. Under the above-mentioned conditions, it is reasonable to consider that the thermal noise from the VCOs of the two PLLs are the physical phenomena responsible for the randomness of the generator's output.

The identification of the phenomena occurring in the noise source is not sufficient to characterize and assess this source. A stochastic model describing the evolution of the internal state E is also needed. As stated in the DGA-MI approach, this model is a function of the time variable t, with values in phase space V. Because it is supposed to take into account the various information available to an observer, it is expressed in the form of a conditional probability similar to:

$$\mathbf{P}\big(E(t)|p_1,\cdots,p_n,E(t_0)\big), \tag{4.14}$$

where $t > t_0$, and $p_1, p_2, \cdots, p_n$ are the parameters of the physical noise source and $E(t_0)$ represents the initial configuration of the internal state. The parameters p_i correspond to the description of the physical environment of the generator (temperature, power supply, etc), but also to the

physical properties of the chip on which the generator is integrated.

The physical parameters of the PLL-based generator are the initial phase φ_0, the standard deviation σ of the white noise coming from the VCO and the duty cycle α of the PLL output signal.

According to the DGA approach, it is not excluded that these parameters and $E(t_0)$ are known to the adversary. However, it is important that he cannot manipulate them beyond certain limits that ensure proper operation of the generator. It may therefore be possible to design an online test detecting abnormal manipulations of these parameters.

Note that in the case of the generator based on PLLs, an abnormal manipulation of these parameters would yield too high phase fluctuations for the PLL to synchronize with the input signal. This would automatically lead to a loss of the PLL lock as discussed in Section 3.1.1. An online test checking whether the PLLs are locked would therefore prevent the opponent from compromising the security of the generator, since a loss of lock would generate an alarm.

The model of the physical noise source, denoted $M(t, p_1, p_2, \cdots, p_n)$, requires a detailed study and understanding of the phenomena occurring in the physical noise source. Therefore, the physical noise source model is the only way to ensure that the generator behaves as expected and provides the required entropy rate. This explains why DGA-MI requires the availability of a stochastic physical noise source model.

This requirement of the DGA-MI approach is also met by the PLL-based generator. Indeed, Bernard *et al.* have proposed a model of the physical noise source of the PLL-based generator [92]. Moreover, the PLL modeling in terms of transfer function carried out in Chapter 3 provides a way to simulate the evolution of the PLL output noise over time. Based on the model of Bernard *et al.* and results from Chapter 3, we can say that noise at the output of the PLL is a bounded random walk of parameters $p_1 = \varphi_0, p_2 = \sigma, p_3 = \alpha$ and $p_4 = b$, where b is the bound on the noise amplitude at the output of the the PLL. The physical model of a PLL-based TRNG can then be summarized as:

$$M(t, p_1, p_2, p_3, p_4) = \mathcal{N}\left(\Delta t, \min\left(b, \sqrt{\Delta t}\sigma\right)\right). \tag{4.15}$$

In order to use the model $M(t, p_1, p_2, ..., p_n)$, it is necessary to provide the values of the physical parameters. The DGA-MI therefore requires the possibility that these parameters can be evaluated experimentally, while specifying the errors committed for each estimate.

To evaluate these parameters, we privilege embedded methods. Indeed, the physical parameters of the PLL-TRNG are:

- the initial phase φ_0, which is quite difficult to evaluate;
- the standard deviation σ of thermal noise from the VCO, which can be related to the number of unstable samples;
- the duty cycle α, which can be approximated by the ratio of the number of samples that are in the logical state 1 over K_D,
- the bound b which remains to be mathematically expressed appears to be constant for the PLLs used.

Thus, apart from the initial phase, it is possible to evaluate the physical parameters using embedded methods. In order to overcome the difficulty raised by the evaluation of φ_0, it is possible to adopt a conservative approach. This approach consists in taking the value of φ_0 which leads to the smallest entropy. An estimation based on this value will then ensure that the required entropy rate is met regardless of the value of φ_0. And since the model gives us $\varphi_0 = \frac{\Delta}{2}$ as the value associated to the smallest entropy, we can ignore the real value of φ_0 without compromising the security of the generator.

In view of the importance of the physical parameters of the noise source, and the impact of their values on the estimation of the entropy rate, the DGA-MI requires that their stability must be evaluated in relation to the:

- physical operating conditions of the generator: temperature, power supply, electromagnetic radiation, etc;
- technological environment of the generator: type of the chip, generator isolated or in the presence of other modules, etc;
- aging effects.

The technical documentation of the PLL guarantees good parameter stability over a wide temperature range (between -55 and $125\,^{\circ}$C) [190]. By default, the PLLs are isolated from the other logic elements of the board in order to reduce their impact on the operation of the PLL. It is therefore reasonable to assume that the PLL would always be isolated on the chip. These elements suggest that the various physical parameters of the source would be stable under the PLL's operating conditions. However, we did not investigate this aspect further. More in-depth studies are thus to be planned for the future in order to confirm these conjectures.

4.2.3 Evaluation of the randomness harvester

Since the randomness harvester converts the phases of the noise source, we can say that it is a deterministic function of the internal state of the generator. Therefore, it should be possible to describe its principle of operation and establish a model of the generator (including the noise source and the randomness harvester). Such a model, according to the DGA-MI approach, is a stochastic model $N(t, p_1, p_2, \cdots, p_n, q_1, q_2, \cdots, q_m)$ with values in the set of binary sequences of arbitrary lengths. The parameters $t, p_1, p_2, \cdots, p_n$ are those of the model $M(t, p_1, p_2, ..., p_n)$ of the physical noise source, while $q_1, q_2, \cdots, q_m$ are the generator parameters. It is permitted for some of the parameters q_i to be adjustable when designing the generator, however, they should not be manipulable by an adversary.

In the case of a generator based on the PLLs, the parameters of the generator are the multiplication and division factors K_M and K_D.

Since a generator that meets the various requirements mentioned above can be described by a stochastic model, DGA-MI demands that TRNG designers must provide the stochastic model of the generator.

In the case of a generator based on PLLs, the stochastic model describing the probable output of the logic level X_i, of the bit at the output of the entropy collector at time $i \times T_0$, is [92]:

$$\mathbf{P}(X_i = 1) \;=\; \mathbf{P}(\varphi_i < \alpha T_1) - \mathbf{P}(\varphi_i < 0) + 1 - \mathbf{P}(\varphi_i < T_1). \tag{4.16}$$

To reduce the statistical defects at the generator output (bias, correlation, etc), DGA-MI requires that the parameters q_i of the TRNG model must be adjustable in order to have the best possible configurations.

As mentioned above, these parameters in the case of a PLL-TRNG are K_M and K_D. They depend on the multiplication and division coefficients M, N, C and P_vco of the two PLLs. These coefficients, as we saw in Chapter 3, are adjustable. As a consequence, the parameters K_M and K_D are also adjustable. According to Section 3.4.2.4, an optimal value of K_M should be greater than 138 in order to increase the damping factor and thus lower the jitter peaking. Since K_M is proportional to M_0 and M_1 according to Equation (4.11), it follows that at least M_0 or M_1 should be large. This carries a non-negligible risk of integrating flicker noise. A compromise should therefore be sought in order to have a high damping factor and a low risk of flicker noise integration. Furthermore, the differential principle requires that the two PLLs have similar configurations, *i.e.* $M_0 \simeq M_1$, $N_0 \simeq N_1$ and $C_0 \simeq C_1$. All these constraints show that the search for optimal parameters is not

an easy task. Fortunately, TRNGs based on PLLs have methods for finding these parameters, making it easier for designers. We will discuss this in more detail in Section 4.3.

To make sure that values of these parameters are in the region of sufficient entropy, tests can be developed to check their values. These types of tests are referred to as parametric tests in the DGA-MI evaluation approach, which requires that these tests must be run at generator start-up and during operation.

This requirement stipulates that the parametric tests must monitor the entropy source to ensure sufficient entropy at the generator output. In the case of the generator based on PLLs, the stochastic model developed by Bernard *et al.* allows the entropy to be expressed as an increasing and continuous function of the variance σ^2 of the jitter. Thus, for a minimum entropy threshold H_{min} (of 0.997 as required by the AIS-31 for example), it is possible to determine the value σ^2_{min} of the variance corresponding to the required entropy threshold. It is then possible to design an embedded test that regularly computes the variance σ^2 and compares it with σ^2_{min}. A value of σ^2 less than σ^2_{min} would then indicate insufficient entropy and generate an alarm.

In addition to parametric tests, it is possible to consider tests verifying the integrity of the randomness harvester. Tests in this category are called deterministic tests in the DGA-MI approach. This name is justified by the fact that the randomness harvester associates in a deterministic way the internal state of the entropy source with the output bit of the generator. The DGA-MI requires that such tests are available to ensure the proper functioning of the randomness harvester.

In the case of the PLL-based generator, it is possible to validate the behavior of the randomness harvester by feeding it with known binary sequences and checking whether the output corresponds to what is expected. Of course, this test cannot be done while the generator is running. Instead, it can be set up to start at startup using a multiplexer. In this scenario, the known sequence will be sent, and in case the test validates the operation of the randomness harvester, the multiplexer will switch to the signals coming from the PLLs.

4.3 Optimal configurations for a PLL-based TRNG

From discussions made in Chapter 3 and Section 4.1, one can see that a TRNG based on PLLs is an interesting construction for guaranteeing production of genuinely random numbers. However, bringing it to good performance is a hard task, mainly because of its large number of parameters to tune. Indeed, finding a suitable set of parameters that satisfy the throughput and entropy

requirements for a given application can easily become very complicated. To achieve this goal, we suggest in this section, a bounded exhaustive search procedure that determines the parameters that ensure optimal performance of a PLL-based TRNG. Compared to a prior method, based on genetic algorithms [191], the proposed method has several advantages that will be discussed below.

4.3.1 Statement of the problem

In the design process of a PLL-based TRNG, three groups of constraints should be identified and respected. The first group is related to physical constraints of the PLL, namely minimum and maximum frequencies for the inputs and outputs of its functional blocks, and allowed intervals of division factors [188, 189]. The second group of constraints concerns settings of the TRNG [82]. The third group is related to the system requirements, namely the minimum throughput and the entropy [66].

We know from Section 4.1, that the operation of a PLL-TRNG is based on the coherent sampling principle. Application of this principle is guaranteed by the operation of the PLL which links the input frequency to its output frequency as specified in Equation (4.1), with K_M and K_D coprime. The value of the frequency multiplication and division factors K_M and K_D highly depends on the PLL configurations. Hence, their values are different for a TRNG based on one PLL and a TRNG based on two PLLs.

The objective of the designer is to set the parameters of the PLL(s) in order to obtain (depending on application requirements) sufficient entropy and bit rate at the generator output. When setting the parameters of the PLL, the designer must also fulfill hardware requirements of the hardwired PLL circuitry.

4.3.1.1 General structure of the PLL and its configuration

We recall that the general structure of a PLL can be summarized as a block diagram depicted in Figure 4.6. Blue blocks cannot be parameterized, as discussed in Chapter 3. They are adjusted once and for all during the design of the PLL according to the desired noise properties. However, the M, N, C and P_{vco} are the multiplication and division coefficients, in white blocks, that are parameterized by the TRNG designer. This therefore enables to configure the PLL by setting these coefficients while strictly respecting relationships between individual blocks. These relationships specify permitted ranges of their input/output frequencies and permitted ranges of coefficients, which are specified in the technical documentation of the PLL block [188].

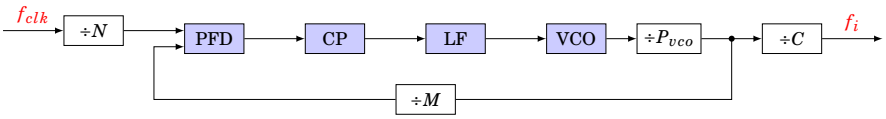


Figure 4.6: PLL block diagram (PFD: phase-frequency detector, CP: charge pump, LF: loop filter, VCO: voltage-controlled oscillator, f_{clk}: input frequency, f_i: output frequency of PLL$_i$, $i = 0, 1$).

A set of values of these multiplication and division coefficients constitute what we call a PLL configuration. Thus, finding an optimal PLL configuration is equivalent to finding values of M, N, C and P_{vco} that guarantee optimal operation of a PLL-based TRNG. Because each PLL has its own multiplication and division coefficients (if two PLLs are used), we will designate in the following by M_i, N_i and C_i those related to PLL$_i$. Thus, the multiplication and division factors of PLL$_i$ will be written:

$$K_{M_i} = M_i \quad \text{and} \quad K_{D_i} = N_i \cdot C_i. \tag{4.17}$$

When considering the architecture of a PLL-based TRNG using two PLLs as in Figure 4.5, the multiplication and division factors are:

$$K_M = K_{M_1} \cdot K_{D_0} = M_1 \cdot N_0 \cdot C_0, \tag{4.18}$$

and

$$K_D = K_{M_0} \cdot K_{D_1} = M_0 \cdot N_1 \cdot C_1. \tag{4.19}$$

Because s_0 is the sampling signal in the architecture of Figure 4.5, the throughput (expressed in Mbit $\cdot$ s^{-1}) can be computed as[4]:

$$R = \frac{f_0}{K_D} = \frac{f_{clk}}{N_0 \times C_0 \times N_1 \times C_1}, \tag{4.20}$$

since:

$$f_i = \frac{M_i}{N_i \cdot C_i} f_{clk}, \tag{4.21}$$

for $i \in \{0, 1\}$. Likewise, the sensitivity to jitter (expressed in μs^{-1}), can be computed as[5]:

$$S = K_D \times f_1 = M_0 \times M_1 \times f_{clk}, \tag{4.22}$$

where frequencies are expressed in MHz.

[4] It can be noted from the Equation (4.20) that the throughput depends only on the division factors of the two PLLs. Thus, in order to have high throughput, it is necessary that these factors are as low as possible. On the other hand, taking small division factors would greatly reduce the frequency range of the input signal. In the case where the designer does not have much freedom in the choice of the input frequency, certain constraints would have to be loosened in order to ensure the proper functioning of the PLL.

[5] Equation (4.22) shows that the larger M_0 and M_1 are, the higher the sensitivity to jitter is. This results in a higher entropy rate per bit at the output of the generator. On the other hand, Section 3.4.2.4 shows that the jitter peaking in the neighborhood of the natural frequency of the PLL decreases as M increases. It is therefore tempting to increase the values of M_0 and M_1. However, Section 3.4.2.4 shows that for large values of M_1 and M_2, the risk of integrating flicker noise is greater. It is therefore necessary to find a trade-off on the values of M_1 and M_2 in order to have the best entropy rate per bit while keeping the risk of integrating flicker noise low.

In addition, for each PLL, the frequencies of the internal signals are related to each other. Thus, for PLL i, one can express the output frequency f_{PFD} of the phase-frequency detector as:

$$f_{PFD_i} = \frac{f_{clk}}{N_i}. \tag{4.23}$$

The output frequency of the VCO is related to that of the PFD by:

$$f_{VCO_i} = M_i \times P_{VCO_i} \times f_{PFD_i}, \tag{4.24}$$

and the output frequency is:

$$f_i = \frac{M_i}{C_i} \times f_{PFD_i}. \tag{4.25}$$

Knowing the relationships between PLL configurations and generator constraints, we can now state the problem to be solved.

4.3.1.2 Problem to solve

When designing TRNGs, one of the main challenges faced by designers is the selection of feasible configurations in the hardware for which security and performance constraints are met. In other words, it means finding:

- all the feasible configurations[6] of the two PLLs;
- among feasible configurations, those that satisfy security (*e.g.* entropy rate evaluated from the model as specified in AIS-31 recommendations [24]) and/or performance of the target application.

The first step in this process is to find values of M_i,N_i,C_i satisfying Equations (4.18) to (4.25), that are constrained by the following inequalities:

$$\begin{cases} M_{min} \leqslant M_i \leqslant M_{max} & (4.26) \\ N_{min} \leqslant N_i \leqslant N_{max} & (4.27) \\ C_{min} \leqslant C_i \leqslant C_{max} & (4.28) \\ f_{PFD_{min}} \leqslant f_{PFD_i} \leqslant f_{PFD_{max}} & (4.29) \\ f_{VCO_{min}} \leqslant f_{VCO_i} \leqslant f_{VCO_{max}} & (4.30) \\ f_{out_{min}} \leqslant f_{out_i} \leqslant f_{out_{max}}, & (4.31) \end{cases}$$

where M_{min}, N_{min}, C_{min}, M_{max}, N_{max} and C_{max} are positive integers, and $f_{PFD_{min}}$, $f_{PFD_{max}}$, $f_{VCO_{min}}$, $f_{VCO_{max}}$, $f_{out_{min}}$ and $f_{out_{max}}$ are positive real numbers, which represent hardware limitations of the PLL given by manufacturers.

[6] Configurations that meet frequency ranges of individual PLL blocks. They therefore fulfill hardware constraints of the technical documentation of the PLL.

In our experiments, we considered PLL-based TRNGs implemented in three different FPGA families: Intel Cyclone V [188], Xilinx Spartan-6 [189] and Microsemi SmartFusion®2 FPGAs [192]. Table 4.1 gives the hardware restrictions for the selected FPGA families.

Parameter	Cyclone V		Spartan-6		SmartFusion®2	
	Min	Max	Min	Max	Min	Max
f_{clk} (MHz)	5	500	19	540	1	200
P_{VCO_i}	1	2	1	1	1	32
N_i	1	512	1	52	1	16384
M_i	1	512	1	64	1	4194304
C_i	1	512	1	128	1	255
f_{PFD_i} (MHz)	5	325	19	500	1	200
f_{VCO_i} (MHz)	600	1300	400	1080	500	1000
f_{out_i} (MHz)	0	460	3.125	400	20	400

Table 4.1: PLL specifications of selected FPGA families.

Once the configurations that are feasible in hardware are found, the designer needs to filter out those that do not attain the required entropy rate and/or sufficient throughput depending on the application objectives and constraints, according to Equations (4.20) and (4.22). The stochastic model of the PLL-based TRNGs [92] can be used to provide thresholds for these parameters, which are necessary to obtain the suitable configurations out of all the feasible ones.

4.3.2 Search of PLL-TRNG configurations

The problem of finding PLL-TRNG parameters is not a new one. Indeed, the original article proposing this TRNG architecture adopted an "expert input" choice of parameters for the PLL [82]. However, as its name indicates, this methods requires some level of expertize, with no guarantee to find optimal configurations. This requirement prevents anyone who does not have experience in designing PLL-TRNGs to find suitable parameters. Moreover, due to the size of the parameters space, this approach might give good results but will probably miss better configurations available. Petura *et al.* therefore questioned this "expert input" approach and derive the parameters of the PLL-TRNG using a genetic algorithm [66, 191]. This new method was indeed able to find better configurations than previously known, having better throughput and entropy. However, since the genetic algorithm finds only locally optimal solutions, which depends on starting values of parameters of the genetic algorithm selected randomly, it was never sure that the solution found is the best one.

We have investigated this subject further to find an analytical method. The proposed method consists in deriving, for each parameter, the exact bounds based on physical constraints specified in Section 4.3.1.

4.3.2.1 Search of all feasible configurations

To find all feasible configurations, the first naive approach could be to perform an exhaustive search on the PLL design space and save configurations that satisfy hardware constraints. Due to the huge number of possible configurations, it is not reasonable to process this way. The improvement we suggest consists in a depth-first search strategy, which is basically a sequential search of all values of M_i, N_i and C_i that lead to feasible configurations of the two PLLs.

This method assumes that f_{clk} is given by the designer whose responsibility is to ensure that this value is between $f_{clk_{min}}$ and $f_{clk_{max}}$ defined in Table 4.1. Then, for each value of P_{VCO_i}, we proceed as follows.

1. From Equation (4.23), one can see that if f_{clk} is given, then N_i is the only variable for which the value has to be found. That value can actually be computed as:

$$N_i = \frac{f_{clk}}{f_{PFD_i}}, \tag{4.32}$$

which provides new bounds for N:

$$N_{min_i} \leqslant N_i \leqslant N_{max_i}, \tag{4.33}$$

where:

$$N_{min_i} = \max\left(N_{min}, \left\lceil \frac{f_{clk}}{f_{PFD_{max}}} \right\rceil\right) \tag{4.34}$$

and:

$$N_{max_i} = \min\left(N_{max}, \left\lfloor \frac{f_{clk}}{f_{PFD_{min}}} \right\rfloor\right). \tag{4.35}$$

Note that $N_{max_i} - N_{min_i} \leqslant N_{max} - N_{min}$, showing that the range for searching N_i is reduced.

2. Equations (4.23) and (4.24) allow to express f_{VCO_i} as a function of f_{clk}, N_i and M_i. Hence, for given values of f_{clk} and N_i, it is possible to find all the values of M_i using:

$$M_i = \frac{N_i \cdot f_{VCO_i}}{f_{clk} \cdot P_{VCO_i}}, \tag{4.36}$$

where f_{VCO_i} is the only variable value, yielding new bounds for M_i:

$$M_{min_i} \leqslant M_i \leqslant M_{max_i}, \tag{4.37}$$

where:

$$M_{min_i} = \max\left(M_{min}, \left\lceil \frac{N_i \cdot f_{VCO_{min}}}{f_{clk} \cdot P_{VCO_i}} \right\rceil\right) \tag{4.38}$$

and:

$$M_{max_i} = \min\left(M_{max}, \left\lfloor \frac{N_i \cdot f_{VCO_{max}}}{f_{clk} \cdot P_{VCO_i}} \right\rfloor\right). \tag{4.39}$$

3. In a way similar to the second step, one can assume f_{clk}, N_i and M_i known. Therefore, Equation (4.25) helps to find the values of C_i:

$$C_i = \frac{f_{clk} \cdot M_i}{N_i \cdot f_i}. \tag{4.40}$$

Knowing that all values are given except for f_{out_i} which ranges from $f_{out_{min}}$ to $f_{out_{max}}$, it follows that:

$$C_{min_i} \leqslant C_i \leqslant C_{max_i}, \tag{4.41}$$

where:

$$C_{min_i} = \max\left(C_{min}, \left\lceil \frac{f_{clk} \cdot M_i}{N_i \cdot f_{out_{max}}} \right\rceil\right) \tag{4.42}$$

and:

$$C_{max_i} = \min\left(C_{max}, \left\lfloor \frac{f_{clk} \cdot M_i}{N_i \cdot f_{out_{min}}} \right\rfloor\right). \tag{4.43}$$

Thanks to Inequalities (4.33), (4.37) and (4.41) we are guaranteed to obtain only feasible configurations and all of them in a reasonable amount of time (several hours instead of years). In the context of a PLL-based TRNG, several practical considerations have to be taken into account, yielding in an improvement of the running time.

4.3.2.2 Search of suitable configurations

A feasible configuration for a PLL-based TRNG is one found according to the process described in Section 4.3.2.1. However, some configurations found through this process may not be suitable for a TRNG design, since several considerations related to the TRNG design were not considered. Indeed, for a PLL-TRNG, it is required that K_D is odd and relatively prime with K_M [82].

Even though the coprimality of K_M and K_D has to be checked by the Euclidean algorithm, it is possible to skip the parity test of K_D. The fact that K_D must be odd, associated with Equation (4.19), implies that M_0, N_1 and C_1 should all be odd. Values of M_0, N_1 and C_1 can then be looked for by a step of 2, starting with the smallest odd number in each range. This consideration reduces by a factor of 2 (for each of these parameters) the number of values that have to be checked, and thus speeds up the algorithm.

Furthermore, for security reasons, the sensitivity to the jitter must be sufficiently high. In addition, performance reasons require the throughput to be as high as possible. Thanks to Equations

(4.20) and (4.22), it follows that K_M and K_D must be bounded. In order to give more flexibility to the designer, we introduce two quantities s_M and s_D as their respective upper bounds[7]:

$$0 \leqslant K_M \leqslant s_M \quad \text{and} \quad 0 \leqslant K_D \leqslant s_D. \tag{4.44}$$

Using Equations (4.18) and (4.19), it follows:

$$0 \leqslant C_0 \leqslant \frac{s_M}{M_1 \cdot N_0} \quad \text{and} \quad 0 \leqslant C_1 \leqslant \frac{s_D}{M_0 \cdot N_1}. \tag{4.45}$$

We can thus define smaller upper bounds C'_{max_0}, C'_{max_1} to C_0 and C_1, respectively, by:

$$C'_{max_0} = \left\lfloor \min\left(C_{max}, \frac{f_{clk} \cdot M_0}{N_0 \cdot f_{out_{min}}}, \frac{s_M}{M_1 \cdot N_0}\right)\right\rfloor \tag{4.46}$$

and

$$C'_{max_1} = \left\lfloor \min\left(C_{max}, \frac{f_{clk} \cdot M_1}{N_1 \cdot f_{out_{min}}}, \frac{s_D}{M_0 \cdot N_1}\right)\right\rfloor. \tag{4.47}$$

These new considerations significantly reduce the number of possible configurations of both PLLs. We thus have a smaller subset among all the feasible configurations for the PLL-based TRNGs implemented in the selected FPGA family. The whole search process is summarized in Algorithm 2. Note that the order in which the for loops are nested is carefully chosen so that the depth-first search is performed optimally by computing the bounds in the correct order since they form a chain of dependency.

4.3.3 Experimental results

4.3.3.1 Implementation considerations

To evaluate the speed and efficiency of the search process, according to hardware limitations specified by manufacturers, we implemented our algorithm in C language. The algorithm takes only two inputs: the reference frequency f_{clk} and the FPGA family, for which we want to generate configurations.

We ran the algorithm on an HP Compaq 6005 Pro MT PC AMD Athlon™ II X2 B24 Processor. When fed with a reference frequency of 125 MHz (we selected the same reference frequency for all families in order to get comparable results), Algorithm 2 found all the suitable configurations in less than 10 seconds for each of the above mentioned FPGA families.

It is important to note that the notion of best configuration is application dependent. Indeed, system requirements are generally specified in terms of minimum jitter sensitivity and minimum throughput. These values are given based on the intended security requirements. Therefore, the

[7] In our experiments, we took $s_M = 1000$ and $s_D = 400$.

Algorithm 2 PLL configuration search algorithm for the PLL-TRNG

compute N_{min_0} from Equation (4.34)
compute N_{max_0} from Equation (4.35)
$N_{min_1} \leftarrow$ round_up_to_odd(N_{min_0})
$N_{max_1} \leftarrow N_{max_0}$
`configs` $\leftarrow$ MAKEEMPTYLIST()
for all P_{VCO_0} in Pvco_vals **do**
  for all P_{VCO_1} in Pvco_vals **do**
    for $N_1 = N_{min_1}$ **to** N_{max_1} by 2 **do**
      compute M_{min_1} from Equation (4.38)
      compute M_{max_1} from Equation (4.39)
      for $N_0 = N_{min_0}$ **to** N_{max_0} **do**
        compute M_{min_0} from Equation (4.38)
        $M_{min_0} \leftarrow$ round_up_to_odd$\left(M_{min_0}\right)$
        compute M_{max_0} from Equation (4.39)
        for $M_0 = M_{min_0}$ **to** M_{max_0} by 2 **do**
          compute C_{min_0} from Equation (4.42)
          for $M_1 = M_{min_1}$ **to** M_{max_1} **do**
            compute C_{min_1} from Equation (4.42)
            $C_{min_1} \leftarrow$ round_up_to_odd$\left(C_{min_1}\right)$
            compute C'_{max_0} from Equation (4.46)
            compute C'_{max_1} from Equation (4.47)
            for $C_1 = C_{min_1}$ **to** C'_{max_1} by 2 **do**
              compute K_D from Equation (4.19)
              for $C_0 = C_{min_0}$ **to** C'_{max_0} **do**
                compute K_M from Equation (4.18)
                if gcd$(K_M, K_D) = 1$ **then**
                  compute f_0 and f_1 from Equation (4.25)
                  compute R from Equation (4.20)
                  compute S from Equation (4.22)
                  save into `configs`, values of $f_{clk}, P_{VCO_0}, P_{VCO_1}, M_0, N_0, C_0, M_1, N_1, C_1, f_0, f_1, K_M, K_D, R, S$
                end if
              end for
            end for
          end for
        end for
      end for
    end for
  end for
end for

best configuration of one application may not be the best for another one. This is the main reason why we adopted this approach of providing all suitable configurations meeting the desired requirements.

Moreover, PLLs are primarily used for system clock generation and there is only a limited number of PLLs available in FPGAs. For this reason, designers are sometimes forced to share a PLL between the TRNG and the application. This adds additional bounds for the output signal fre-

quency, which must be taken into account in the algorithm. The timing analysis of the proposed PLL-TRNG design showed a maximum supported clock frequency of around 250MHz on all three tested FPGA families, which represents the limiting frequency of the logic resources in the design. To comply with these limits, we decreased the PLL maximum output frequency from manufacturers' limit to 250MHz in order to exclude the PLL configurations too fast for the circuitry.

The minimum value of the sensitivity to jitter can be determined from the required entropy rate using the stochastic model [92]. For the Shannon entropy rate of 0.997 required by AIS-31 [15], the minimum sensitivity to jitter obtained from the model must be higher than $0.09\,\mathrm{ps}^{-1}$. So we limited our search to configurations satisfying this security condition.

4.3.3.2 Results and discussions

The search returned 188 suitable configurations out of 389853(0.048%) feasible ones for Intel Cyclone V, 8 out of 89025(0.0089%) for Xilinx Spartan-6, and 9976 out of 2339412(0.426%) for SmartFusion®2. These figures show that the number of configurations satisfying security conditions is smaller than 1% of the total number of feasible configurations for every FPGA family tested. This reinforces the belief that manual search is nearly impossible, thereby pointing out the importance of the proposed method.

Experimental results are given in Table 4.2 where, for each FPGA family, three configurations are given. The first one, referred to as "Highest S" is the one with the best sensitivity to jitter for a throughput R of at least $0.5\,\mathrm{Mbit}\cdot\mathrm{s}^{-1}$. The second one referred to as "Highest R" is the one with the best throughput for a sensitivity to jitter S of at least $0.09\,\mathrm{ps}^{-1}$. The third one, referred to as "Highest $R\cdot S$" is the one providing the best compromise between the throughput and the sensitivity to the jitter. For every configuration in this table, we can see that the security requirement $S_{min} > 0.09\,\mathrm{ps}^{-1}$ is always met, ensuring security of the TRNG.

To validate the quality of the chosen configurations provided by this method, we ran the AIS-31 statistical test suite on the output bit sequences of the TRNG. Since the sensitivity limit was chosen according to the stochastic model, it is not surprising that outputs of all the configurations passed the statistical tests successfully.

It is important to note that for certain security and throughput constraints, the method developed could not find configurations. Since the method was developed to find the best configurations in all practical cases, the fact that it cannot find any means that the constraints formulated are not realistic. Indeed, it is impossible to have huge throughput for any security threshold. The

Config.	f_{clk} (MHz)	P_{VCO_0}	P_{VCO_1}	M_0	N_0	C_0	M_1	N_1	C_1	f_0 (MHz)	f_1 (MHz)	K_M	K_D	R (Mbit/s)	S (ps^{-1})	$R \cdot S$
Intel Cyclone V																
Highest S	125	1	1	7	1	4	113	19	3	218.75	247.807	452	399	0.548	0.0988	0.0542
Highest R	125	2	1	43	11	2	17	3	3	244.318	236.111	374	387	0.631	0.0913	0.0577
Highest $R \cdot S$	125	1	1	19	2	5	41	7	3	237.5	244.047	410	399	0.595	0.0973	0.0579
Xilinx Spartan-6																
Highest S, R, and $R \cdot S$	125	1	1	43	5	5	17	3	3	215	236.11	425	387	0.555	0.0913	0.0507
Microsemi SmartFusion®2																
Highest S	125	1	1	7	1	4	113	19	3	218.75	247.807	452	399	0.548	0.098	0.054
Highest R	125	1	4	29	5	3	25	13	1	241.66	240.384	375	377	0.641	0.090	0.058
Highest $R \cdot S$	125	1	4	23	3	4	33	17	1	239.58	242.647	396	391	0.612	0.094	0.058

Table 4.2: Two-PLL-TRNG configurations for sensitivity $S > 0.09$ ps^{-1}.

simulations carried out have shown that the maximum throughput is inversely proportional to the minimum level of security. Thus, the higher the required security level is, the lower the maximum throughput will be.

4.3.3.3 Comparison with the previous method

In order to compare this new method with the one based on the genetic algorithm, we implemented a version of it which uses only one PLL. For each of the three FPGA families, we ran this algorithm with the reference frequency recorded in [191, Table II]. Table 4.3 presents the results we obtained.

	f_{clk} (MHz)	P_{vco}	M	N	C	f_1 (MHz)	K_M	K_D	R (Mbit/s)	S (ps^{-1})
	Intel Cyclone V									
Best configuration from [191]	350	1	131	37	3	413	131	111	3.15	0.045
Configuration found by the proposed method	350	1	136	37	3	428.82	136	111	3.15	**0.047**
	Xilinx Spartan-6									
Best configuration from [191]	430	1	47	21	5	192	47	105	4.095	0.020
Configuration found by the proposed method	430	1	47	21	3	320.79	47	63	**6.82**	0.020
	Microsemi SmartFusion®2									
Best configuration from [191]	200	2	216	127	1	340	216	127	1.574	0.043
Configuration found by the proposed method	200	2	253	127	1	398.425	253	127	1.574	**0.05**

Table 4.3: Comparison of one-PLL-TRNG configurations found by the proposed algorithm with those found using the genetic algorithm.

As one could expect, the proposed algorithm found all the configurations provided by the method based on the genetic algorithm. Moreover, as can be seen in Table 4.3, it also found better configurations ensuring higher throughput and jitter sensitivity. It thus gives the designer a very efficient tool to find the best configuration, which fulfills hardware constraints and satisfies high

security requirements in the given true random number generation context.

The method described here is shown to be an efficient tool generating all suitable PLL-TRNG configurations out of all feasible PLL configurations in a TRNG design based on one or two PLLs. This solves the problem of finding configurations of a PLL-TRNG, in which the published genetic algorithm could not be applied because of a high number of variables. Hence, the use of this new method extends the set of configurations found by the genetic algorithm, providing higher throughput and jitter sensitivity. Furthermore, this method is fast enough to generate all suitable configurations within seconds for both one-PLL and two-PLL designs. This approach guarantees the global optimality of the results found since all feasible configurations are obtained and all suitable configurations are selected from them according to the specific constraints of the application. It can also be used alongside with a stochastic model to update the entropy requirements.

4.4 Conclusion

In this chapter, we have presented the general functioning of a TRNG based on PLLs. The fact that the PLL is first used to synthesize a clock signal containing a very low level of jitter may at first sight be a limitation. However, the PLL offers the opportunity to adjust the components of the output jitter by a careful choice of its parameters. This PLL functionality thus makes it possible to guarantee a jitter made up of the desired sources of noise, something that is not possible to do with other sources of randomness such as ROs. Furthermore, its physical and electrical isolation from the rest of the FPGA logic makes it possible to guarantee a low sensitivity to environmental phenomena occurring on the chip. Moreover, its presence in most FPGA families helps to reduce the cost of producing TRNGs based on PLLs.

We also briefly presented an approach to evaluate the security of TRNGs. This approach proposed by DGA-MI is meant to be a complement to AIS-31 for the evaluation of generators intended for applications with very high security requirements. In previous work, Lubicz has established the implementability of this approach by illustrating it with the principle based on ROs. Following the recommendation of DGA-MI, we have illustrated this approach with the PLL-based principle. This proves not only that this approach is not restrictive from a design point of view, but also that the PLL-based principle meets the requirements for very high levels of security.

This last evaluation approach raised the problem of finding parameters to ensure optimal generator operation. In the case of generators based on PLLs, this means looking for PLL parameters that guarantee an entropy that meets the security criteria and a sufficient throughput for a better

generator performance. Work prior to the study in this chapter shows that this is a difficult problem, particularly because of the large number of possible parameters, making it almost impossible to search manually. To this difficulty are added numerous constraints related to the operation of the PLL, preventing previous methods from adapting to TRNGs based on two PLLs because they are more complex. To solve this issue, we adopted an exhaustive depth-first search strategy associated to new constraints obtained from those provided by PLL manufacturers. This solution provides PLL configurations for TRNGs based on either one or two PLLs, but also ensures optimality of the configurations. This method, as well as the results obtained, have been published at DATE 2018[8].

[8] E. Noumon Allini, O. Petura, V. Fischer, F. Bernard, "Optimization of the PLL configuration in a PLL-based TRNG design", DATE 2018: 1265-1270

Résumé

Dans ce chapitre, nous avons présenté le fonctionnement général d'un TRNG basé sur les PLLs. Le fait que la PLL soit prinicpalement utilisée pour synthétiser un signal d'horloge contenant un très faible niveau de jitter peut, à première vue, constituer une limitation. Cependant, la PLL offre la possibilité d'ajuster les composantes du jitter de sortie par un choix judicieux de ses paramètres. Cette fonctionnalité de la PLL permet donc de garantir jitter composée des sources de bruit souhaitées, ce qui n'est pas possible avec d'autres sources d'aléa tel que les ROs. De plus, son isolation physique et électrique du reste de la logique du FPGA permet de garantir une faible sensibilité aux phénomènes environnementaux se produisant sur la carte. De plus, sa présence dans la plupart des familles de FPGA permet de réduire le coût de production des TRNG basés sur les PLL.

Nous avons également présenté brièvement une approche pour évaluer la sécurité des TRNGs. Cette approche proposée par la DGA-MI se veut être un complément à l'AIS-31 pour l'évaluation des générateurs destinés à des applications ayant des exigences de sécurité très élevées. Dans des travaux antérieurs à cette thèse, Lubicz a établi la faisabilité de cette approche en l'illustrant par le principe basé sur les ROs. Suite aux recommandations de la DGA-MI, nous avons illustré cette approche avec le principe basé sur les PLLs. Cette illustration a prouvé non seulement que cette approche n'est pas restrictive d'un point de vue conception, mais aussi que le principe basé sur les PLLs répond aux exigences de très hauts niveaux de sécurité.

Cette dernière approche d'évaluation a soulevé le problème de recherche des paramètres assurant un fonctionnement optimal du générateur. Dans le cas des générateurs basés sur les PLLs, cela revient à trouver les paramètres des PLL qui garantissent une entropie répondant aux critères de sécurité et un débit suffisant. Les travaux préalables à l'étude présentée dans ce chapitre montrent qu'il s'agit d'un problème difficile, notamment en raison du grand nombre de paramètres possibles, ce qui rend la recherche exhaustive presque impossible. À cette difficulté s'ajoutent de nombreuses contraintes liées au fonctionnement de la PLL, empêchant les méthodes précédentes de s'adapter aux TRNG basés sur deux PLL car elles sont plus complexes.

Afin de résoudre ce problème, nous avons adopté une recherche exhaustive basée sur la stratégie de recherche en profondeur, associée à de nouvelles contraintes obtenues à partir de celles fournies par les fabricants des PLLs. Les résultats expérimentaux et théoriques montrent que cette nouvelle méthode permet d'avoir des configurations de PLL pour les TRNGs basées sur une PLL, comme ceux basés sur deux PLLs. Elle assure également l'optimalité des configurations trou-

vées. Ces résultats constituent une avancée, en comparaison aux méthodes antérieures qui ne s'appliquaient que sur les constructions basées sur une seule PLL. En outre, les méthodes antérieures n'assuraient pas l'optimalité des configurations trouvées. Cette méthode, ainsi que les résultats obtenus ont été publiés à la DATE 2018[9].

[9] E. Noumon Allini, O. Petura, V. Fischer, F. Bernard, "Optimization of the PLL configuration in a PLL-based TRNG design", DATE 2018: 1265-1270

Contributions

The work carried out in the course of this PhD has led to the following contributions.

- We have shown that generating random numbers using counter values is more efficient than the sampling-based method. It improves the throughput of TRNGs, while serving as a basis for online tests.
- We have drawn attention to the risks associated with the use of classical variance to characterize jitter. This risk comes from the low-frequency components of jitter that lead to a divergence in the behavior of the classical variance and lead to results that cannot be interpreted.
- We have proposed the use of Allan variance (and its variants) as an alternative solution to characterize phase and frequency fluctuations. These variances, widely used in the timekeeping domain, have the advantage of converging even in the presence of low-frequency noise. Moreover, since these variances depend on the sampling period, they describe the temporal structure of the signals, and thus identify the dominant noise type.
- We have pursued and advanced Patrick Haddad's work on the estimation of the proportion of thermal noise in the jitter variance. The accuracy of this estimation shows a strong dependence on the share of flicker noise, suggesting that an accurate estimation of the share due to thermal noise will be almost impossible as the proportion of flicker noise increases due to the decrease in the size of the transistors.
- We studied the design of TRNGs based on PLLs, as well as the search for optimal parameters. This study led to the development of an efficient search tool for these parameters, based on the stochastic model of the generator. The interest of this tool is to reduce the design time of PLL-TRNGs while providing configurations that respect security and performance constraints.
- We have studied a new approach to the evaluation of TRNGs proposed by DGA-MI and have illustrated it by using PLL-based TRNGs. As this approach was initially illustrated

using an RO based generator, we have shown that it can also be applied to other types of generators. We have also shown that the PLL-based generator can be used for systems with high security requirements.

- We have studied the design of PLLs and carried out simulations based on models adapted to the FPGAs we are working on. This has highlighted the fact that the PLL output noise comes mainly from the VCO. We also studied the behavior of the PLL as a source of randomness for the generation of random numbers, specifying the conditions on the parameters allowing the hypotheses of the models to be valid.

Some of these contributions have been published in the following scientific journals and conferences:

Scientific journal publication

[193] 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, Volume 2018, Issue 3*, pp. 214–242, Aug. 2018

International conference publication

[194] E. Noumon Allini, O. Petura, V. Fischer, and F. Bernard, "Optimization of the PLL configuration in a PLL-based TRNG design," in *2018 Design, Automation Test in Europe Conference Exhibition (DATE)*, pp. 1265–1270, IEEE, Mar. 2018

[161] O. Petura, M. Laban, E. Noumon Allini, and V. Fischer, "Two Methods of the Clock Jitter Measurement Aimed at Embedded TRNG Testing," in *TRUDEVICE – 8th Conference on Trustworthy Manufacturing and Utilization of Secure Devices (TRUDEVICE 2018), Dresden, Germany*, p. 5, March 2018

[195] E. Noumon Allini, F. Bernard, and V. Fischer, "An Illustration of a New Certification Approach for TRNGs," in *Workshop on Cryptographic Architectures Embedded in Logic Devices, Cryptarchi 2017*, June 2017

Les travaux réalisés dans le cadre de ce doctorat ont conduit aux contributions suivantes.

- Nous avons montré que la génération de nombres aléatoires à l'aide de valeurs de compteur est plus efficace que la méthode basée sur l'échantillonnage. Elle améliore le débit des TRNGs, tout en servant de base aux tests en ligne.

- Nous avons attiré l'attention sur les risques liés à l'utilisation de la variance classique pour caractériser le jitter. Ce risque provient des composantes basse fréquence du jitter qui conduisent à une divergence dans le comportement de la variance classique et conduisent à des résultats qui ne peuvent pas être interprétés.

- Nous avons proposé l'utilisation de la variance d'Allan (et ses variantes) comme solution alternative pour caractériser les fluctuations de phase et de fréquence. Ces variances, largement utilisées dans le domaine de l'étude de la stabilité des fréquences des oscillateurs ultra stable, ont l'avantage de converger même en présence de bruit à basse fréquence. De plus, comme ces variances dépendent de la période d'échantillonnage, elles décrivent la structure temporelle des signaux, et identifient ainsi le type de bruit dominant.

- Nous avons poursuivi et fait avancer les travaux de Patrick Haddad sur l'estimation de la proportion du bruit thermique dans la variance du jitter. La précision de cette estimation montre une forte dépendance de la part du bruit flicker, ce qui suggère qu'une estimation précise de la part due au bruit thermique sera presque impossible car la proportion du bruit flicker augmente en raison de la diminution de la taille des transistors.

- Nous avons étudié la conception de TRNGs basés sur les PLLs, ainsi que la recherche de paramètres optimaux. Cette étude a conduit au développement d'un outil de recherche efficace de ces paramètres, basé sur le modèle stochastique du générateur. L'intérêt de cet outil est de réduire le temps de conception des PLL-TRNGs tout en fournissant des configurations qui respectent les contraintes de sécurité et de performance.

- Nous avons étudié une nouvelle approche de l'évaluation des TRNGs proposée par la DGAMI et l'avons illustrée en utilisant des TRNGs basés sur les PLLs. Comme cette approche a été initialement illustrée en utilisant un générateur basé sur les ROs, nous avons montré qu'elle peut également être appliquée à d'autres types de générateurs. Nous avons également montré que le générateur basé sur les PLLs peut être utilisé pour des systèmes ayant des exigences de sécurité élevées.

- Nous avons étudié la conception des PLLs et réalisé des simulations basées sur des modèles adaptés aux FPGAs sur lesquels nous travaillons. Cela a permis de mettre en évidence

le fait que le bruit en sortie des PLLs provient principalement du VCO. Nous avons également étudié le comportement de la PLL en tant que source d'aléa pour la génération de nombres aléatoires, en précisant les conditions sur les paramètres permettant de valider les hypothèses des modèles.

Certaines de ces contributions ont été publiées dans les revues et conférences scientifiques suivantes :

Publication dans les revues scientifiques

[193] 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, Volume 2018, Issue 3*, pp. 214–242, Aug. 2018

Publication dans les conférences internationales

[194] E. Noumon Allini, O. Petura, V. Fischer, and F. Bernard, "Optimization of the PLL configuration in a PLL-based TRNG design," in *2018 Design, Automation Test in Europe Conference Exhibition (DATE)*, pp. 1265–1270, IEEE, Mar. 2018

[161] O. Petura, M. Laban, E. Noumon Allini, and V. Fischer, "Two Methods of the Clock Jitter Measurement Aimed at Embedded TRNG Testing," in *TRUDEVICE – 8th Conference on Trustworthy Manufacturing and Utilization of Secure Devices (TRUDEVICE 2018), Dresden, Germany*, p. 5, March 2018

[195] E. Noumon Allini, F. Bernard, and V. Fischer, "An Illustration of a New Certification Approach for TRNGs," in *Workshop on Cryptographic Architectures Embedded in Logic Devices, Cryptarchi 2017*, June 2017

Conclusion

In this thesis, we studied the design of a random number generator for applications with high security requirements. According to the received guidelines, we focused on TRNGs implementable in logic circuits. This type of generator uses various physical phenomena such as clock jitter. Since clock jitter is widely used to implement TRNGs in logic circuits, we opted for TRNGs using clock jitter as a source of randomness. In this thesis, we focused our efforts on the compliance of TRNGs with a new evaluation approach proposed by the DGA-MI. This approach aims at complementing the requirements of the PTG.3 class of AIS-31, in particular with respect to the physical noise source.

In order to ensure that the entropy at the output of the noise source comes from the desired phenomena, such as thermal noise, significant emphasis has been placed on the measurement methods of jitter. As such a process requires a thorough knowledge of jitter and its components, jitter modeling is required. For this purpose, we have used tools developed by the timekeeping research community. A study of these tools allowed us to verify that the use of classical variance to characterize the jitter can compromise the security of the generator due to the autocorrelated components of the jitter.

We have proposed the use of the Allan variance to characterize jitter, as is done in the timekeeping domain. A thorough study of this variance and its variants made us see its numerous advantages such as its stability, its implementability in hardware, and its low resource consumption compared to existing methods. Moreover, its operation allows a direct implementation of the counter values, which are the basis of the only existing embedded tests of TRNGs.

Furthermore, the Allan variance has a time version that allows direct characterization of the jitter in the time domain. This characterization makes it possible to study the temporal structure of the jitter, and to deduce the dominant noise type as a function of the accumulation time. In particular, the Allan time variance distinguishes white noise from flicker noise. This property was verified in simulation using the noise generator of Kasdin and Walter. This noise generator is an

algorithmic method widely used in the timekeeping research domain. It can be used to simulate the different types of noise often encountered in practice. The understanding of the Allan time variance, which was reinforced by the various simulations, was then used to investigate hypotheses concerning the constitution of the jitter at the output of the oscillators such as ROs and PLLs.

In this work, special attention has been paid to PLL as a source of randomness for TRNGs. In order to be consistent with the DGA-MI approach, it is imperative to understand the operation of the PLL as well as the effect of its various parameters on the quality of the output noise. For this purpose, we have adapted and extended the PLL model available in the state-of-the-art to the PLLs implemented in the FPGAs we used. This led us to a model of the PLL in terms of transfer functions. This model was then simulated in order to verify the different assumptions made about the PLL as a noise source.

One of these assumptions concerns the source of the output noise of the PLL. Although designers of TRNGs consider that the noise comes from the VCO, we have seen no state-of-the-art arguments to support this fact. To check this assumption, we injected noise into the different blocks of the PLL, and by observing the output, we were able to establish that the noise at the output of the PLL does indeed come from the VCO. The assumption that the flicker noise at the output of the PLL is negligible remains true as long as the coefficient of division M is small. However, it is necessary that this coefficient is large to accumulate enough entropy. Conversely, a large value of M would contribute to amplify the noise of the input signal. These observations show that the constraints on M, in order to have a good quality of randomness, are often in conflict with each other.

A similar situation is repeated with the other parameters of the PLL. Since the PLL behaves like a high-pass filter with respect to the VCO, the PLL tends to distort the nature of the signals present in the frequency region below its natural frequency ω_n. Signals present at frequencies above ω_n are transmitted at the output of the PLL with no change in nature. Thus, to avoid flicker noise being present at the output of the PLL, it is necessary that ω_n is large. However, a large value of ω_n would not allow entropy to accumulate long enough before the signals are distorted. These results indicate the need to make trade-offs in the choice of values for the parameters.

Having understood how the PLL works as a source of randomness, we have described the principle of the generator based on PLLs. This description allowed us to recall fundamental relationships that govern the operation of the PLL-based TRNG, and to explain the reasons for considering a TRNG with two PLLs instead of one PLL. We also described the DGA-MI evaluation approach by

illustrating it with the PLL-based generator. We then found that the PLL-based generator meets all the requirements of this approach. This proves not only that the DGA-MI approach is not restricted only to RO-based TRNGs, but also that the PLL-based principle can safely be used for systems with high security requirements.

The DGA-MI approach raises the problem of adjusting the TRNG parameters to ensure excellent generator operation. In the case of the generator based on PLLs, this means finding configurations that ensure good sensitivity to the jitter, while having the highest possible throughput. Previous work has been carried out to this effect, proposing a method based on a genetic algorithm. However, due to the large number of generator parameters using two PLLs, this method could not be applied to the TRNGs of interest. We then proposed a method based on a depth-first search strategy. Experimental results showed that the latter method not only applies to generators based on two PPLs, but also produces better results than the previous method in less time.

During this thesis, we continued the estimation of the share of jitter due to thermal noise. This was done using the Allan time variance. We were able to come up with a method that produces good results in simulations. However, this method is sensitive to the proportion of jitter due to flicker noise. The higher the proportion, the less accurate it is. This can be explained by the fact that for large portions of flicker noise, thermal noise tends to be drowned out and therefore difficult to measure. In order to avoid such a situation, it is important that the oscillators are designed in such a way that the flicker noise proportion is as low as possible.

However, with the reduction of transistor size, it would be difficult to keep this proportion low. An alternative solution would be to characterize the flicker noise as well as its entropy contribution. Since it is an autocorrelated noise, it would be reasonable to opt for a characterization using Markov chains. This possibility is strengthened by the fact that a relatively recent result makes it possible to evaluate the min entropy of phenomena characterized by Markov chains.

The study of the PLL also highlighted the fact that the noise at the output of the PLL is bounded. A determination of these bounds would make it possible to bound the entropy present at the output of the PLLs. This would then allow the generator model based on the PLLs to be strengthened. It should be noted, however, that the fact that the noise is bounded is probably due to the feedback action of the PLL. This PLL feedback prevents the noise from accumulating freely and resets the random walk at the output of the PLL. A consequence of this mechanism is that the random walks between two corrections of the PLL do not have the same statistics. In other words, the noise at the output of the PLL can be considered stationary as long as the PLL correction does

not intervene, not after the correction. A thorough study of this matter should be made in order to determine the minimum time taken by the PLL to correct the feedback signal.

Perspectives

Although we have made progress in understanding the sources of physical noise, some aspects remain to be further investigated.

- As the PLL is a feedback system, it locks its output signal to its input signal. However, this locking is not instantaneous. After a sudden change operated on the input signal, there is always a delay before the output signal is locked again. During this delay, called PLL settling time, the PLL no longer behaves as a linear system. It is therefore important to determine this settling time, in order not to generate numbers before this time has elapsed.
- Each time the PLL feedback operates, the PLL output random walk is reset. This implies that the noise at the output of the PLL is stationary as long as the PLL does not adjust the output signal. It is therefore important to determine the time that a PLL will take to adjust the signal in order to define the limits of the stationarity of the output noise of the PLL.
- We have confirmed that the noise accumulated at the output of the PLL is bounded. Even if the experimental results show that this bound is constant for PLLs implanted in Cyclone V FPGAs, we have not yet established its mathematical expression in order to know the factors that influence it. A possible extension of the work carried out in the course of this thesis could also concern the determination of this mathematical expression.
- Results obtained during this thesis were obtained using the stationarity assumption. However, it turns out that depending on the accumulation time, this assumption may not be reasonable, so it is necessary to adopt new characterizations of electronic noise that do not use the stationarity assumption. The use of Markov chains or wavelets seems to be good alternatives.
- Due to the reduction in transistor size, it will be increasingly difficult to filter flicker noise and still have enough entropy at the output of the random number generators. It is therefore imperative to propose a characterization of the flicker noise in order to evaluate its entropy contribution.

The preceding points will have to be investigated in depth in order to improve our understanding of how TRNGs work, which will enable us to better evaluate them.

Au cours de cette thèse, nous avons étudié la conception d'un générateur de nombres aléatoires pour des applications ayant des exigences de sécurité élevées. Selon les directives reçues, nous nous sommes concentrés sur les TRNGs implantables dans les circuits logiques. Ce type de générateur utilise divers phénomènes physiques tels que le jitter d'horloge. Comme le jitter d'horloge est largement utilisé pour implémenter des TRNGs dans les circuits logiques, nous avons opté pour des TRNGs utilisant le jitter d'horloge comme source d'aléa. Nous avons concentré nos efforts sur la conformité des TRNGs avec une nouvelle approche d'évaluation proposée par la DGA-MI. Cette approche vise à compléter les exigences de la classe PTG.3 de l'AIS-31, notamment en ce qui concerne la source de bruit physique.

Pour garantir que l'entropie à la sortie de la source de bruit provient des phénomènes souhaités, tels que le bruit thermique, un accent important a été mis sur les méthodes de mesure du jitter. Comme un tel processus nécessite une connaissance approfondie du jitter et de ses composantes, une modélisation du jitter est nécessaire. À cette fin, nous avons utilisé des outils développés par la communauté de recherche sur la stabilité des fréquences d'oscillateurs ultra stables. L'étude de ces outils nous a permis de vérifier que l'utilisation de la variance classique pour caractériser le jitter peut compromettre la sécurité du générateur à cause de ses composants autocorrélés, tels que le bruit flicker.

Nous avons proposé d'utiliser la variance d'Allan pour caractériser le jitter, comme cela se fait dans le domaine de l'étude de la stabilité des fréquences d'oscillateurs ultra stables. Une étude approfondie de cette variance et de ses variantes nous a permis de constater ses nombreux avantages tels que sa stabilité, son implantabilité dans le matériel, et sa faible consommation en ressources comparativement aux méthodes existantes. De plus, son fonctionnement permet une implémentation directe des valeurs de comptage, qui sont à la base des seuls tests embarqués existants pour les TRNGs.

En outre, la variance d'Allan a une version temporelle qui permet de caractériser directement le jitter dans le domaine temporel. Cette caractérisation permet d'étudier la structure temporelle du jitter, et d'en déduire le type de bruit dominant en fonction du temps d'accumulation. En particulier, la variance temporelle d'Allan permet de distinguer le bruit blanc du bruit flicker. Cette propriété a été vérifiée en simulation à l'aide du générateur de bruit de Kasdin et Walter. Ce générateur de bruit est une méthode algorithmique recommandée dans le domaine de la recherche sur la stabilité des fréquences d'oscillateurs ultra stables. Il peut être utilisé pour simuler les différents types de bruit souvent rencontrés dans la pratique. La compréhension de la variance temporelle d'Allan, qui a été renforcée par les différentes simulations, a ensuite été

utilisée pour étudier les hypothèses concernant la constitution du jitter à la sortie des oscillateurs tels que les ROs et les PLLs.

Dans le cadre de ce travail, une attention particulière a été accordée aux PLLs en tant que source d'aléa pour les TRNGs. Afin d'être cohérent avec l'approche de la DGA-MI, il est impératif de comprendre le fonctionnement de la PLL ainsi que l'effet de ses différents paramètres sur la qualité du bruit de sortie. À cette fin, nous avons adapté et étendu le modèle de PLL disponible dans l'état de l'art aux PLLs implantées dans les FPGAs que nous avons utilisés. Cela nous a conduit à un modèle de la PLL en termes de fonctions de transfert. Ce modèle a ensuite été simulé afin de vérifier les différentes hypothèses faites sur la PLL en tant que source de bruit.

L'une de ces hypothèses concerne la source de bruit en sortie de la PLL. Bien que les concepteurs des TRNGs considèrent que ce bruit provient du VCO, nous n'avons vu aucun élément de l'état de l'art pour étayer ce fait. Pour vérifier cette hypothèse, nous avons injecté du bruit dans les différents blocs de la PLL, et en observant la sortie, nous avons pu établir que le bruit à la sortie de la PLL provient bien du VCO. L'hypothèse selon laquelle le bruit flicker à la sortie de la PLL est négligeable reste vraie tant que le coefficient de division M est faible. Cependant, il est nécessaire que ce coefficient soit important pour accumuler une entropie suffisante. Inversement, une valeur importante de M contribuerait à amplifier le bruit du signal d'entrée. Ces observations montrent que les contraintes sur M, afin d'avoir une bonne qualité d'aléa, sont souvent en conflit les unes avec les autres.

Une situation similaire se répète avec les autres paramètres de la PLL. Comme la PLL se comporte comme un filtre passe-haut par rapport au VCO, la PLL a tendance à déformer la nature des signaux présents dans la région de fréquence en dessous de sa fréquence naturelle ω_n. Les signaux présents à des fréquences supérieures à ω_n sont transmis à la sortie de la PLL sans changement de nature. Ainsi, pour éviter que le bruit flicker soit présent à la sortie de la PLL, il est nécessaire que ω_n soit important. Cependant, une valeur élevée de ω_n ne permettrait pas à l'entropie de s'accumuler suffisamment avant que les signaux soient déformés. Ces résultats indiquent qu'il est nécessaire de faire des compromis dans le choix des valeurs des paramètres.

Après avoir compris le fonctionnement de la PLL comme source d'aléa, nous avons décrit le principe du générateur basé sur les PLLs. Cette description nous a permis de rappeler les relations fondamentales qui régissent le fonctionnement du TRNG basé sur les PLLs, et d'expliquer les raisons pour lesquelles on envisage un TRNG avec deux PLLs au lieu d'une seule. Nous avons également décrit la démarche d'évaluation de la DGA-MI en l'illustrant avec le générateur à base

de PLL. Nous avons ensuite constaté que le générateur à base de PLL répond à toutes les exigences de cette approche. Cela prouve non seulement que l'approche de la DGA-MI n'est pas limitée aux TRNG basés sur les ROs, mais aussi que le principe basé sur les PLLs peut être utilisé en toute sécurité pour des systèmes ayant des exigences de sécurité élevées.

L'approche de la DGA-MI pose le problème de l'ajustement des paramètres du TRNG pour assurer un fonctionnement optimal du générateur. Dans le cas du générateur basé sur les PLLs, il s'agit de trouver des configurations qui assurent une bonne sensibilité au jitter, tout en ayant le plus haut débit possible. Des travaux antérieurs ont été menés à cet effet, proposant une méthode basée sur un algorithme génétique. Cependant, en raison du grand nombre de paramètres du générateur utilisant deux PLLs, cette méthode n'a pas pu être appliquée aux TRNGs qui nous intéressent. Nous avons alors proposé une méthode basée sur une stratégie de recherche en profondeur. Les résultats expérimentaux ont montré que cette dernière méthode ne s'applique pas seulement aux générateurs basés sur deux PLL, mais qu'elle produit également de meilleurs résultats que la méthode précédente en moins de temps.

Nous avons également poursuivi l'estimation de la part du jitter due au bruit thermique. Cela a été fait en utilisant la variance temporelle d'Allan. Nous avons réussi à mettre au point une méthode qui donne de bons résultats dans les simulations. Cependant, cette méthode est sensible à la proportion de jitter due au bruit flicker. Plus cette proportion est élevée, moins elle est précise. Cela peut s'expliquer par le fait que pour de grandes portions de bruit flicker, le bruit thermique a tendance à être noyé et donc difficile à mesurer. Afin d'éviter une telle situation, il est important que les oscillateurs soient conçus de manière à ce que la proportion de bruit de scintillement soit la plus faible possible.

Cependant, avec la réduction de la taille des transistors, il serait difficile de maintenir cette proportion à un niveau bas. Une solution alternative consisterait à caractériser le bruit de scintillement ainsi que sa contribution à l'entropie. Comme il s'agit d'un bruit autocorrélé, il serait raisonnable d'opter pour une caractérisation utilisant des chaînes de Markov. Cette possibilité est renforcée par le fait qu'un résultat relativement récent permet d'évaluer la min-entropie des phénomènes caractérisés par des chaînes de Markov.

L'étude de la PLL a également mis en évidence le fait que le bruit à la sortie de la PLL est borné. La détermination de ces bornes permettrait de borner l'entropie disponible à la sortie des PLLs. Cela permettrait ensuite de renforcer le modèle stochastique du générateur basé sur les PLLs. Il convient toutefois de noter que le fait que le bruit soit borné est probablement dû à l'action

de rétroaction de la PLL. Cette rétroaction de la PLL empêche le bruit de s'accumuler librement et réinitialise la marche aléatoire à la sortie de la PLL. Une conséquence de ce mécanisme est que les marches aléatoires entre deux corrections de la PLL n'ont pas les mêmes statistiques. En d'autres termes, le bruit à la sortie de la PLL peut être considéré comme stationnaire tant que la correction de la PLL n'intervient pas, et non après la correction. Une étude approfondie de cette question doit être effectuée afin de déterminer le temps minimum pris par la PLL pour corriger le signal de retour.

Perspectives à la thèse

Bien que nous ayons fait des progrès dans la compréhension des sources de bruit physique, certains aspects restent à approfondir.

- Comme la PLL est un système asservi, elle verrouille son signal de sortie sur son signal d'entrée. Toutefois, ce verrouillage n'est pas instantané. Après un changement soudain opéré sur le signal d'entrée, il y a toujours un délai avant que le signal de sortie ne soit à nouveau verrouillé. Pendant ce délai, appelé temps de réponse de la PLL, la PLL ne se comporte plus comme un système linéaire. Il est donc important de déterminer ce temps de stabilisation, afin de ne pas générer de nombres avant que ce temps ne soit écoulé.

- Chaque fois que la rétroaction de la PLL intervient, la marche aléatoire de la sortie de la PLL est réinitialisée. Cela implique que le bruit à la sortie de la PLL est stationnaire tant que la PLL n'ajuste pas le signal de sortie. Il est donc important de déterminer le temps que prendra une PLL pour ajuster le signal afin de définir les limites de la stationnarité du bruit de sortie de la PLL.

- Les résultats obtenus au cours de cette thèse ont été obtenus en utilisant l'hypothèse de stationnarité. Cependant, il s'avère qu'en fonction du temps d'accumulation, cette hypothèse peut ne pas être raisonnable, il est donc nécessaire d'adopter de nouvelles caractérisations des bruits électroniques qui n'utilisent pas cette hypothèse. L'utilisation de chaînes de Markov ou d'ondelettes semble être une bonne alternative.

- Nous avons confirmé que l'accumulation du bruit en sortie de la PLL est bornée. Même si les résultats expérimentaux montrent que cette borne est constante pour les PLL implantées dans les FPGAs Cyclone V, nous n'avons pas encore établi son expression mathématique afin de connaître les facteurs qui l'influencent. Une extension éventuelle des travaux réalisés dans le cadre de cette thèse pourrait également concerner la détermination de cette expression mathématique.

- En raison de la réduction de la taille des transistors, il sera de plus en plus difficile de filtrer le bruit flicker et d'avoir encore suffisamment d'entropie à la sortie des générateurs de nombres aléatoires. Il est donc impératif de proposer une caractérisation du bruit flicker afin d'évaluer sa contribution à l'entropie.

Les points précédents devront être étudiés en profondeur afin d'améliorer notre compréhension du fonctionnement des TRNG, ce qui nous permettra de mieux les évaluer.

List of Figures

List of Tables

References

[1] S. Singh, *The Code Book: The Secret History of Codes and Code-Breaking*. Fourth Estate, 2002.

[2] H. R. Nemati, H. R. Nemati, and L. Yang, *Applied Cryptography for Cyber Security and Defense*. Hershey, PA, USA: IGI Global, 1st ed., 2010.

[3] M.-H. Education, ed., *Modern Cryptography: Applied Mathematics for Encryption and Information Security*, 2016.

[4] R. L. Rivest, A. Shamir, and L. Adleman, "A Method for Obtaining Digital Signatures and Public-key Cryptosystems," *Communications of the ACM*, vol. 21, pp. 120–126, February 1978.

[5] T. ElGamal, "A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms," *IEEE Transactions on Information Theory*, vol. 31, pp. 469–472, July 1985.

[6] V. S. Miller, "Use of Elliptic Curves in Cryptography," in *Advances in Cryptology - CRYPTO'85 Proceedings*, vol. 218 of *Lecture Notes in Computer Science*, pp. 417–426, Springer Berlin Heidelberg, 1986.

[7] A. Kerckhoffs, "La cryptographie militaire," *Journal des Sciences Militaires*, vol. IX, pp. 5–38, January 1883. http://www.petitcolas.net/kerckhoffs/crypto_militaire_1.pdf.

[8] A. J. Menezes, S. A. Vanstone, and P. C. V. Oorschot, *Handbook of Applied Cryptography*. Boca Raton, FL, USA: CRC Press, Inc., 1st ed., 1996.

[9] R. T. Kneusel, *Random Numbers and Computers*. Cham: Springer International Publishing, 2018.

[10] V. Fischer, "A Closer Look at Security in Random Number Generators Design," in *Constructive Side-Channel Analysis and Secure Design* (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, W. Schindler, and S. A. Huss, eds.), vol. 7275, pp. 167–182, Berlin, Heidelberg: Springer Berlin Heidelberg, 2012.

[11] W. Schindler, "Random Number Generators for Cryptographic Applications," in *Cryptographic Engineering* (K. Ç.K, ed.), (Boston, MA), pp. 5–23, Springer, 2009.

[12] Y. Cao, *Securing Hardware Random Number Generators against Physical Attacks*. PhD thesis, KU Leuven, 2016.

[13] W. Killmann and W. Schindler, "A Design for a Physical RNG with Robust Entropy Estimators," in *Cryptographic Hardware and Embedded Systems - CHES 2008: 10th International Workshop, Washington, D.C., USA, August 10-13, 2008. Proceedings* (E. Oswald and P. Rohatgi, eds.), (Berlin, Heidelberg), pp. 146–163, Springer Berlin Heidelberg, 2008.

[14] V. Fischer, P. Haddad, and A. Cherkaoui, "Ring oscillators and self-timed rings in true random number generators," in *Oscillator Circuits: Frontiers in Design, Analysis and Applications* (Y. Nishio, ed.), vol. 32 of *Materials, Circuits and Devices*, pp. 267–292, Institution of Engineering and Technology, 2016.

[15] W. Killmann and W. Schindler, "A proposal for: Functionality classes and evaluation methodology for true (physical) random number generators," standard, Bundesamt für Sicherheit in der Informationstechnik, Bonn, Germany, September 2001.

[16] S. Keller and T. A. Hall, "The NIST SP 800-90a Deterministic Random Bit Generator Validation System (DRBGVS)," tech. rep., NIST, 2015.

[17] 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," Tech. Rep. NIST SP 800-90b, National Institute of Standards and Technology, Gaithersburg, MD, Jan. 2018.

[18] E. Barker and J. Kelsey, "Recommendation for Random Bit Generator (RBG) Constructions," tech. rep., NIST, 2012.

[19] W. Schindler and W. Killmann, "Evaluation Criteria for True (Physical) Random Number Generators Used in Cryptographic Applications," in *Cryptographic Hardware and Embedded Systems - CHES 2002: 4th International Workshop Redwood Shores, CA, USA, August 13-15, 2002 Revised Papers* (B. S. Kaliski, ç. K. Koç, and C. Paar, eds.), (Berlin, Heidelberg), pp. 431–449, Springer Berlin Heidelberg, 2003.

[20] D. S. Wilks and R. L. Wilby, "The weather generation game: a review of stochastic weather models," *Progress in Physical Geography: Earth and Environment*, vol. 23, no. 3, pp. 329–357, 1999.

[21] E. Adams and A. Rollings, *Fundamentals of game design*. Voices that matter, Berkeley, CA: New Riders, 2nd ed ed., 2010.

[22] J. Buchmann, *Introduction to cryptography*. Undergraduate texts in mathematics, New York: Springer, 2nd ed., 2004.

[23] S. Tezuka, *Uniform Random Numbers: Theory and Practice*, vol. 315 of *The Springer International Series in Engineering and Computer Science 315*. Springer US, 1995.

[24] W. Killmann and W. Schindler, "A proposal for: Functionality classes for random number generators," tech. rep., Bundesamt für Sicherheit in der Informationstechnik (BSI), Bonn, 2011.

[25] D. Bose, "Uniform Pseudo-Random Number Generation," January 2018.

[26] P. L'Ecuyer, "Random number generation," in *Handbook of computational statistics: concepts and methods* (J. E. Gentle, W. Härdle, and Y. Mori, eds.), Heidelberg ; New York: Springer, 2nd rev. and updated ed ed., 2012.

[27] P. L'Ecuyer, "Uniform random number generation," *Annals of Operations Research*, pp. 77–120, 1994.

[28] C. Paar and J. Pelzl, *Understanding cryptography: a textbook for students and practitioners*. Heidelberg ; New York: Springer, 2010. OCLC: ocn527339793.

[29] W. Schindler, "Functionality Classes and Evaluation Methodology for Deterministic Random Number Generators," standard, Bundesamt für Sicherheit in der Informationstechnik, Bonn, Germany, December 1999.

[30] R. Mita, G. Palumbo, S. Pennisi, and M. Poli, "A novel pseudo random bit generator for cryptography applications," in *9th International Conference on Electronics, Circuits and Systems*, vol. 2, (Dubrovnik, Croatia), pp. 489–492, IEEE, 2002.

[31] E. Barker, "Recommendation for Key Management Part 1: General," Tech. Rep. NIST SP 800-57pt1r4, National Institute of Standards and Technology, Jan. 2016.

[32] D. E. Knuth, *The Art of Computer Programming, Volume 2 (3rd Ed.): Seminumerical Algorithms*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1997.

[33] D. R. Stinson, *Cryptography: Theory and Practice, Third Edition*. Discrete Mathematics and Its Applications, Taylor & Francis, 2005.

[34] J. von Neumann, "Various techniques used in connection with random digits," in *Monte Carlo Method* (A. Householder, G. Forsythe, and H. Germond, eds.), pp. 36–38, Washington, D.C.: U.S. Government Printing Office: National Bureau of Standards Applied Mathematics Series, 12, 1951.

[35] E. D. Cashwell and C. J. Everett, *A Practical Manual on the Monte Carlo Method for Random Walk Problems*. International tracts in computer science and technology and their application, London: Pergamon Press, New York 1959.

[36] G. E. Forsythe, "Generation and Testing of Random Digits at the National Bureau of Standards," in *Monte Carlo Method*, vol. 12 of *Applied Mathematics Series*, Los Angeles: National Bureau of Standards, 1951.

[37] P. C. Hammer, "The mid-square method of generating digits," in *Monte Carlo Method*, vol. 12 of *Applied Mathematics Series*, (Los Angeles), National Bureau of Standards, 1951.

[38] J. W. Mauchly, "Pseudo-random numbers," 1949.

[39] O. Taussky and J. Todd, "Generation and testing of pseudo-random numbers," in *Symposium on Monte Carlo Methods*, pp. 15–28, Herbert A. Meyer (Wiley, New York), 1956.

[40] D. H. Lehmer, "Mathematical methods in large-scale computing units," in *Proceedings of the Second Symposium on Large Scale Digital Computing Machinery*, (Cambridge, United Kingdom), pp. 141–146, Harvard University Press, 1951.

[41] A. Rotenberg, "A new pseudo-random number generator," *J. ACM*, vol. 7, pp. 75–77, Jan. 1960.

[42] R. C. Tausworthe, "Random Numbers Generated by Linear Recurrence Modulo Two," *Mathematics of Computation - Math. Comput.*, vol. 19, p. 9, 05 1965.

[43] P. L'Ecuyer, "Random numbers for simulation," *Communications of the ACM*, vol. 33, pp. 85–97, Oct. 1990.

[44] A. Klein, *Stream ciphers*. New York: Springer, 1st ed ed., 2013.

[45] T. G. Lewis and W. H. Payne, "Generalized Feedback Shift Register Pseudorandom Number Algorithm," *Journal of the ACM*, vol. 20, pp. 456–468, July 1973.

[46] M. Matsumoto and Y. Kurita, "Twisted GFSR generators," *ACM Transactions on Modeling and Computer Simulation*, vol. 2, pp. 179–194, July 1992.

[47] M. Matsumoto and T. Nishimura, "Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator," *ACM Transactions on Modeling and Computer Simulation*, vol. 8, pp. 3–30, Jan. 1998.

[48] C. K. Caldwell, "Mersenne primes: History, theorems and lists." https://primes.utm.edu/mersenne, December 2017. accessed 20/12/2018.

[49] W. E. Brown, "Random Number Generation in C++11," March 2013.

[50] G. van Rossum, *The Python Library Reference*. Network Theory Limited (1 mars 2011), September 2018.

[51] C. Dutang and P. Kiener, "CRAN Task View: Probability Distributions." Retrieved 2018-09-18, June 2018.

[52] G. Markowsky, "The Sad History of Random Bits," *Journal of Cyber Security and Mobility*, vol. 3, no. 1, pp. 1–24, 2014.

[53] M. Blum and S. Micali, "How to Generate Cryptographically Strong Sequences of Pseudo-random Bits," *SIAM J. Comput.*, vol. 13, pp. 850–864, Nov. 1984.

[54] L. Blum, M. Blum, and M. Shub, "A Simple Unpredictable Pseudo-Random Number Generator," *SIAM Journal on Computing*, vol. 15, pp. 364–383, May 1986.

[55] J. Hurd, "Blum integers." Trinity College, Cambridge, January 1997. retrieved 20/12/2018, http://www.gilith.com/talks/cambridge1997.pdf.

[56] N. Ferguson and B. Schneier, *Practical Cryptography*. New York: Wiley, 2003. OCLC: ocm51568066.

[57] M. Stipčević, "Fast nondeterministic random bit generator based on weakly correlated physical events," *Review of Scientific Instruments*, vol. 75, pp. 4442–4449, Nov. 2004.

[58] Z. Gutterman, B. Pinkas, and T. Reinman, "Analysis of the Linux random number generator," in *2006 IEEE Symposium on Security and Privacy (S&P'06)*, (Berkeley/Oakland, CA), pp. 15 pp.–385, IEEE, 2006.

[59] M. Herrero-Collantes and J. C. Garcia-Escartin, "Quantum Random Number Generators," *Reviews of Modern Physics*, vol. 89, p. 48, Feb. 2017. arXiv: 1604.03304.

[60] I. Vasyltsov, E. Hambardzumyan, Y.-S. Kim, and B. Karpinskyy, "Fast digital trng based on metastable ring oscillator," in *Cryptographic Hardware and Embedded Systems - CHES 2008, 10th International Workshop, Washington, D.C., USA, August 10-13, 2008. Proceedings*, vol. 5154 of *Lecture Notes in Computer Science*, pp. 164–180, Springer, 2008.

[61] J. Szczepanski, E. Wajnryb, J. Amigó, M. V. Sanchez-Vives, and M. Slater, "Biometric random number generators," *Computers & Security*, vol. 23, pp. 77–84, Feb. 2004.

[62] H. F. Murry, "A general approach for generating natural random variables," *IEEE Transactions on Computers*, vol. C-19, pp. 1210–1213, Dec 1970.

[63] C. Petrie and J. Connelly, "A noise-based IC random number generator for applications in cryptography," *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, vol. 47, pp. 615–621, May 2000.

[64] P. Kohlbrenner and K. Gaj, "An Embedded True Random Number Generator for FPGAs," in *Proceedings of the 2004 ACM/SIGDA 12th International Symposium on Field Programmable Gate Arrays*, FPGA '04, (New York, NY, USA), pp. 71–78, ACM, February 22-24 2004.

[65] M. Bucci and R. Luzzi, "Fully Digital Random Bit Generators for Cryptographic Applications," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 55, pp. 861–875, Apr. 2008.

[66] O. Petura, U. Mureddu, N. Bochard, V. Fischer, and L. Bossuet, "A survey of AIS-20/31 compliant TRNG cores suitable for FPGA devices," in *2016 26th International Conference on Field Programmable Logic and Applications (FPL)*, (Lausanne, Switzerland), pp. 1–10, IEEE, Aug. 2016.

[67] A. Cherkaoui, V. Fischer, A. Aubert, and L. Fesquet, "A Self-Timed Ring Based True Random Number Generator," in *2013 IEEE 19th International Symposium on Asynchronous Circuits and Systems*, (Santa Monica, CA, USA), pp. 99–106, IEEE, May 2013.

[68] V. Fischer and M. Drutarovský, "True Random Number Generator Embedded in Reconfigurable Hardware," in *Cryptographic Hardware and Embedded Systems - CHES 2002: 4th International Workshop Redwood Shores, CA, USA, August 13-15, 2002 Revised Papers* (B. S. Kaliski, ç. K. Koç, and C. Paar, eds.), (Berlin, Heidelberg), pp. 415–430, Springer Berlin Heidelberg, 2003.

[69] D. Lubicz and N. Bochard, "Towards an Oscillator Based TRNG with a Certified Entropy Rate," *IEEE Transactions on Computers*, vol. 64, pp. 1191–1200, Apr. 2015.

[70] F. Dupuis, O. Fawzi, and R. Renner, "Entropy accumulation," *arXiv:1607.01796 [quant-ph]*, p. 40, Jul 2016. arXiv: 1607.01796.

[71] V. Fischer, M. Drutarovský, M. Šimka, and N. Bochard, "High performance true random number generator in Altera stratix FPLDs," in *Field Programmable Logic and Application: 14th International Conference, FPL 2004, Leuven, Belgium, August 30-September 1, 2004. Proceedings* (J. Becker, M. Platzner, and S. Vernalde, eds.), (Berlin, Heidelberg), pp. 555–564, Springer Berlin Heidelberg, 2004.

[72] B. Chor and O. Goldreich, "Unbiased bits from sources of weak randomness and probabilistic communication complexity," in *26th Annual Symposium on Foundations of Computer Science (sfcs 1985)*, pp. 429–442, Oct 1985.

[73] M. Stipčević and c. K. Koç, "True Random Number Generators," in *Open Problems in Mathematics and Computational Science* (c. K. Koç, ed.), pp. 275–315, Cham: Springer International Publishing, 2014.

[74] P. Lacharme, "Post-Processing Functions for a Biased Physical Random Number Generator," in *Fast Software Encryption* (K. Nyberg, ed.), vol. 5086, pp. 334–342, Berlin, Heidelberg: Springer Berlin Heidelberg, 2008.

[75] P. Lacharme, "Analysis and Construction of Correctors," *IEEE Transactions on Information Theory*, vol. 55, pp. 4742–4748, Oct. 2009.

[76] V. Fischer and D. Lubicz, "Embedded Evaluation of Randomness in Oscillator Based Elementary TRNG," 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 O. Pastor, eds.), vol. 7908, pp. 527–543, Berlin, Heidelberg: Springer Berlin Heidelberg, 2014.

[77] T. Chaney and C. Molnar, "Anomalous Behavior of Synchronizer and Arbiter Circuits," *IEEE Transactions on Computers*, vol. C-22, pp. 421–422, Apr. 1973.

[78] L. M. Reyneri, D. Del Corso, and B. Sacco, "Oscillatory metastability in homogeneous and inhomogeneous flip-flops," *IEEE Journal of Solid-State Circuits*, vol. 25, pp. 254–264, Feb. 1990.

[79] R. Land, "Power-up behavior of clocked devices," Application report SCHA005A, Texas Instruments, February 2015.

[80] M. H. Tooley, *Electronic circuits: fundamentals and applications*. London: Routledge, fourth edition ed., 2015.

[81] N. Da Dalt and A. Sheikholeslami, *Understanding Jitter and Phase Noise: A Circuits and Systems Perspective*. Cambridge University Press, 1 ed., feb 2018.

[82] V. Fischer and M. Drutarovsky, "True Random Number Generator Embedded in Reconfigurable Hardware," in *Cryptographic Hardware and Embedded Systems - CHES 2002*, vol. 2523 of *LNCS*, pp. 415–430, Redwood Shores, CA, USA, Springer Verlag, 2002.

[83] 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, pp. 109–119, Jan. 2007.

[84] A. Cherkaoui, V. Fischer, A. Aubert, and L. Fesquet, "Comparison of Self-Timed Ring and Inverter Ring Oscillators as entropy sources in FPGAs," in *2012 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, (Dresden), pp. 1325–1330, IEEE, Mar. 2012.

[85] S. C. Bagal, V. V. Deotare, D. V. Padole, and S. C. Bagal, "Generation of True Random Number using analog Phase Locked Loop," in *International Journal of Advanced Research in Electronics and Communication Engineering (IJARECE)*, pp. 1913–1916, 2015.

[86] E. Friedman, "Clock distribution networks in synchronous digital integrated circuits," *Proceedings of the IEEE*, vol. 89, pp. 665–692, May 2001.

[87] W. Maichen, *Digital timing measurements: from scopes and probes to timing and jitter*. No. 33 in Frontiers in electronic testing, Dordrecht: Springer, 2006. OCLC: ocm70840961.

[88] ITU, "ITU-T Recommendation G.810: Definitions and Terminology for Synchronization Networks," tech. rep., International Telecommunication Union, 1996.

[89] A. Demir, A. Mehrotra, and J. Roychowdhury, "Phase noise in oscillators: a unifying theory and numerical methods for characterization," *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, vol. 47, pp. 655–674, May 2000.

[90] T. C. Weigandt, B. Kim, and P. R. Gray, "Analysis of timing jitter in CMOS ring oscillators," in *Proceedings of IEEE International Symposium on Circuits and Systems - ISCAS '94*, vol. 4, (London, UK), pp. 27–30, IEEE, 1994.

[91] JEDEC, "Definition of skew specifications for standard logic devices," tech. rep., JEDEC Standard, 2003.

[92] F. Bernard, V. Fischer, and B. Valtchanov, "Mathematical Model of Physical RNGs Based on Coherent Sampling," *Tatra Mountains Mathematical Publications*, vol. 45, no. 1, pp. 1–14, 2010.

[93] M. Grujic, V. Rozic, B. Yang, and I. Verbauwhede, "A Closer Look at the Delay-Chain based TRNG," in *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*, (Florence), pp. 1–5, IEEE, May 2018.

[94] B. Yang, V. Rožic, M. Grujic, N. Mentens, and I. Verbauwhede, "ES-TRNG: A High-throughput, Low-area True Random Number Generator based on Edge Sampling," *IACR*

Transactions on Cryptographic Hardware and Embedded Systems, Volume 2018, Issue 3, pp. 267–292, Aug. 2018.

[95] 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, 2011.

[96] B. Ham, "Fibre Channel - Methodologies for Jitter and Signal Quality Specification (working draft)," tech. rep., International Committee for Information Technology Standardization (INCITS), December 2003.

[97] 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), pp. 1–6, IEEE, Apr. 2008.

[98] D. S. L. D. S. L. Cardwell, *From Watt to Clausius : the rise of thermodynamics in the early industrial age*. London : Heinemann Educational, 1971. Includes bibliographical references.

[99] J. C. Príncipe, *Information theoretic learning: Renyi's entropy and kernel perspectives*. Information science and statistics, New York: Springer, 2010.

[100] C. E. Shannon, "A Mathematical Theory of Communication," *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 5, pp. 3–55, 1948.

[101] A. Rényi, "On measures of entropy and information," in *Proceedings of the Fourth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Contributions to the Theory of Statistics*, (Berkeley, California), pp. 547–561, 1961.

[102] R. V. Hogg, E. A. Tanis, and D. L. Zimmerman, *Probability and statistical inference*. Boston: Pearson, ninth edition ed., 2015.

[103] G. E. Crooks, "On Measures of Entropy and Information," January 2015.

[104] Karmeshu and N. R. Pal, "Uncertainty, Entropy and Maximum Entropy Principle - An Overview," in *Entropy Measures, Maximum Entropy Principle and Emerging Applications* (Karmeshu and J. Kacprzyk, eds.), vol. 119 of *Studies in Fuzziness and Soft Computing*, (Berlin, Heidelberg), Springer Berlin Heidelberg, 2003.

[105] R. V. L. Hartley, "Transmission of information," *The Bell System Technical Journal*, vol. 7, pp. 535–563, July 1928.

[106] R. M. Gray, *Entropy and information theory*. New York: Springer, 2nd ed ed., 2011. OCLC: ocn669910367.

[107] H. Fisser, "Some Tests Applied to Pseudo-Random Numbers Generated by v. Hoerner's Rule," *Numerische Mathematik*, vol. 3, pp. 247–249, dec 1961.

[108] T. E. Tkacik, "A Hardware Random Number Generator," in *Cryptographic Hardware and Embedded Systems - CHES 2002* (G. Goos, J. Hartmanis, J. van Leeuwen, B. S. Kaliski, c. K. Koç, and C. Paar, eds.), vol. 2523, pp. 450–453, Berlin, Heidelberg: Springer Berlin Heidelberg, 2003.

[109] K. Wold and C. H. Tan, "Analysis and Enhancement of Random Number Generator in FPGA Based on Oscillator Rings," in *2008 International Conference on Reconfigurable Computing and FPGAs*, (Cancun, Mexico), pp. 385–390, IEEE, Dec 2008.

[110] G. K. Kanji, *100 statistical tests*. London ; Thousand Oaks, Calif: Sage Publications, 3rd ed ed., 2006.

[111] H. R. Neave, "The teaching of hypothesis-testing," *Bulletin in Applied Statistics*, vol. 3, pp. 55–63, 1976.

[112] G. J. Privitera, *Statistics for the Behavioral Sciences*. SAGE Publications, 2017.

[113] A. Rukhin, J. Soto, J. Nechvatal, M. Smid, E. Barker, S. Leigh, M. Levenson, M. Vangel, D. Banks, A. Heckert, J. Dray, and S. Vo, "A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications," tech. rep., National Institute of Standards and Technology, 2010. Special Publication (NIST SP) - 800-22 Rev 1a.

[114] D. Allan and J. Barnes, "A Modified "Allan Variance" with Increased Oscillator Characterization Ability," in *Thirty Fifth Annual Frequency Control Symposium*, pp. 470–475, IEEE, 1981.

[115] G. Marsaglia, "The marsaglia random number cdrom, with the diehard battery of tests of randomness," tech. rep., Florida State University under a grant from The National Science Foundation, 1985. accessed on 16 November 2018.

[116] J. Walker, "ENT: A Pseudorandom Number Sequence Test Program." http://www.fourmilab.ch/random/, January 2008.

[117] U. M. Maurer, "A universal statistical test for random bit generators," *J. Cryptol.*, vol. 5, pp. 89–105, Mar. 1992.

[118] Y. Wang and T. Nicol, "On statistical distance based testing of pseudo random sequences and experiments with PHP and Debian OpenSSL," *Computers & Security*, vol. 53, pp. 44–64, Sept. 2015.

[119] X. Li, A. B. Cohen, T. E. Murphy, and R. Roy, "Scalable parallel physical random number generator based on a superluminescent LED," *Optics Letters*, vol. 36, p. 1020, Mar. 2011.

[120] F. Rodríguez-Henríquez, A. D. Pérez, N. A. Saqib, and c. K. Koç, "Cryptographic Algorithms on Reconfigurable Hardware," *Springer US*, p. 384, 2007.

[121] M. Dichtl, "How to Predict the Output of a Hardware Random Number Generator," in *Cryptographic Hardware and Embedded Systems - CHES 2003* (G. Goos, J. Hartmanis, J. van Leeuwen, C. D. Walter, c. K. Koç, and C. Paar, eds.), vol. 2779, pp. 181–188, Berlin, Heidelberg: Springer Berlin Heidelberg, 2003.

[122] C. Hochberger, C. Li, M. Raitza, and M. Vogt, "Influence of operating conditions on ring oscillator-based entropy sources in FPGAs," in *22nd International Conference on Field Programmable Logic and Applications (FPL)*, (Oslo, Norway), pp. 555–558, IEEE, Aug. 2012.

[123] G. R. Cooper and C. D. McGillem, *Probabilistic Methods of Signal and System Analysis*. Oxford University Press, 3rd ed., 1998.

[124] G. Žitković, "Introduction to stochastic processes - lecture notes." https://www.ma.utexas.edu/users/gordanz/notes/introduction_to_stochastic_processes.pdf, December 2010. accessed 17/01/2018.

[125] W. B. Davenport and W. L. Root, *An Introduction to the Theory of Random Signals and Noise*. New York: IEEE Press [u.a.], 1987. OCLC: 256317594.

[126] Y. Yamamoto, *Fundamentals of Noise Processes*. Cambridge University Press, 2017.

[127] P. R. Babu, *Probability Theory and Random Processes*. McGraw Hill Education, 2015.

[128] A. Bhargava, "On the Theory of Testing for Unit Roots in Observed Time Series," *The Review of Economic Studies*, vol. 53, no. 3, pp. 369–384, 1986.

[129] G. Nason, "A test for second-order stationarity and approximate confidence intervals for localized autocovariances for locally stationary time series," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 75, pp. 879–904, 11 2013.

[130] D. Kwiatkowski, P. C. Phillips, P. Schmidt, and Y. Shin, "Testing the null hypothesis of stationarity against the alternative of a unit root: How sure are we that economic time series have a unit root?," *Journal of econometrics*, vol. 54, no. 1-3, pp. 159–178, 1992.

[131] J. A. Barnes, A. R. Chi, L. S. Cutler, D. J. Healey, D. B. Leeson, T. E. McGunigal, J. A. Mullen, W. L. Smith, R. L. Sydnor, R. F. C. Vessot, and G. M. R. Winkler, "Characterization of frequency stability," *IEEE Transactions on Instrumentation and Measurement*, vol. IM-20, pp. 105–120, May 1971.

[132] W. J. Riley, *Handbook of Frequency Stability Analysis*, vol. 1065 of *NIST special publication*. U.S. Department of Commerce, National Institute of Standards and Technology, July 2008.

[133] E. Rubiola, *Phase noise and frequency stability in oscillators*. The Cambridge RF and microwave engineering series, Cambridge, UK ; New York: Cambridge University Press, 2009. OCLC: 227031868.

[134] J. Rutman, "Characterization of phase and frequency instabilities in precision frequency sources: Fifteen years of progress," *Proceedings of the IEEE*, vol. 66, no. 9, pp. 1048–1075, 1978.

[135] F. Vernotte, "Stabilité temporelle et fréquentielle des oscillateurs : modèles," *Techniques de l'ingénieur Métrologie temps-fréquence*, vol. base documentaire : TIB415DUO., no. ref. article : r680, p. 13, 2006.

[136] P. Prandoni and M. Vetterli, *Signal processing for communications*. Communication and information sciences, Lausanne: EPFL Press, 2008. OCLC: 635543346.

[137] J.-M. Bony, *Cours d'Analyse: Téorie des Distributions et Analyse de Fourier*. Éditions de l'École Polytechnique, 2001.

[138] A. Lesfari, *Distributions, Analyse de Fourier et transformation de Laplace*. Ellipses, 2012.

[139] J. D. Hamilton, *Time Series Analysis*. Princeton University Press, hardcover ed., 1994.

[140] L. Koralov and Y. G. Sinai, *Theory of Probability and Random Processes*. Universitext, Berlin, Heidelberg: Springer Berlin Heidelberg, 2007.

[141] M. L. Stein, *Interpolation of spatial data: some theory for kriging.* Place of publication not identified: Springer, 1999. OCLC: 830022093.

[142] J. R. Barry, E. A. Lee, and D. G. Messerschmitt, *Digital Communication*. Springer US, 2004.

[143] J. A. Barnes and S. J. Jarvis, "Efficient numerical and analog modeling of flicker noise processes," Tech. Rep. 604, National Bureau of Standards, Gaithersburg, MD, 1971.

[144] E. Masry, "Flicker noise and the estimation of the Allan variance," *IEEE Transactions on Information Theory*, vol. 37, pp. 1173–1177, July 1991.

[145] H. Tian and A. El Gamal, "Analysis of 1/f noise in CMOS APS," in *Sensors and Camera Systems for Scientific, Industrial, and Digital Photography Applications* (N. Sampat, T. Yeh, M. M. Blouke, N. Sampat, G. M. W. Jr., and T. Yeh, eds.), vol. 3965, (San Jose, CA), pp. 168–176, International Society for Optics and Photonics, SPIE, May 2000.

[146] E. Milotti, "1/f noise: a pedagogical review," *arXiv preprint physics/0204033*, p. 26, 2002.

[147] J. J. Benedetto, S. E. Scott, and R. Kerby, "A Wiener–Wintner theorem for 1/f power spectra," *Journal of Mathematical Analysis and Applications*, vol. 279, pp. 740–755, March 2003.

[148] J. P. Gleeson, "Exactly solvable model of continuous stationary 1/f noise," *Physical Review E*, vol. 72, p. 7, July 2005.

[149] N. J. Kasdin and T. Walter, "Discrete simulation of power law noise (for oscillator stability evaluation)," in *Proceedings of the 1992 IEEE Frequency Control Symposium*, pp. 274–283, IEEE, May 1992.

[150] H. Thielemann, "Sampling-rate-aware noise generation," *arXiv:1103.4118 [cs]*, p. 8, Mar. 2011. arXiv: 1103.4118.

[151] C. A. Greenhall, "FFT-Based Methods for Simulating Flicker FM," in *Proceedings of the 34th Annual Precise Time and Time Interval (PTTI) Systems and Applications Meeting*, p. 12, 2002.

[152] N. Ashby, "Discrete simulation of power law noise," *arXiv:1103.5062 [physics]*, p. 25, Mar. 2011. arXiv: 1103.5062.

[153] P. Uhrich, "Stabilité des Oscillateurs ultra-stables." Journées X-ENSUPS de physique, 2007.

[154] D. Allan, "Statistics of Atomic Frequency Standards," *Proceedings of the IEEE*, vol. 54, no. 2, pp. 221–230, 1966.

[155] D. Allan, "Time and Frequency (Time-Domain) Characterization, Estimation, and Prediction of Precision Clocks and Oscillators," *IEEE Transactions on Ultrasonics, Ferroelectrics and Frequency Control*, vol. 34, pp. 647–654, Nov. 1987.

[156] A. van der Ziel, "Flicker Noise in Electronic Devices," in *Advances in Electronics and Electron Physics*, vol. 49, pp. 225–297, Elsevier, 1979.

[157] J. Barnes, "Atomic Timekeeping and the Statistics of Precision Signal Generators," *Proceedings of the IEEE*, vol. 54, no. 2, pp. 207–220, 1966.

[158] J. Snyder, "An Ultra-High Resolution Frequency Meter," in *Thirty Fifth Annual Frequency Control Symposium*, pp. 464–469, IEEE, May 1981.

[159] D. Howe, D. Allan, and J. Barnes, "Properties of Signal Sources and Measurement Methods," in *Thirty Fifth Annual Frequency Control Symposium*, pp. 669–716, IEEE, 1981.

[160] F. Vernotte, "Stabilité temporelle et fréquentielle des oscillateurs : outils d'analyse," *Techniques de l'ingénieur Métrologie temps-fréquence*, vol. base documentaire : TIB415DUO., no. ref. article : r681, p. 13, 2006. fre.

[161] O. Petura, M. Laban, E. Noumon Allini, and V. Fischer, "Two Methods of the Clock Jitter Measurement Aimed at Embedded TRNG Testing," in *TRUDEVICE – 8th Conference on Trustworthy Manufacturing and Utilization of Secure Devices (TRUDEVICE 2018), Dresden, Germany*, p. 5, March 2018.

[162] W. F. Sheppard, "On the Calculation of the most Probable Values of Frequency-Constants, for Data arranged according to Equidistant Division of a Scale," *Proceedings of the London Mathematical Society*, vol. s1-29, no. 1, pp. 353–380, 1897.

[163] P. Haddad, Y. Teglia, F. Bernard, and V. Fischer, "On the assumption of mutual independence of jitter realizations in P-TRNG stochastic models," in *2014 Design, Automation Test in Europe Conference Exhibition (DATE)*, pp. 1–6, March 2014.

[164] M. Laban, M. Drutarovsky, V. Fischer, and M. Varchola, "Platform for testing and evaluation of PUF and TRNG implementations in FPGAs," in *TRUDEVICE – 6th Conference on Trustworthy Manufacturing and Utilization of Secure Devices (TRUDEVICE 2016), Barcelona, Spain*, Nov 2016.

[165] B. Razavi, K. F. Lee, and R. H. Yan, "Design of high-speed, low-power frequency dividers and phase-locked loops in deep submicron CMOS," *IEEE Journal of Solid-State Circuits*, vol. 30, pp. 101–109, Feb 1995.

[166] L. DeVito, J. Newton, R. Croughwell, J. Bulzacchelli, and F. Benkley, "A 52mhz And 155mhz Clock-recovery PLL," in *1991 IEEE International Solid-State Circuits Conference. Digest of Technical Papers*, (San Francisco, CA, USA), pp. 142–306, IEEE, Feb 1991.

[167] J. Alvarez, H. Sanchez, G. Gerosa, and R. Countryman, "A Wide-Bandwidth Low-Voltage PLL for PowerPCTMMicroprocessors," *IEEE Journal of Solid-State Circuits*, vol. 30, pp. 383–391, April 1995.

[168] V. F. Kroupa, *Phase lock loops and frequency synthesis*. Chichester: Wiley, 2003. OCLC: 248801658.

[169] W. J. Gruen, "Theory of AFC Synchronization," *Proceedings of the IRE*, vol. 41, pp. 1043–1048, Aug 1953.

[170] A. Fahim, *Clock Generators for SOC Processors - Circuits and Architectures*. Springer-Verlag US, 2005.

[171] R. E. Best, *Phase-locked loops: design, simulation and applications*. McGraw-Hill professional engineering, New York, NY: McGraw-Hill, 5. ed ed., 2003. OCLC: 249285118.

[172] F. M. Gardner, *Phaselock Techniques*. John Wiley & Sons, 2005.

[173] J. W. Bergmans, *Digital baseband transmission and recording*. Springer Science & Business Media, 1996.

[174] D. H. Wolaver, *Phase-Locked Loop Circuit Design*. Englewood Cliffs, New Jersey: Prentice Hall, 1st ed., February 1991.

[175] D. R. Stephens, *Phase-Locked Loops for Wireless Communications: Digital, Analog and Optical Implementations*. Kluwer Academic Publishers, 2nd ed., November 2001.

[176] M. Correvon, "Boucles à verrouillage de phase." https://www.chireux.fr/mp/cours/electronique/Chap13.pd September 2008. accessed 04/10/2019.

[177] B. Razavi, *Design of Analog CMOS Integrated Circuits*. McGraw Hill Higher Education, 1st ed., September 2000.

[178] D. Talbot, *Frequency acquisition techniques for phase locked loops*. Hoboken, New Jersey: John Wiley & Sons, Inc, 2012.

[179] K. J. Åström and R. M. Murray, *Feedback systems: an introduction for scientists and engineers*. Princeton: Princeton University Press, 2008. OCLC: ocn183179623.

[180] W. F. Egan, *Phase-Lock Basics*. Hoboken, NJ, USA: John Wiley & Sons, Inc., 2 ed., oct 2007.

[181] N. Bochard, C. Marchand, O. Pet'ura, L. Bossuet, and V. Fischer, "Evariste III: A new multi-FPGA system for fair benchmarking of hardware dependent cryptographic primitives." Workshop on Cryptographic Hardware and Embedded Systems, CHES 2015, September 2015. poster.

[182] A. Hajimiri and T. Lee, *The Design of Low Noise Oscillators*. Kluwer academic publishers, 2003.

[183] Teledyne LeCroy, *WavePro 7 Zi-A Series*, February 2017.

[184] Intel, "Possible Causes for PLL Loss of Lock," 2020.

[185] Silicon Labs, *Crystal Oscillator (XO) (10 MHz to 1.4 GHz)*, 2018.

[186] C. Liu and J. A. McNeill, "A digital-pll-based true random number generator," in *Research in Microelectronics and Electronics, 2005 PhD*, vol. 1, pp. 113–116, July 2005.

[187] V. Fischer, F. Bernard, and N. Bochard, "Modern random number generator design – Case study on a secured PLL-based TRNG," *it - Information Technology*, vol. 61, pp. 3–13, feb 2019.

[188] Altera corporation, *Cyclone V Device Datasheet (CV51002)*, 2015.

[189] Xilinx, *Spartan-6 FPGA Clocking Resources*, 2015.

[190] Intel, 101 Innovation Drive, San Jose, CA 95134, *Cyclone V Device Datasheet*, December 2016.

[191] O. Petura, U. Mureddu, N. Bochard, and V. Fischer, "Optimization of the PLL Based TRNG Design Using the Genetic Algorithm," in *IEEE International Symposium on Circuits and Systems, ISCAS*, pp. 2202–2205, 2017.

[192] Microsemi, *SmartFusion2 and IGLOO2 Clocking Resources*, 2015.

[193] 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, Volume 2018, Issue 3*, pp. 214–242, Aug. 2018.

[194] E. Noumon Allini, O. Petura, V. Fischer, and F. Bernard, "Optimization of the PLL configuration in a PLL-based TRNG design," in *2018 Design, Automation Test in Europe Conference Exhibition (DATE)*, pp. 1265–1270, IEEE, Mar. 2018.

[195] E. Noumon Allini, F. Bernard, and V. Fischer, "An Illustration of a New Certification Approach for TRNGs," in *Workshop on Cryptographic Architectures Embedded in Logic Devices, Cryptarchi 2017*, June 2017.

[196] J. A. Barnes and D. W. Allan, "Variances based on data with dead time between the measurements," Tech. Rep. NBS TN 1318, National Bureau of Standards, Gaithersburg, MD, 1990.

[197] D. W. Allan, "Should the Classical Variance Be Used As a Basic Measure in Standards Metrology?," *IEEE Transactions on Instrumentation and Measurement*, vol. IM-36, pp. 646–654, June 1987.

Appendix A

LTI and random processes

In this thesis, various systems dealing with random signals have been considered and modeled as linear time-invariant (LTI) systems. This consideration comes from the field of automation which is used to deal with deterministic signals rather than random ones as it is the case in this thesis. We therefore study here, in the general case, the response of a LTI system to a random input. This appendix aims at helping the reader to understand the use we made of such methods.

A.1 Introduction to linear time-invariant systems

A linear time-invariant (LTI) system is a theoretical system which has two basic properties: linearity and time-invariance. The linearity implies the principle of superposition, which yields both laws of additivity and homogeneity. Hence, a system $\mathscr{H}$ would be linear if for any scalar α and any signals $x(t)$ and $y(t)$, the following holds:

$$\mathscr{H}[\alpha x(t) + y(t)] = \alpha \mathscr{H}[x(t)] + \mathscr{H}[y(t)]. \tag{A.1}$$

The time-invariance of the system $\mathscr{H}$ means that any delay in the input would be reflected in the output. Thus, for any $r \in \mathbb{R}_+$ and any signals $x(t)$ and $y(t)$, one has:

$$y(t) = \mathscr{H}[x(t)] \quad \Longrightarrow \quad y(t-r) = \mathscr{H}[x(t-r)]. \tag{A.2}$$

LTI systems are used in various fields, like in automation, where there serve as models for the description of physical systems. They are usually represented as in Figure A.1, where $x(t)$ is the input signal to the system and $y(t)$ is its output signal.

When the input signal $x(t)$ is an impulse function[1], the output is given the name of the impulse response function of the system, and denoted $h(t)$. The notion of impulse response is an impor-

[1] The impulse function is mathematically represented by a Dirac δ function, usually taken at the time origin $t = 0$.

Figure A.1: Representation of a linear time-invariant system.

tant one when dealing with LTI systems. Indeed, any LTI system is characterized by its impulse response, it therefore makes it easier to analyze such systems.

The impulse response actually characterizes LTI systems in the time domain. However, it is often practical to study systems in the frequency domain. In this case, the system is characterized by the Laplace transform[2] of its impulse response. This Laplace transform is called the transfer function of the LTI and is often denoted $H(s)$. Hence, one has:

$$H(s) := \int_{-\infty}^{+\infty} h(t)e^{-st}dt. \tag{A.3}$$

Note that the use of impulse response and transfer function is only relevant in the case of LTI systems.

In this thesis, we assume that LTI systems we are dealing with are stable systems with are stable in the BIBO (bounded-input/bounded-output) sense. It means that every bounded input applied to the system results in a bounded output. The physical realizability and stability constraints on the LTI system considered here are given by:

$$\forall t < 0, \quad h(t) = 0 \tag{A.4}$$

and

$$\int_{-\infty}^{+\infty} |h(t)|\,dt < +\infty. \tag{A.5}$$

A.2 Response of LTI to random input

From Section A.1, we know that a LTI system produces an output signal following a signal applied to its input. This output signal is called the LTI response to the input signal. In this section, we are interested in the response of a LTI when the input is a random signal, especially when it is a stationary process.

A.2.1 Analysis in the time domain

Let $\{x(t)\}$ be the input to the system and $\{y(t)\}$ be its output. Then, $x(t)$ and $y(t)$ are related by:

$$y(t) = \int_{-\infty}^{+\infty} x(t-\lambda)h(\lambda)d\lambda. \tag{A.6}$$

[2]Some authors consider the Fourier transform. Since the Laplace transform generalizes the Fourier transform, these two approaches are valid.

Note that $y(t)$ can also be defined as:

$$y(t) = \int_{-\infty}^{+\infty} x(\lambda)h(t-\lambda)d\lambda, \tag{A.7}$$

because the convolution is commutative.

A.2.1.1 Expected value of y

The expected value of the system's output, at a given time t, can be computed as:

$$\langle y(t)\rangle = \left\langle \int_{-\infty}^{+\infty} x(t-\lambda)h(\lambda)d\lambda \right\rangle = \int_{-\infty}^{+\infty} \langle x(t-\lambda)\rangle\, h(\lambda)d\lambda. \tag{A.8}$$

Since $\{x(t)\}$ is a stationary process, one can write:

$$\langle x(t-\lambda)\rangle = \langle x(t)\rangle = \langle x\rangle, \tag{A.9}$$

which is independent of t. This allows to write:

$$\langle y(t)\rangle = \int_{-\infty}^{+\infty} \langle x\rangle h(\lambda)d\lambda = \langle x\rangle \int_{-\infty}^{+\infty} h(\lambda)d\lambda, \tag{A.10}$$

which shows that the expected value of $y(t)$ does not depend on the time t.

From Equation (A.3), it follows:

$$H(0) = \int_{-\infty}^{+\infty} h(\lambda)d\lambda, \tag{A.11}$$

thus:

$$\langle y\rangle = \langle x\rangle \cdot H(0). \tag{A.12}$$

A.2.1.2 Mean-square value of y

The mean-square value of y can be computed as:

$$\langle y^2(t)\rangle = \left\langle \int_{-\infty}^{+\infty} x(t-\lambda)h(\lambda)d\lambda \int_{-\infty}^{+\infty} x(t-\mu)h(\mu)d\mu \right\rangle \tag{A.13}$$

$$= \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} \langle x(t-\lambda)x(t-\mu)\rangle h(\lambda)h(\mu)d\mu d\lambda. \tag{A.14}$$

By the change of variable $t' = t-\lambda$, one can write:

$$\begin{aligned}\langle x(t-\lambda)x(t-\mu)\rangle &= \langle x(t')x(t'+\lambda-\mu)\rangle \\ &= R_x(\lambda-\mu),\end{aligned}$$

where $R_x(t)$ is the autocorrelation function of the stationary process $\{x(t)\}$. It then follows that the integrand is independent of time and thus:

$$\langle y^2\rangle = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} R_x(\lambda-\mu)h(\lambda)h(\mu)d\mu d\lambda. \tag{A.15}$$

A.2.1.3 Autocorrelation function of y

The autocorrelation function of the output y can be computed as:

$$\begin{aligned} R_y(\tau) &= \langle y(t)y(t+\tau)\rangle && \text{(A.16)}\\ &= \left\langle \int_{-\infty}^{+\infty} x(t-\lambda)h(\lambda)d\lambda \int_{-\infty}^{+\infty} x(t+\tau-\mu)h(\mu)d\mu \right\rangle && \text{(A.17)}\\ &= \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} \langle x(t-\lambda)x(t+\tau-\mu)\rangle h(\lambda)h(\mu)d\mu d\lambda. && \text{(A.18)} \end{aligned}$$

By the change of variable $t' = t-\lambda$, one can write:

$$\begin{aligned} \langle x(t-\lambda)x(t+\tau-\mu)\rangle &= \langle x(t')x(t'+\lambda+\tau-\mu)\rangle && \text{(A.19)}\\ &= R_x(\lambda+\tau-\mu). && \text{(A.20)} \end{aligned}$$

It follows that the autocorrelation function of y is given by:

$$R_y(\tau) = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} R_x(\lambda+\tau-\mu)h(\lambda)h(\mu)d\mu d\lambda. \quad \text{(A.21)}$$

A.2.2 Analysis in the frequency domain

The most common method for representing linear systems in the frequency domain is the transfer function which can be expressed either in the Fourier domain as $H(\omega)$, or in the Laplace domain $H(s)$. The function $H(\omega)$ (respectively$H(s)$) is the Fourier (respectively the Laplace) transform of the system impulse response $h(t)$. In frequency domain, the input $\{x(t)\}$ and output $\{y(t)\}$ of LTI system are related by:

$$Y(\omega) = H(\omega)X(\omega), \quad \text{(A.22)}$$

where $X(\omega)$ (respectively $Y(\omega)$) is the Fourier transform of $x(t)$ (respectively $y(t)$).

Since $\omega = 2\pi f$, Equation (A.22) can be expressed in term of f as:

$$Y(f) = H(f)X(f). \quad \text{(A.23)}$$

In Laplace domain, the relation between input and output is as follows:

$$Y(s) = H(s)X(s). \quad \text{(A.24)}$$

Since $\{x(t)\}$ is a stationary random process, the same goes for $\{y(t)\}$ and thus one can compute its

power spectral density as:

$$\begin{aligned} S_y(\omega) &= \int_{-\infty}^{+\infty} R_y(\tau) e^{-i\omega\tau} d\tau \\ &= \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} R_x(\lambda+\tau-\mu)h(\lambda)h(\mu)e^{-i\omega\tau} d\mu d\lambda d\tau \\ &= \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} R_x(\tau)h(\lambda)h(\mu)e^{-i\omega(\tau+\mu-\lambda)} d\mu d\lambda d\tau \\ &= \int_{-\infty}^{+\infty} h(\lambda)e^{i\omega\lambda} d\lambda \int_{-\infty}^{+\infty} h(\mu)e^{-i\omega\mu} d\mu \int_{-\infty}^{+\infty} R_x(\tau)e^{-i\omega\tau} d\tau \\ &= H(-\omega)H(\omega)S_x(\omega) \\ &= |H(\omega)|^2 S_x(\omega), \end{aligned} \tag{A.25}$$

where $\omega \longmapsto |H(\omega)|^2$ is called power transfer function.

Equation (A.25) can be also expressed as:

$$S_y(f) = |H(f)|^2 S_x(f). \tag{A.26}$$

In terms of Laplace variable, Equation (A.25) becomes:

$$S_y(s) = H(s)H(-s)S_x(s). \tag{A.27}$$

Appendix B

Dead time between successive measurements

In this thesis, we adopted a formalism which assumes that data was acquired in a continuous way. This assumption was made first because we used equipments which were able to measure output signals of oscillators in a continuously. Moreover, the measuring device was not stopped during data acquisition. However, practitioners do not always proceed that way since some intentionally wait between measurements, creating the phenomenon of dead time between consecutive measurements, as illustrated in Figure B.1.

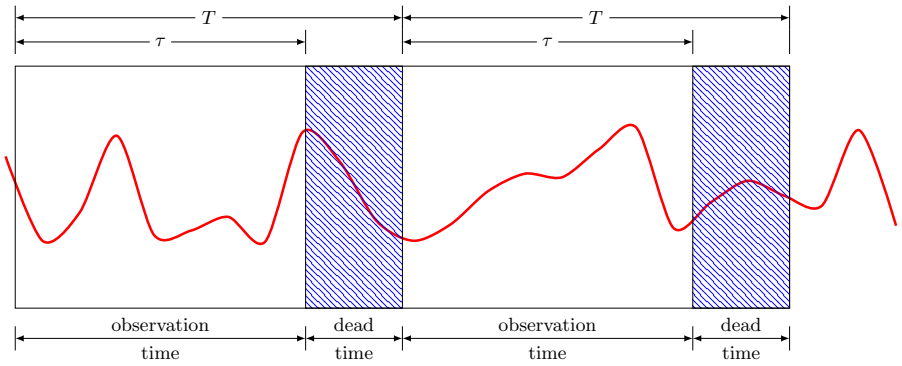


Figure B.1: Illustration of dead time between successive measurements.

Dead time is the (time) gap between consecutive measurements of a signal. It has various origins, most common ones are physical characteristics of the device and measurement method adopted by the experimenter. Physics of the device may cause an inability to capture the next event as soon as the current one ends, yielding a delay between successive measurements. Even if the device creates no dead time, the measurement method adopted by the experimenter can make him deliberately wait between measurements. In either case, this phenomenon has significant effects

on the results, especially for non-white noises. Indeed, it introduces bias, reduces confidence in the results, prevents proper conversion of frequency to phase, etc [132, Section 5.15]. Moreover, no information is available concerning the behavior of the oscillator during the dead time. These are the reasons that explains why we adopted a method that ensures measurements with no dead time.

It is true, however, that in some cases, measurements with dead time are inevitable. In theses cases, it is important to take into account effects of the dead time and apply some corrections [196]. This procedure is nevertheless problematic for data having multiple noise types.

Appendix C

Extensions of the properties of the classical variance to the Allan variance

Because of the limitations of the classical variance, we propose to use the Allan variance to study the sources of randomness in the context of random number generation. This can lead to legitimate questions, especially regarding properties. We will see in this appendix that some properties of the classical varaince can extend to the Allan variance. However, we will limit the discussion to those that are commonly used.

C.1 Allan variance generalizes the classical variance

If x is a stationary and uncorrelated random process, we know that its statistical variance exists [197]. If we call μ the expected value of x, then:

$$\begin{aligned}
\text{avar}(x) &= \frac{1}{2}\mathbb{E}\left[(x_{i+1}-x_i)^2\right] \\
&= \frac{1}{2}\mathbb{E}\left(\left[(x_{i+1}-\mu)-(x_i-\mu)\right]^2\right) \\
&= \frac{1}{2}\mathbb{E}\left[(x_{i+1}-\mu)^2-(x_{i+1}-\mu)(x_i-\mu)+(x_i-\mu)^2\right] \\
&= \frac{1}{2}\mathbb{E}\left[(x_{i+1}-\mu)^2\right]-\frac{1}{2}\mathbb{E}\left[(x_{i+1}-\mu)(x_i-\mu)\right]+\frac{1}{2}\mathbb{E}\left[(x_i-\mu)^2\right] \\
&= \frac{1}{2}\text{var}(x_{i+1})-\frac{1}{2}\mathbb{E}\left[(x_{i+1}-\mu)(x_i-\mu)\right]+\frac{1}{2}\text{var}(x_i). \qquad \text{(C.1)}
\end{aligned}$$

Since x is stationary, one has:

$$\text{var}(x_{i+1}) = \text{var}(x_i) = \text{var}(x) \qquad \text{(C.2)}$$

and:

$$\mathbb{E}(x_{i+1}) = \mathbb{E}(x_i) = \mu. \tag{C.3}$$

Moreover, the uncorrelatedness of x implies:

$$\mathbb{E}(x_{i+1}x_i) = \mathbb{E}(x_{i+1})\mathbb{E}(x_i). \tag{C.4}$$

It then follows:

$$\begin{aligned}\mathbb{E}\left[(x_{i+1}-\mu)(x_i-\mu)\right] &= \mathbb{E}\left(x_{i+1}x_i - \mu x_{i+1} - \mu x_i + \mu^2\right)\\ &= \mathbb{E}(x_{i+1}x_i) - \mu\mathbb{E}(x_{i+1}) - \mu\mathbb{E}(x_i) + \mu^2\\ &= 0.\end{aligned} \tag{C.5}$$

Hence:

$$\operatorname{avar}(x) = \operatorname{var}(x). \tag{C.6}$$

This proves that in the case of a stationary and uncorrelated random process, the classical variance and the Allan variance lead to the same results. This implies that the Allan variance is a generalization of the classical variance.

C.2 Multiplication by a scalar

Given a real number λ and a stationary random process x, then λx is also a stationary random process. Its Allan variance is then:

$$\operatorname{avar}(\lambda x) = \frac{1}{2}\mathbb{E}\left[(\lambda x_{i+1} - \lambda x_i)^2\right] = \lambda^2\frac{1}{2}\mathbb{E}\left[(x_{i+1}-x_i)^2\right] = \lambda^2\operatorname{avar}(x). \tag{C.7}$$

C.3 Sum of independent random processes

If x and y are two independent stationary random processes, one has:

$$\begin{aligned}\operatorname{avar}(x+y) &= \frac{1}{2}\mathbb{E}\left[(x_{i+1}+y_{i+1}-x_i-y_i)^2\right]\\ &= \frac{1}{2}\mathbb{E}\left[(x_{i+1}-x_i+y_{i+1}-y_i)^2\right]\\ &= \frac{1}{2}\mathbb{E}\left[(x_{i+1}-x_i)^2+(x_{i+1}-x_i)(y_{i+1}-y_i)+(y_{i+1}-y_i)^2\right]\\ &= \frac{1}{2}\mathbb{E}\left[(x_{i+1}-x_i)^2\right]+\frac{1}{2}\mathbb{E}[(x_{i+1}-x_i)(y_{i+1}-y_i)]+\frac{1}{2}\mathbb{E}\left[(y_{i+1}-y_i)^2\right]\\ &= \operatorname{avar}(x)+\frac{1}{2}\mathbb{E}[(x_{i+1}-x_i)(y_{i+1}-y_i)]+\operatorname{avar}(y).\end{aligned} \tag{C.8}$$

Because the processes x and y are independent, it follows:

$$\mathbb{E}\left(x_i y_j\right) = \mathbb{E}(x_i)\mathbb{E}\left(y_j\right), \tag{C.9}$$

for any $i, j \in \mathbb{N}$. Since they are stationary:

$$\mathbb{E}\left(x_j\right) = \mathbb{E}(x_i) = \mathbb{E}(x) \quad \text{and} \quad \mathbb{E}\left(y_j\right) = \mathbb{E}(y_i) = \mathbb{E}(y),$$

for any $i, j \in \mathbb{N}$. Hence:

$$\begin{aligned}
\mathbb{E}[(x_{i+1} - x_i)(y_{i+1} - y_i)] &= \mathbb{E}[x_{i+1}y_{i+1} - x_{i+1}y_i - x_i y_{i+1} + x_i y_i] \\
&= \mathbb{E}[x_{i+1}y_{i+1}] - \mathbb{E}[x_{i+1}y_i] - \mathbb{E}[x_i y_{i+1}] + \mathbb{E}[x_i y_i] \\
&= \mathbb{E}[x_{i+1}]\mathbb{E}[y_{i+1}] - \mathbb{E}[x_{i+1}]\mathbb{E}[y_i] - \mathbb{E}[x_i]\mathbb{E}[y_{i+1}] \\
&\quad + \mathbb{E}[x_i]\mathbb{E}[y_i] \\
&= \mathbb{E}[x]\mathbb{E}[y] - \mathbb{E}[x]\mathbb{E}[y] - \mathbb{E}[x]\mathbb{E}[y] + \mathbb{E}[x]\mathbb{E}[y] \\
&= 0.
\end{aligned} \tag{C.10}$$

We can then conclude:

$$\operatorname{avar}(x + y) = \operatorname{avar}(x) + \operatorname{avar}(y). \tag{C.11}$$