Générateurs de suites binaires vraiment aléatoires : modélisation et implantation dans des cibles FPGA

Boyan Valtchanov

Thèse

Présentée à

l'Université Jean Monnet de Saint-Etienne

pour obtenir

le grade de : Docteur de l'Université de Saint-Etienne

Spécialité : Image, Vision et Signal

par

Boyan Valtchanov

et intitulée

Générateurs de suites binaires vraiment aléatoires : modélisation et implantation dans des cibles FPGA

Thèse soutenue le 14 Décembre 2010 devant la commission d'examen :

Président du jury :	Robert Fouquet	Professeur, Université Jean Monnet, Saint-Etienne
Rapporteurs :	Guy Gogniat	Professeur, Université de Bretagne Sud
	Lionel Torres	Professeur, Université de Montpellier 2
Examinateurs :	Werner Schindler	Professeur, Bundesamt für Sicherheit in der Informationstechnik - Allemagne
	Wayne Burleson	Professeur, University of Massachusetts Amherst - USA
	Yannick Teglia	Security and Cryptology Expert, ST Microelectronics
Directeur :	Viktor Fischer	Professeur, Université Jean Monnet, Saint-Etienne
Co-directeur :	Alain Aubert	Maître de conférences, Université Jean Monnet, St-Etienne

Table des matières

Table des figures

Liste des tableaux

Principales notations

σ_X	Ecart type d'une population de mesures X ou données simulées.
μ_X	Valeur moyenne d'une population de mesures X ou données simulées.
$H(X)$	Entropie d'une variable aléatoire X.
$H(X\|Y)$	Entropie conditionnelles de deux variables aléatoires X et Y.
$H(X,Y)$	Entropie conjointe de deux variables aléatoires X et Y.
$B(X)$	Biais statistique d'une suite X dans {0,1}.
$Mi(A,B)$	Information Mutuelle entre les variables aléatoires A et B.
$N(\mu,\sigma^2)$	Loi Normale ou Gaussienne de moyenne μ et variance σ^2.
$\lfloor \frac{A}{B} \rfloor$	Partie entière de la division de A par B.
$mod(X,2)$	Opération mathématique *modulo* 2. Cette opération convertit une suite de nombres en une suite de bits en effectuant la conversion nombre impair='1', nombre pair='0'.
$Prob(X=\omega)$	Probabilité que la variable aléatoire X prenne la valeur ω.
ρ	Coefficient de corrélation.
F_X	Fréquence d'un signal périodique X, exprimée en hertz.
T_X	Période temporelle d'une signal périodique X, exprimée en secondes.
$\bar{X}$	Moyenne sur l'ensemble des n éléments d'une population de mesures $X_1, X_2, X_3, ...X_n$
K, M, G	Exposants au sens électronique, Kilo = 10^3 , Méga = 10^6, Giga = 10^9
Kbs, Mbs	Débits exprimés en bits par seconde : Kbs - Kilobits par seconde, Mbs Mégabits par seconde.

Remerciments

Je tiens à remercier en premier lieu Viktor Fischer de m'avoir accueilli dans l'équipe AMTeS, de m'avoir permis d'aborder les domaines des FPGA et de la conception VHDL à travers cette activité si fascinante qu'est la génération de vrai aléa. Je ne saurais assez exprimer mes remerciements à Alain Aubert qui était mon co-directeur, de m'avoir assisté tout au long de ces 3 années et de toujours avoir été d'un conseil et d'un soutien fort appréciable, merci Alain !

Je voudrais remercier également toute "l'équipe" de m'avoir accueilli et de m'avoir permis de partager ces moments de vie qui resteront à jamais gravés dans ma mémoire, merci à Nathalie Bochard, Florent Bernard, Abdourhamane Idrissa et Lubos Gaspar. Également un remerciement à Corinne Fournier et à Jérôme Gire, merci de m'avoir encouragé et aidé techniquement (LaTeX !). Je tiens à remercier Robert Fouquet d'avoir accepté de présider mon jury de thèse et de toujours avoir été présent par ses conseils. Je voudrais particulièrement remercier Thierry Fournel d'avoir accordé du crédit à mes idées et de m'avoir encouragé tout au long de ma thèse, merci Thierry !

Pendant ces 3 années inoubliables, j'ai eu l'occasion d'enseigner à l'ISTASE (Télécom Saint-Etienne), ce fut pour moi une expérience riche et inoubliable que de passer de "l'autre côté". Je remercie particulièrement Manuel Flury de m'avoir encouragé et de m'avoir donné mes premiers vrais faces à faces avec les étudiants en me confiants la direction de ses heures de TD et de m'avoir accordé la possibilité de réalisation en travaux tutorés. Un très grand merci aux électroniciens, Gilbert Varrenne et Thierry Bru, d'avoir partagé avec moi leur expérience et leur bonne humeur en TP d'électronique !

Je voudrais exprimer toute ma gratitude à Guy Gogniat et Lionel Torres pour avoir accepté d'examiner mon manuscrit de thèse et d'avoir été présents à mon jury de thèse.

Également je tiens à remercier particulièrement Werner Schindler ainsi que Wayne Burleson et Yannick Teglia d'avoir accepté de participer à mon jury de thèse.

Introduction

1 Contexte scientifique

La cryptographie a toujours joué un rôle important dans la vie de tous les jours, tant de façon directe qu'indirecte. Dans l'antiquité, à ses débuts, elle était utilisée principalement à des fins militaires. Le but était alors la communication entre deux parties sans qu'aucune autre ne soit en mesure de participer à cette communication. Durant la première partie du $20^{ème}$ siècle, la cryptographie s'est enrichie des progrès faits en mathématiques et c'est ce qui a donné la cryptographie moderne. La cryptographie moderne a joué un rôle important durant la première et seconde guerre mondiale sur la scène militaire et politique internationale. Cependant, depuis peu, la cryptographie est présente sur la scène civile également. Les réseaux informatiques, les transactions bancaires, les communications numériques ont tous besoin de la sécurité que peut fournir la cryptographie en garantissant une confidentialité de l'information véhiculée. Avec l'avènement des équipements mobiles de plus en plus petits (téléphones cellulaires, GPS, compteurs électriques intelligents, ordinateurs portables, etc...) et surtout des standards de communication sans fil (*wireless*) le besoin de cryptographie est d'autant plus important que la facilité d'interception augmente du fait que le médium de transmission de l'information soit l'air.

Depuis les années 70 environs, la cryptographie est donc sortie des bureaux des services de sécurité des grandes nations et est devenue une discipline scientifique à part entière, étudiée dans le monde entier dans les universités et les laboratoires civils. Des standards de chiffrements, des organismes de certifications sont apparus.

Auguste Kerckhoffs , un célèbre cryptologue néerlandais du $19^{ème}$ siècle, a énoncé le principe, qui maintenant porte son nom et qui stipule que la sécurité d'un système de chiffrement doit reposer uniquement sur le secret de la clé de chiffrement, l'algorithme de chiffrement étant lui, complètement défini et connu par toutes les parties. Bien que dans l'absolu ce principe n'est pas scrupuleusement appliqué (*cf.* standard de chiffrement A5/1 utilisé dans le système de communication cellulaire GSM), il reste un des piliers de la cryptographie moderne. Si ce principe est appliqué et que la sécurité réside en effet dans la confidentialité de la clé alors pour attaquer le système il ne reste qu'à essayer toutes les clés possibles pour trouver celle qui permet de déchiffrer correctement la transmission. C'est ce qui est appelé l'attaque par force brute (*bruteforce attack*). Pour être sûr de son inefficacité, des clés de plus en plus grandes ont été imaginées. Car en même temps l'industrie des semi-conducteurs a fait des progrès impressionnants permettant ainsi à des équipements spécialisés (*cf.* DEEP Crack, Copacobana, etc...)

de retrouver les clés en des temps de plus en plus courts.

Les clés confidentielles de chiffrement sont en l'occurrence des suites de bits, de taille fixe et définies dans les standards de chiffrement(DES, triple DES, AES, RSA, etc...). Ces tailles sont en constante augmentation avec les années qui passent. Pour rendre une attaque exhaustive de recherche de clés (attaque par force brute) plus difficile, on choisit ces clés à partir de suites aléatoires et on s'arrange par la suite de les distribuer grâce à des protocoles de distributions de clés (ex : Kerberos, etc...).

Les systèmes de chiffrement sont de nos jours non seulement logiciels mais aussi de plus en plus embarqués dans les circuits intégrés eux mêmes. Les clés doivent donc être générées dans le matériel. L'utilisation de suites pseudo-aléatoires (des suites de bits d'apparence aléatoire, mais qui en fait sont issues d'équations mathématiques totalement déterministes et donc prévisibles) n'est pas une solution tolérée en cryptographie car ces suites sont généralement périodiques (bien que de périodes très longues) et on peut en théorie facilement les prédire.

La communauté scientifique utilise abondamment les sigles TRNG et PRNG pour désigner un tel générateur. La signification de TRNG étant : *True Random Number Generator* ou générateur de nombres vraiment aléatoires, par opposition à PRNG : *Pseudo Random Number Generator* ou générateur de nombres pseudo-aléatoires, pseudo car ces nombres paraissent aléatoires mais sont en effet issus d'une formule mathématique complètement déterministe et figée. Aussi nous utiliserons dans le cadre de cette thèse abondamment ces abréviations.

Le domaine de recherche des TRNG est difficile d'accès car il s'appuie sur plusieurs domaines : physique, électronique analogique, architecture matérielle, statistiques, mathématiques, cryptographie. De ce fait la communauté scientifique qui s'efforce de proposer des générateurs de bonne qualité est forcément réduite et marginale.

La description mathématique des primitives et protocoles cryptographiques utilise abondamment des suites de nombres aléatoires. Malheureusement les concepteurs souvent théoriciens, ne se posent pas la question de l'obtention de telles suites dans l'implantation du protocole sur une plate-forme matérielle. Ces suites sont principalement utilisées pour la génération de clés de chiffrement, des *challenges*, bits de masquage, *nonces*, bourrages, etc...

2 Problématiques de la thèse

Pour générer des clés de chiffrement dans le matériel, il faut disposer d'une source réellement aléatoire *i.e.* qui dépend d'un processus physique imprévisible. Le bruit électronique, provoqué par le mouvement aléatoire des électrons, qui est omniprésent en électronique mais qui est généralement de faible amplitude peut constituer cette source.

Un autre terme, sera abondamment utilisé dans ce manuscrit, il s'agit du terme *jitter* dont la définition technique sera précisément donnée dans le Chapitre 2. Le mot *jitter* est un substantif anglais dont la signification est : *nervosité* ou *petits tremblements*. Dans un contexte technique on l'utilise pour désigner l'instabilité d'une fréquence ou de la période d'un signal périodique. On rencontre dans la littérature en français le terme *jigue* ou même *jitte*, cependant nous avons préféré utiliser le terme sous sa forme anglaise *jitter* pour des raisons linguistiques.

Il est de plus en plus courant d'utiliser des circuits intégrés programmables et même reprogrammables afin de réduire les coûts de production. Le marché est actuellement dominé par les circuits FPGA ou *Field Programmable Gate Arrays*. Ce sont des circuits numériques contenant plusieurs centaines de milliers de portes élémentaires que le concepteur d'applications peut interconnecter à sa guise, grâce à des outils de conception assistés par ordinateur (CAO) et à des langages de description matérielle HDL (*Hardware Description Languages*). Ces circuits logiques sont très intéressants pour la cryptographie embarquée car ils permettent une flexibilité nécessaire à l'implantation des évolutions. Certains équipements cryptographiques étant produits en petites et moyennes quantités, les FPGA sont les composants idéaux car ils permettent une mise sur le marché rapide et peu coûteuse par rapport à des circuits intégrés spécifiques (*ASIC : Application Specific Integrated Circuits*).

Trouver une source vraiment aléatoire, facilement accessible et commune à un grand nombre de fabricants de FPGA n'est pas une tâche aisée. En effet les FPGA sont destinés à contenir des architectures numériques (*DSP*, processeurs, SoC, etc...) Or ces derniers doivent se comporter de manière complètement déterministe et les sources potentielles d'aléa sont généralement perçues comme nuisibles au bon fonctionnement de ces architectures comme le jitter d'horloge ou la métastabilité des portes logiques. Les fabricants de FPGA s'efforcent donc de les réduire avec la miniaturisation des technologies (*technology scale down*) de fabrication de circuits sur silicium.

Une autre problématique propre à la recherche dans le domaine des TRNG est le fait que contrairement aux algorithmes de chiffrement, qui eux sont déterministes et donc prévisibles et bornés, il n'existe pas de certification reconnue internationalement en ce qui concerne le bon fonctionnement des générateurs vraiment aléatoires. Les outils qui sont utilisés dans la quasi totalité des papiers scientifiques, *i.e.* les tests statistiques, sont inadaptés pour les générateurs vraiment aléatoires. Ces tests ont été conçus pour valider la qualité de suites pseudo-aléatoires et donc pour certifier des PRNG (à l'exception de la norme AIS31 qui sera étudiée).

Une dernière problématique et non des moindres émerge : puisque la source d'aléa doit être de nature physique pour être vraiment aléatoire alors celle-ci est extrêmement sensible aux conditions externes telles que la température de fonctionnement mais aussi et surtout à la qualité de l'environnement du circuit FPGA, c'est à dire de la qualité

de réalisation de la carte PCB qui l'accueille, de la qualité de l'alimentation, de la qualités de composants externes etc... Ainsi un même principe de TRNG sur une même puce FPGA peut fournir des suites aléatoires de qualité différente sur deux cartes PCB différentes !

3 Objectifs de la thèse

C'est pour répondre à tous ces problèmes que nous avons réalisé les travaux exposés dans cette thèse. Nous nous limiterons exclusivement aux circuits logiques programmables de type FPGA. Nous ne nous intéresserons pas aux générateurs pseudo-aléatoires car ceux ci ne sont pas jugés intéressants pour la génération de clés confidentielles pour la cryptographie, bien que leur présence est incontestable dans beaucoup de TRNG dits hybrides combinant un TRNG et un PRNG. Le fonctionnement des PRNG étant totalement déterministe, d'autres outils d'analyse sont nécessaires et ils ne seront donc pas détaillés ici. Nous nous contenterons de signaler leur présence dans le chapitre consacré à l'état de l'art.

Ne seront pas traités également les générateurs (également pseudo-aléatoires) dit GRNG (*Gaussian Random Number Generators*) ou URNG (*Uniform Random Number Generators*) qui sont des générateurs implantables ou pas dans des circuits intégrés et qui produisent des nombres suivant des lois de probabilités prédéfinies (loi normale, loi uniforme, etc...) Ces générateurs sont utilisés en simulation et en communication mais sont inadaptés au domaine couvert par cette thèse qui est la génération de clés pour la cryptographie.

Une méthode d'extraction de vrai aléa fiable est l'amplification directe du bruit électronique. Malheureusement elle est pour le moment inapplicable dans les circuits FPGA car ces derniers sont de nature uniquement numérique.

Un des principaux objectifs de cette thèse est de décrire le TRNG de manière complète et de l'analyser à tous les niveaux et non pas seulement, comme c'est souvent le cas, avec des tests statistiques sur la suite aléatoire produite. Ces tests ne sont d'ailleurs pas adaptés car ils sont destinés à des générateurs pseudo-aléatoires.

Dans cette optique une caractérisation de la source d'aléa est nécessaire.

La modélisation d'un TRNG est un autre objectif de cette thèse. On veut obtenir un modèle et comprendre d'où et surtout comment on obtient l'aléa à la sortie. Ceci afin d'être sur que les suites, qui seront ensuite utilisées pour fournir des clés de chiffrement, sont vraiment aléatoires et ne dépendent pas de phénomènes déterministes prévisibles et manipulables.

Conformément aux recommandations de la norme AIS31 la réalisation d'un TRNG

testable en ligne pendant son fonctionnement sera recherché. Ce TRNG sera capable de sortir un signal d'alarme si la source d'aléa cesse de fonctionner ou si elle est manipulée.

Enfin un objectif très important sur le générateur recherché sera qu'à la sortie les bits présents devront être exclusivement aléatoires et non pas un mélange de bits aléatoires et de bits pseudo-aléatoires. Ceci sous-entends que si jamais la source d'aléa venait à disparaître, la sortie du générateur devra être constante et ne présenter aucune variation.

On vise à réaliser un TRNG qui n'ai pas besoin de correcteurs. En effet, comme nous allons le montrer, très souvent dans la littérature les TRNG sont validés par les suites de tests statistiques après avoir appliqué un post-traitement déterministe qui a pour but de masquer les imperfections du TRNG. Le problème étant que si on évalue la qualité d'une suite produite par un TRNG après l'avoir passée par un post-traitement, la qualité de l'aléa se retrouve masquée par cette fonction de post-traitement qui est déterministe et n'apporte aucun aléa. Ce fait est encore mal compris dans la communauté et les tests statistiques sont utilisés pour valider des TRNG. Nous faisons la différence entre un correcteur et un post-traitement plus sophistiqué néanmoins nous nous efforcerons dans cette thèse d'évaluer les TRNG avant toute sorte de traitement déterministe.

La thèse doit donc répondre à plusieurs questions qui se posent sur la génération d'aléa dans les circuits logiques et précisément dans les FPGA.

Chapitre 1

TRNG dans les circuits logiques - État de l'art

Dans cette partie, un état de l'art et une synthèse bibliographique de la génération d'aléa sur cible logique programmable seront présentés. Dans un premier temps sera décrit l'architecture interne des circuits FPGA (Field Programmable Gate Arrays) et leur environnement direct : banques d'entrée/sortie, alimentations, ressources internes, granularité, etc... C'est en effet de ces paramètres que dépendent les caractéristiques des sources que l'on se propose d'utiliser pour la génération de bits vraiment aléatoires. Le chapitre aborde ensuite la description de générateurs, tels que décrits par la communauté scientifique dans une logique de présentation historique. Bien que les cibles ASIC (Application Specific Integrated Circuits) ne seront pas traités dans le cadre de cette thèse, un survol des TRNG implantés dans ces cibles sera présenté. Historiquement les premiers TRNG étaient implantés dans ce type de cible et par la suite, avec l'avènement des cibles programmables, et leur intérêt particulier pour la cryptographie, ces principes ont été en partie réadaptés pour des cibles reconfigurables (FPGA). Ensuite la description des principes de génération d'aléa spécifiques aux cibles FPGA sera présentée ainsi qu'une méthodologie de classification. Une description des divers principes de correction et de post-traitement généralement utilisés dans la littérature sera présentée. La dernière partie de ce chapitre sur l'état de l'art de la génération d'aléa sera consacrée aux méthodes utilisées pour certifier et quantifier la qualité de suites de bits produits par ces générateurs (i.e. batteries de tests statistiques).

1 Introduction

Les générateurs de nombres aléatoires sont utilisés dans plusieurs domaines et ce terme regroupe un vaste ensemble de dispositifs électroniques. Une catégorisation des différents types de générateurs est présentée à la Figure 1.1. Historiquement les

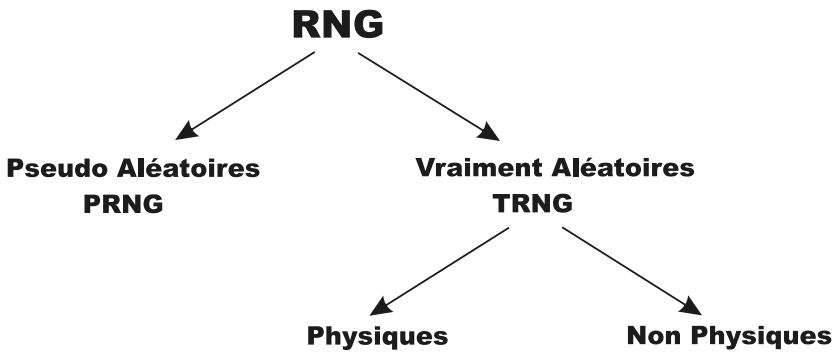

Figure 1.1 Classification des générateurs d'aléa.

premiers recueils de nombres aléatoires sont apparus dans le milieu du 20^{eme} siècle sous la forme de longues listes de nombres aléatoires suivant une certaine loi de probabilité statistique, destinées principalement pour le calcul. A cette époque les dispositifs électroniques n'étaient pas largement disponibles comme maintenant. Leur utilisation était destinée à la résolution de problèmes calculatoires de diverses natures avec les méthodes émergentes à l'époque dites de Monte-Carlo. Ces méthodes reposaient sur un échantillonnage aléatoire pour fournir les résultats, faisant ainsi référence aux jeux de hasard où l'aléatoire est le fondement du jeu. Les problèmes qui peuvent être traités avec ces méthodes sont très variés : calcul numérique d'intégrales, résolution d'équations aux dérivés partielles, finances, simulation, etc... Bien que les premières suites de nombres aléatoires édités par la RAND Corp. aient été produits par un générateur électronique vraiment aléatoire, les méthodes de Monte-Carlo exigent des nombres qui ont seulement l'apparence aléatoire. En effet l'imprévisibilité des nombres n'est pas recherchée, ce qui permet l'utilisation de suites déterministes (*i.e.* décrites à l'aide de formules mathématiques analytiques) qui peuvent être très faciles à obtenir.

Une autre application des générateurs de nombres aléatoires est leur utilisation dans des automates de jeux ou comme outils de décision dans des jeux informatiques. Dans cet exemple l'imprévisibilité de chaque nouveau nombre est importante car on voudrait éviter tout risque qu'un joueur puisse prévoir les actions de l'automate avec lequel il joue souvent de l'argent. Nous appelerons generateur de nombres vraiment aléatoires, ou TRNG en anglais (True Random Number Generator) un générateur qui produit des nombres imprévisibles et dont on ne peut pas décrire le comportement par une équation. Par opposition à un générateur qui produit des nombres qui sont issus de formules déterministes que nous appellerons générateurs de nombres pseudo-aléatoires ou PRNG en anglais (*Pseudo Random Number Generators*).

L'objet d'étude principal de cette thèse, est l'utilisation de nombres aléatoires dans la cryptographie. En effet beaucoup de primitives cryptographiques et même des systèmes de chiffrements entiers reposent sur l'utilisation de nombres aléatoires. Parmi les plus célèbres, citons le chiffrement par masque jetable (*one time pad*) où le message à chiffrer est mélangé à une suite aléatoire caractère par caractère ou bit par bit dans

le cas d'un message numérique. Ce système de chiffrement bien que simple d'apparence est un des plus sûrs qui ont été jamais mis au point. Les deux parties doivent disposer de la même suite de nombres aléatoires pour pouvoir déchiffrer le message. Outre cet inconvénient sur la distribution de la clé de chiffrement (ici la suite aléatoire) l'autre particularité du masque jetable est l'utilisation unique de la suite aléatoire, créant ainsi un besoin de disposer en continu d'une suite de nombres vraiment aléatoires.

Le chiffrement par masque jetable n'est pas le seul à demander de grandes quantités de nombres aléatoires. En effet les clés de chiffrement de plus en plus grandes doivent être générées de manière vraiment aléatoire pour être imprévisibles. Cependant les nombres ainsi générés doivent être de bonne qualité pour pouvoir être utilisés comme clés de chiffrement. Les nombres doivent être statistiquement indépendants les uns des autres et suivre une loi de distribution uniforme tout en dépendant d'un phénomène physique de nature aléatoire.

Certaines applications de haut niveau (au sens informatique) utilisent des phénomènes aléatoires qui ne sont pas de nature physique. Ainsi un système d'exploitation qui ne dispose pas de générateur de nombres aléatoire intégré peut collecter une suite aléatoire en observant le temps entre les frappes des touches du clavier de l'ordinateur ou le temps entre deux accès au disque dur. Ces générateurs ne sont pas très fiables mais sont néanmoins utilisés dans des logiciels où de très petites sources de nombres aléatoires sont nécessaires (PGP, ...).

Les nombres aléatoires sont également utilisés en cryptographie pour la génération de vecteurs d'initialisation, de bits de bourrage, de vecteurs de challenge ou de *nonces*.

Par ailleurs, les FPGA sont de plus en plus complexes et offrent de plus en plus de capacités et de puissance de calcul, ce qui en fait une plate-forme de premier choix pour l'implantation de systèmes de chiffrement car ils offrent une flexibilité incomparable grâce à leur possibilité de reconfiguration. Il existe plusieurs fabricants de FPGA, mais ceux qui sont les plus présents sur le marché et de fait les plus utilisés sont *Xilinx*, *Altera* et *Actel*, avec chacun des familles de produits pour chaque gamme d'applications. Chaque type de FPGA est différent et possède différentes sources potentielles pour la génération d'aléa. Dans la section suivante, nous allons nous intéresser aux sources d'aléa disponibles pour chacune de ces familles.

Nous présenterons dans ce chapitre un état de l'art sous forme de synthèse bibliographique des principaux articles décrivant des générateurs qui peuvent être facilement implantables dans des FPGA. Cette synthèse bibliographique recense les principaux articles scientifiques qui traitent de la génération d'aléa pour la cryptographie. Nous avons développé ceux qui à nos yeux sont les plus importants. Certains TRNG seront juste référencés et ne seront pas décrits en détail car généralement leurs auteurs n'ont pas fourni de description de l'implantation dans les FPGA mais se sont

limités à des simulations et à des implantations dans des cibles ASIC. Nous avons choisi de faire une distinction entre les TRNG pouvant être implantés dans des cibles FPGA et ceux destinés aux cibles ASIC. Les articles consacrés aux TRNG implantables dans les FPGA sont décrits de façon chronologique. Nous présenterons également des critères de classification dans la Section 5 de ce Chapitre et nous comparerons les différents TRNG entre eux selon ces critères. Cependant une évaluation rigoureuse est extrêmement difficile à réaliser car les résultats présentés par les auteurs sont obtenus sur des cibles matérielles différentes (circuits FPGA et cartes support) dans des conditions différentes de fonctionnement. Or comme nous allons le montrer dans le Chapitre 2 de cette thèse, tous ces facteurs ont une très grande influence sur la qualité de l'aléa et donc dans le flux aléatoire obtenu. Pour une telle évaluation, il faudrait disposer d'une plate-forme de test unique (avec différentes cibles FPGA) et évaluer les implantations des différents principes de génération dans les mêmes conditions de fonctionnement. Toute autre approche d'évaluation nous semble inappropriée.

2 Ressources matérielles disponibles dans les différentes cibles FPGA

Les cibles reconfigurables FPGA sont une plate-forme idéale pour le développement de matériel cryptographique car elles permettent une évolution des versions des protocoles de chiffrement ou une mise à jour en cas de découverte de nouvelles attaques. Ces plates-formes sont maintenant en directe concurrence avec les circuits intégrés ASIC car elles embarquent un nombre croissant de fonctionnalités complexes (PLL, RAM, multiplieurs, processeurs, ...).

Cependant, l'implantation de générateurs de bits vraiment aléatoires reste problématique car le concepteur ne dispose pas de toutes les libertés. Ainsi si, par exemple un équilibrage précis des temps de propagation est requis pour le développement d'un TRNG il est nettement plus difficile de le réaliser dans un FPGA que dans un ASIC. De plus, les sources potentiellement utilisables comme source de vrai aléa qui sont le bruit thermique, sa réalisation sur le plan temporel - le jitter ou les états métastables des portes logiques sont logiquement considérés comme des limitations des performances par les constructeurs de FPGA. De ce fait, leur présence et leur accessibilité se trouvent diminuées avec chaque nouvelle famille de FPGA qui sort sur le marché. Dans ce contexte, le développement de TRNG pour ces cibles reste un domaine actif de recherche.

Trois des principaux fabricants de FPGA seront traités dans le cadre de cette thèse pour l'implantation de TRNG : *Xilinx*, *Actel* et *Altera*. Chaque fabricant possède sa propre gamme de produits qui présente ses propres caractéristiques en ce qui concerne les sources utilisables pour la génération d'aléa.

Par exemple l'état métastable d'une bascule, obtenu en violant les temps de *se-*

tup and hold, n'est pas identique sur chaque famille. De plus il n'est pas le même sur une bascule D, qui est généralement implantée dans chaque élément logique du FPGA et une bascule *latch* implantée dans une LUT (*LookUp Table*). Dans le premier cas, les concepteurs se sont efforcés au maximum de réduire la zone d'accessibilité de l'état métastable par la réduction du temps de *setup and hold* alors que dans le second cas ce temps est généralement beaucoup plus important.

Dans le cadre de cette thèse, nous nous arrêterons plus précisément sur le jitter comme source d'aléa. Ainsi le Chapitre 2 montrera que les caractéristiques de celui-ci sont différentes d'une famille à l'autre. Le profil du jitter sera différent s'il est observé dans une PLL dans la famille Actel ou dans une PLL de la famille Altéra ou Xilinx.

Le jitter est fondamentalement une manifestation de l'intégrité du signal. Ainsi il est fortement dépendant de l'environnement du circuit FPGA : alimentation, capacités de découplage, environnement électromagnétique de la carte support et aussi activité du circuit FPGA lui même. Ainsi certaines familles présentent l'avantage de disposer de brochages d'alimentation séparés pour certains éléments internes (comme les PLL par exemple). De cette manière, si un soin particulier est apporté à l'alimentation, on peut garantir un jitter dépendant le moins possible de l'environnement. Généralement les perturbations induites par une mauvaise alimentation sont de nature déterministe donc non aléatoire et peuvent de surcroît être manipulées. Cette problématique, est à notre connaissance jusqu'à présent faiblement traitée par la communauté scientifique au regard de la génération d'aléa.

Puisque les FPGA sont une plate-forme principalement numérique, les sources physiques d'aléa y sont limités. Comme nous allons le montrer dans la Section 4 de ce chapitre, une grande partie des TRNG proposés utilisent d'une manière ou d'une autre le jitter ou l'instabilité dans le temps d'un signal périodique. Or comme nous allons le montrer dans le Chapitre 2, ces jitters sont différents selon l'élément qui produit le signal (oscillateur en anneau, PLL, DFS, oscillateur embarqué, etc...).

Nous présentons une synthèse des différentes sources d'aléa potentielles basées sur le jitter dans les différentes familles de FPGA étudiées dans le cadre de cette thèse dans le Tableau 1.1. Comme on peut le constater, les différentes familles de FPGA disposent de plusieurs sources différentes de signaux jittés. Le plus souvent dans la littérature ce sont les signaux issus d'oscillateurs en anneau qui sont utilisés comme source d'aléa dans les TRNG. Les caractéristiques du jitter présent dans ces oscillateurs ainsi que dans les différents composants de synthèse de fréquences seront étudiés en détail dans le Chapitre 2. Nous avons noté avec un * les cibles FPGA matériellement implantés dans une carte support fournie par la compagnie *Micronic*. Ce sont des cartes support spécialement conçues pour l'évaluation et la comparaison de différents TRNG sur différentes cibles FPGA. Pour les autres cibles présentés dans le Tableau 1.1 ce sont des cartes d'évaluation standard qui ont servi pour accueillir la cible FPGA. Ainsi la FPGA Xilinx Spartan 3 est integrée sur une carte d'évaluation Digilent Starter Kit.

Fabricant	Famille	Circuit FPGA	Osc. en anneau	Osc. embarqué	Synthèse de fréq.	Osc. externe
Xilinx	Spartan 3	XC3S200	**oui**	non	2 DFS	**oui**
Xilinx	Spartan 3AN*	XC3S700AN	**oui**	non	8 DFS	**oui**
Actel	Fusion*	M7AFS600FG256	**oui**	**oui**	2 DFS 2 PLL	**oui**
Actel	Fusion	M7AFS600FG484	**oui**	**oui**	2 PLL	**oui**
Actel	Igloo	AGL125GNG132QN	**oui**	**oui**	1 PLL	**oui**
Altera	Cyclone III*	EP3C25F256I7	**oui**	non	4 PLL	**oui**

Tableau 1.1 Sources de signaux jittés disponibles dans les plate-formes utilisées dans le cadre de cette thèse.

La cible Actel Fusion, sur une carte d'évaluation System Management. La cible Actel Igloo sur une carte Actel Icicle Evaluation Kit. Ces précisions sont très importantes car comme nous le montrerons dans le Chapitre 2 le jitter est fortement dépendant de l'environnement externe de la cible FPGA et donc de la carte support nottament au regard de l'alimentation utilisée.

3 TRNG dans les cibles ASIC

La génération d'aléa dans des cibles spécialisés (ASIC), où le concepteur dispose de beaucoup plus de libertés, diffère notablement de celle que l'on peut espérer obtenir dans des cibles reconfigurables. En effet des phénomènes physiques de nature aléatoire, comme le bruit électronique intrinsèque à tous les composants électroniques, y sont plus facilement accessibles et exploitables. Bien que les TRNG conçus pour des cibles ASIC sortent du cadre de cette thèse, un bref aperçu de la littérature qui traite de ce sujet sera exposé car historiquement ce sont les premiers TRNG publiés dans la communauté scientifique et de fait les principes d'extraction et de correction seront largement repris dans la suite par les concepteurs de TRNG destinés aux circuits reconfigurables.

3.1 TRNG proposé par Fairfield *et al.*

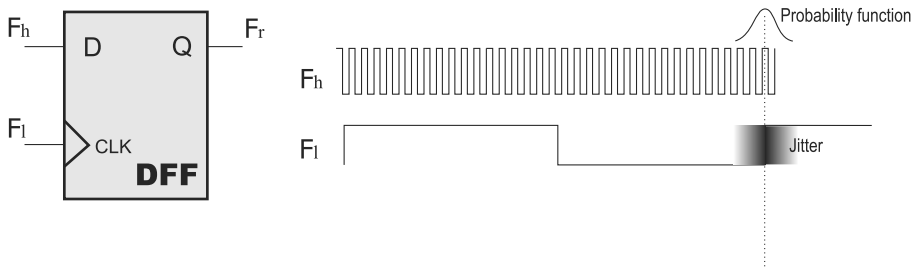

Figure 1.2 Elément d'extraction physique d'aléa basé sur le jitter proposé dans [FMC85].

Historiquement [FMC85] est le premier article dans la communauté scientifique qui traite du sujet d'un TRNG embarqué dans du matériel électronique dédié. Les auteurs proposent d'utiliser le jitter comme source d'aléa en notant que cette source avait été utilisée par la *RAND Corp.* dans les années 40 pour la génération des célèbres tableaux de nombres aléatoires [Cor55]. Pour extraire l'aléa contenu dans le jitter, les auteurs de [FMC85] proposent d'utiliser une simple bascule DFF fonctionnant sur front montant, principe qui sera retenu et très largement repris dans la suite par la communauté scientifique. Il est intéressant de noter que le terme *digital mixing* est utilisé ici pour décrire l'extraction de l'aléa alors que par la suite les auteurs parleront plus généralement d'échantillonnage (*sampling*). Le principe est illustré à la Figure 1.2.

Le signal F_h est non-ajustable et interne au circuit, il est obtenu grâce à un oscillateur embarqué, sa fréquence est de 8Mhz. F_l est par contre réglable grâce à des composants passifs externes, des capacités et des résistances. Les auteurs préconisent l'utilisation d'un oscillateur RC pour le signal F_l au lieu d'un oscillateur LC car ce dernier possède un facteur de qualité Q plus élevé et donc pourrait contenir moins d'aléa.

Une analyse mathématique non négligeable et assez détaillée est présentée ce qui permet de relever deux problèmes. Premièrement, le fait que le signal F_h peut ne pas présenter un rapport cyclique exactement de 50/50, pourrait induire un *biais* (*cf.* Section 8.2.1 de ce chapitre pour une définition du *biais*) dans le flux binaire aléatoire de sortie. Deuxièmement le jitter présent sur le front montant de F_l peut se révéler être insuffisant pour garantir une équiprobabilité des bits à '1' et des '0'. Les auteurs proposent donc une méthode pour calculer les paramètres (jitter et fréquences) nécessaires pour garantir un *biais* qui tends vers 0.

La taille du jitter est quantifiée par son écart-type à la moyenne (déviation standard) ou σ. La probabilité de prédire le bit b(i+1) en connaissant le bit précèdent b(i) est évaluée en fonction du rapport entre la période du signal F_h (notée T_h) et la taille du jitter (σ). Les résultats publiés dans [FMC85] sont reproduits à la Figure 1.3. On voit clairement que pour que la probabilité de prédiction soit proche de 0,5 c'est à dire que la connaissance du bit B(i) n'augmente en rien la probabilité de prédiction aléatoire de 0,5 , la condition suivante doit être satisfaire :

$$\frac{2 \times \sigma_{T_l}}{T_h} > 0,75 \tag{1.1}$$

Il est même fortement conseillé par les auteurs que ce rapport dépasse 1,25. Si cette condition n'est pas respectée, deux conséquences s'en suivent : une probabilité non équilibrée entre les états haut et bas sera présente dans le flux aléatoire, induisant ainsi un *biais*, et une corrélation à court terme entre deux ou plusieurs bits successifs pourra être présente dans le flux de sortie.

Dans la réalité, il est difficile de satisfaire cette condition car la taille du jitter présente dans les circuits est généralement faible ce qui se traduirait par l'obligation

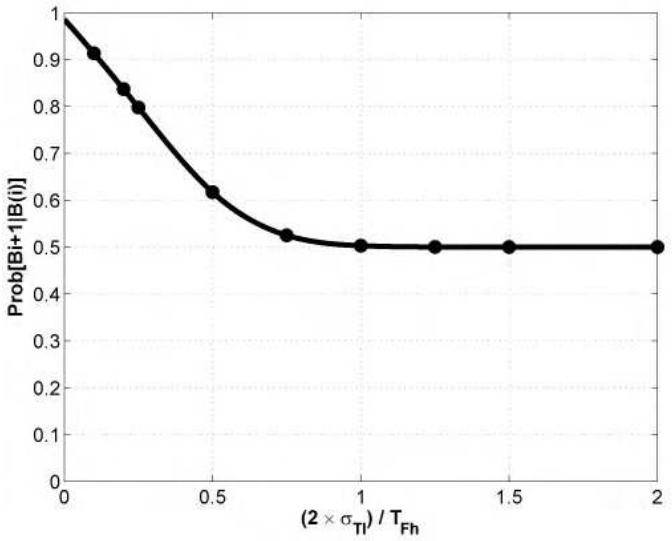

Figure 1.3 Étude du biais en fonction de la taille du jitter et de la fréquence du signal.

d'utiliser une fréquence excessivement élevée pour le signal F_h. Or ici F_h est fixé à 8Mhz soit $T_h = 125$ ns et la déviation standard du signal F_l est expérimentalement mesurée à 2,11 ns à une fréquence de 3,674 kHz.

Pour palier cette limitation, il est proposé d'utiliser un correcteur ou filtre de parité qui va effectivement corriger le *biais*, le prix à payer étant le débit qui s'en trouvera fortement diminué. Le correcteur utilisé par les auteurs de [FMC85] sera également largement repris par la communauté par la suite et son fonctionnement sera décrit plus en détail dans la Section 6.2 de ce chapitre. Pour éviter la corrélation bit à bit, les auteurs proposent également d'utiliser un brouilleur (*scrambler*) assez complexe et donc coûteux en termes de ressources matérielles. Ce brouilleur va par ailleurs corriger le biais enlevant ainsi le besoin du filtre de parité, si on est prêt à en payer le coût matériel.

Un autre aspect très intéressant, généralement ignoré, est étudié dans [FMC85]. En effet selon le rapport qui existe entre les fréquences des signaux F_h et F_l, on aura à la sortie de la bascule DFF un résultat qui présentera un certain motif, dans lequel certains bits seront aléatoires et d'autres pas. Les auteurs étudient donc les séquences pseudo-aléatoires obtenues en fonction de la fréquence du signal F_l et concluent qu'une fréquence basse pour ce signal est plus avantageuse mais alors le débit s'en trouve diminué.

Les auteurs proposent ensuite une logique de contrôle du TRNG assez complexe et robuste qui permet notamment de détecter des dysfonctionnements des oscillateurs.

Un des aspects non abordés dans [FMC85] est le fait que le signal F_h présentera

inévitablement du jitter qui va s'accumuler entre deux fronts montants successifs de F_l (ce phénomène d'accumulation sera étudié en détail dans la suite de cette thèse). Ainsi la source d'aléa n'est pas clairement identifiée ni clairement étudiée. Un modèle Gaussien est supposé en ce qui concerne l'instabilité du signal jitté F_l, or cette hypothèse est faiblement argumentée et étudiée.

Dans cet article, les auteurs sont conscients que le signal aléatoire est de mauvaise qualité et sont donc obligés d'utiliser un post-traitement pour le corriger. Cette approche est particulièrement bien défendue dans cet article et étudiée mathématiquement. Le fait que le signal instable basse fréquence soit obtenu grâce à des composants passifs externes au circuit est à notre avis une faiblesse dans la sécurité de ce TRNG.

3.2 *The Intel RNG* [JK99]

Le générateur présenté dans [JK99] a été développé par le leader mondial des semi-conducteurs *Intel* et a été utilisé dans un *chipset* pour fournir aux systèmes d'exploitation une source de vrai aléa conformément aux recommandations données dans [CS94] RFC1750. En effet l'absence de source d'aléa physique dans les systèmes d'exploitation est un problème et les méthodes logicielles donnent parfois de très mauvais résultats ([GW96], [Zal01]). La description donnée dans [JK99] est une évaluation de ce générateur par la société leader dans le domaine de la cryptographie *Cryptography Research*. La source d'aléa est ici le bruit thermique, encore appelé bruit Johnson, aux bornes d'une résistance intégrée. Pour s'affranchir de sources de bruit externes, présentant souvent des corrélations, les concepteurs de ce TRNG ont réalisé une source différentielle en soustrayant le bruit obtenu aux bornes de deux résistances intégrées identiques. De cette manière, les perturbations induites par le bruit d'alimentation ou les sources électromagnétiques environnantes ou encore la dépendance à la température sont fortement réduites. Le bruit différentiel ainsi obtenu est appliqué à un oscillateur commandé en tension ou VCO (*Voltage Controlled Oscillator*) de basse fréquence qui vient échantillonner un oscillateur haute fréquence, le rapport des fréquences étant de 1 :100. C'est le schéma traditionnel du TRNG comme dans [FMC85]. Les concepteurs ont prévu un correcteur de type Von Neumann pour corriger les imperfections à la sortie de l'échantillonneur (mixeur). Le correcteur de Von Neumann (*cf.* Section 6.1 de ce chapitre) est efficace mais son principal défaut est de ne pas garantir un débit constant. D'après les auteurs de [JK99], pour ce générateur, il produit en moyenne 1 bit de sortie pour 6 bits d'entrée, témoignant du fort *biais* existant dans le flux binaire aléatoire avant le correcteur ou post-traitement.

Dans [JK99], les auteurs évaluent la qualité des bits aléatoires obtenus, avant et après le correcteur, ce qui est indéniablement une bonne approche. Plusieurs suites de tests statistiques sont utilisés : NIST, FIPS140-1, DIEHARD sur des suites aléatoires d'au moins 80Mbits. Les auteurs ne présentent malheureusement pas les résultats, mais laissent entendre que les tests statistiques passent après le correcteur. Le débit obtenu est de 75Kbits par seconde.

Un autre TRNG très similaire est présenté dans [ZZW08].

3.3 Autres TRNG dans les cibles ASIC

Dans ce paragraphe sont présentés plusieurs principes de TRNG destinés aux cibles ASIC qui bien que non implantables dans des cibles FPGA nous semblent intéressants.

Dans [BB99] les auteurs présentent un principe de génération qui est basé sur l'amplification et la quantification de bruit blanc filtré. Ce générateur est donc en très large partie analogique et une implantation dans un FPGA semble impossible. Le bruit analogique est d'abord amplifié et ensuite transformé en une suite numérique grâce à un comparateur à hystérésis. Plusieurs dispositions sont prises en considération pour garantir un fonctionnement normal même si des variations de conditions externes et de composants surviennent. Les auteurs sont conscients qu'un post-traitement déterministe et complexe, bien que capable d'éliminer le *biais* et les éventuelles corrélations bit-à-bit peut masquer l'aléa. Le plus important dans cet article est que ce générateur est décrit par un modèle mathématique qui lie la corrélation des bits aléatoires de la sortie du TRNG aux paramètres internes du générateur, exprimant de cette façon le débit limite que l'on peut espérer obtenir. Les auteurs de [BB99] ne donnent aucun résultat d'implantation.

Un autre générateur, proposé par Killmann *et al.* dans [KS08], est particulièrement intéressant car les auteurs proposent un estimateur robuste de l'entropie minimale que peut fournir le TRNG. Dans cet article, les auteurs insistent sur le fait que l'entropie est une propriété des variables aléatoires et non de leur réalisation (bits observés), ce qui en fait une grandeur très difficilement mesurable physiquement. D'où l'importance d'un modèle mathématique pour pouvoir travailler avec les variables aléatoires. La source d'aléa est ici une paire de diodes *Zeners* intégrées qui peut fournir un bruit de presque 1 mV d'amplitude et d'une fréquence maximale de 10Mhz. Comme dans [JK99], les deux signaux issus des deux diodes sont fournis à un amplificateur de différence pour s'affranchir autant que possible des sources de bruit environnantes. La sortie de l'amplificateur est ensuite transmise à un *trigger de Schmitt* qui mesure les temps des transitions 0-1 uniquement du signal (comparés à un seuil égal à la valeur moyenne du signal à la sortie de l'amplificateur). En sortie est présente une suite de niveaux haut et bas dont la longueur dans le temps est aléatoire. L'extraction se fait grâce à un compteur modulo 2. Le principe est décrit à la Figure 1.4. Le débit obtenu est de 500 Kbs.

Bucci *et al.* proposent plusieurs principes de TRNG [BGL+03], [BGL+06], [BL05], [BL07], [BBL04]. Malheureusement aucune implantation FPGA n'est proposée et certains principes semblent peu adaptés aux cibles FPGA. Néanmoins les principes sont accompagnés souvent par une modélisation mathématique d'un niveau cependant inférieur à [KS08].

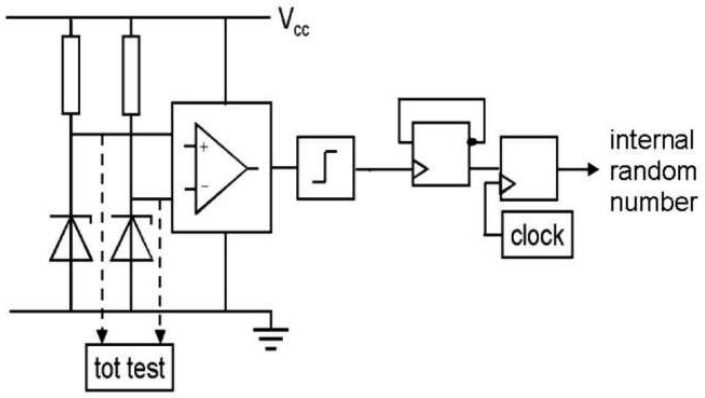

Figure 1.4 TRNG proposé par Killman et Schindler basé sur la détection de jitter différentielle dans deux diodes Zenner.

Dans [HOB+06], les auteurs proposent deux descriptions de TRNG destinés à des cibles ASIC, à faible consommation d'énergie et avec une sensibilité faible aux perturbations déterministes environnantes. Il est à noter que la consommation est un paramètre très important, souvent critique lorsque les TRNG sont embarqués dans des cartes à puces car le budget énergétique disponible peut alors être très faible. Dans ce contexte, les auteurs annoncent une consommation de 292 μW dans une technologie standard de 0.35 μm. Les deux principes proposés dans [HOB+06] sont basés sur la métastabilité d'une bascule *latch* dont la description électronique est fournie. Les deux TRNG utilisent des techniques actives de compensation afin de garantir la fonctionnalité même si des perturbations sont présentes. Les auteurs utilisent deux registres à décalage non linéaires (NFSR) modifiés, dans une configuration dite en *alternating step generator* comme dans [TLL07] pour corriger le *biais* résiduel. Les débits obtenus sont de 500 Kbs, 50 Kbs et 5 Kbs.

Petrie *et al.* [PC00] proposent un générateur basé sur l'amplification directe de bruit. Ce bruit est ensuite transmis à un convertisseur analogique/numérique. Ils obtiennent un débit de 1.4 Mbs. Bien que n'utilisant pas une source de bruit différentielle, ils prétendent obtenir une suite aléatoire de bonne qualité même avec des perturbations sur l'alimentation obtenant ainsi un TRNG robuste.

Liu *et al.* dans [LM05] proposent un TRNG qui fournit une suite aléatoire de bonne qualité avec un débit de 100 Kbs. La source d'aléa est le jitter présent dans une PLL. Ils utilisent également un correcteur Von Neumann ainsi que plusieurs diviseurs de fréquences. Le principe est réalisé dans un ASIC en technologie CMOS 1.5 μm.

Un autre TRNG est proposé dans [KC02]. Son principe est basé sur un circuit bistable volontairement placé dans un état métastable pouvant ainsi fournir une suite

aléatoire à 100 Mbs. Cependant ce principe utilise également un correcteur XOR.

Les auteurs de [PPL09] proposent une méthode très originale et peu coûteuse qui permet de disposer de nombres aléatoires dans un système informatique de haut niveau de type PC. Le principe est basé sur la collision entre l'écriture dans la DRAM et le rafraîchissement de celle ci. Ce générateur bien qu'exploitant une source physique est entièrement réalisé en *software* dans le PC. Les suites ainsi obtenues passent les tests NIST SP800-22 et sont donc de bonne qualité.

Citons également [SS04] dans lequel les auteurs présentent un TRNG implanté dans une cible ASIC 0.18 μm fonctionnant à une vitesse de 1Ghz et basé sur un réseau d'oscillateurs en anneau contrôlés.

4 TRNG dans les cibles FPGA

Nous présenterons dans cette section une synthèse des principaux TRNG présentés par la communauté scientifique qui sont directement implantables dans les cibles FPGA. Les différents principes de génération sont décrits de façon chronologique. Cet ordre a été établi par année de publication, cependant il faut noter que certains principes ont souvent été présentés à la communauté scientifique lors de conférences et sont donc antérieurs à la date de publication finale, souvent dans une revue à comité de lecture. C'est le cas notamment pour le générateur proposé par Sunnar *et al.* publié dans la revue *IEEE Transactions on Computers* mais connu de la communauté dès 2005.
Une classification de ces TRNG sera discutée dans la Section 5 de ce chapitre.

4.1 TRNG proposé par Tsoi *et al.*

Deux versions de générateurs sont proposées par Tsoi *et al.* [TLL03] [TLL07]. Les principes sont semblables mais la première version [TLL03] exige pour fonctionner des composants externes au FPGA pour réaliser la source d'aléa. Dans cet article, une confusion peut survenir du fait que les auteurs utilisent le sigle PRNG pour désigner deux choses différentes : *Pseudo Random Number Generator* et *Physical Random Number Generator*. Par la suite comme partout dans cette thèse, sous le sigle PRNG nous entendrons *Pseudo Random Number Generator*.

4.1.1 TLL03

Historiquement avec le TRNG proposé par Fischer *et al.* [FD03], c'est un des premiers générateurs complètement intégrable dans une cible FPGA. Il reprend également le principe proposé dans [FMC85] où un signal basse fréquence instable F_l (*i.e.*

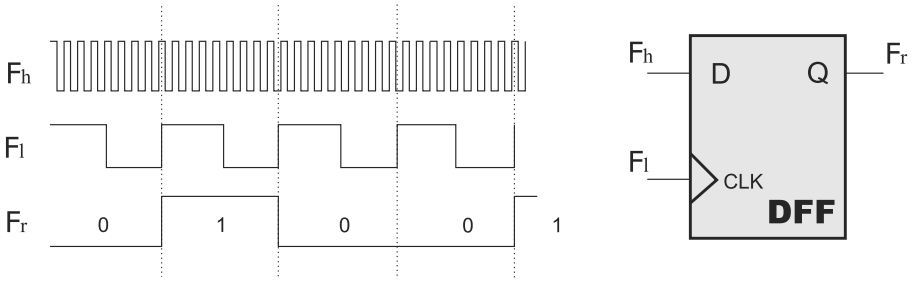

Figure 1.5 Échantillonneur proposé dans [TLL03].

avec un jitter important) échantillonne un signal haute fréquence plus stable F_h. Le principe de cet échantillonnage est présenté à la Figure 1.5. L'élément d'échantillonnage étant comme dans beaucoup de générateurs d'aléa une simple bascule DFF. Dans cet article, les auteurs proposent un TRNG et un PRNG qui peuvent toutefois fonctionner séparément. L'algorithme pour réaliser le PRNG est BBS *Blum Blum Shub* et ne sera pas étudié ici car son fonctionnement est indépendant. Un débit important n'est pas recherché par les auteurs.

Un point caractéristique de ce principe est que la réalisation du signal contenant le jitter et donc qui est utilisé comme source d'aléa est externe au FPGA, ce qui d'un point de vue de la sécurité est peu recommandable. En effet une attaque simple contre ce générateur serait d'injecter un signal que l'attaquant peut contrôler et moduler compromettant ainsi sérieusement la sécurité de tout le système cryptographique. Le circuit utilisé pour produire le signal F_l est présenté à la Figure 1.6. Les *buffers* et les inverseurs sont réalisés en interne dans le FPGA en utilisant les blocs d'entrée-sortie (IOB) du FPGA. Les composants externes sont la capacité C et les deux résistances, dont une ajustable, pour modifier la fréquence du signal F_l. Les auteurs constatent que plus la fréquence du signal est basse plus le jitter est important sans toutefois en donner une explication. Bien que les auteurs annoncent ne pas avoir besoin d'utiliser un brouilleur (*scrambler*) à la sortie de leur générateur pour passer les tests statistiques ils utilisent tout de même comme dans [FMC85] un filtre de parité d'ordre 4 qui élimine le *biais* mais qui inévitablement induit une diminution du débit par 4. Le TRNG complet avec le filtre de parité est présenté à la Figure 1.7.

Les auteurs n'abordent pas dans cet article ni dans [TLL07] la question de la qualité et les caractéristiques de la source d'aléa elle même et des éventuelles corrélations qu'elle peut comporter.

Pour valider la sortie du TRNG, ils utilisent deux suites de tests statistiques : NIST SP800-22 et DIEHARD, pour différentes fréquences de l'oscillateur basse fréquence. Les fréquences 265 kHz, 151 kHz, 115 kHz, 52 kHz, 36.8 kHz et 18.8 kHz sont étudiées. Ils constatent que les tests NIST passent à partir de 115 kHz dans le sens décroissant. Pour cette même fréquence de fonctionnement, les tests DIEHARD passent

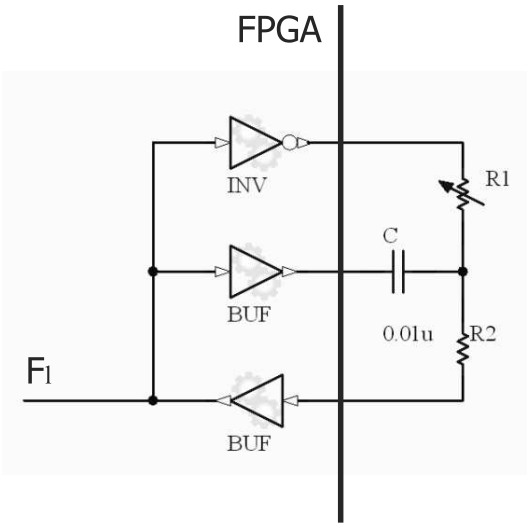

Figure 1.6 Circuit oscillant comportant des éléments externes pour obtenir la source d'aléa dans [TLL03].

aussi (ce qui n'est pas le cas pour les fréquences de fonctionnement 52 et 36.8). Le débit final est donc de 115/4=29 Kbs.

Les auteurs utilisent une mémoire comme buffer de sortie ainsi que comme interface au PRNG. Un compteur est utilisé pour adresser ce bloc mémoire et qui n'apparaît pas sur la Figure 1.7. La taille de cette mémoire n'est pas discutée dans le texte. Le signal haute fréquence F_h est obtenu grâce à un conditionneur de fréquence (DFS : *Digital Frequency Synthesis*) interne du FPGA basé sur des DLL (*Delay Lock Loop*). La fréquence ainsi obtenu est de 266 Mhz. En conclusion ce générateur présente l'avantage

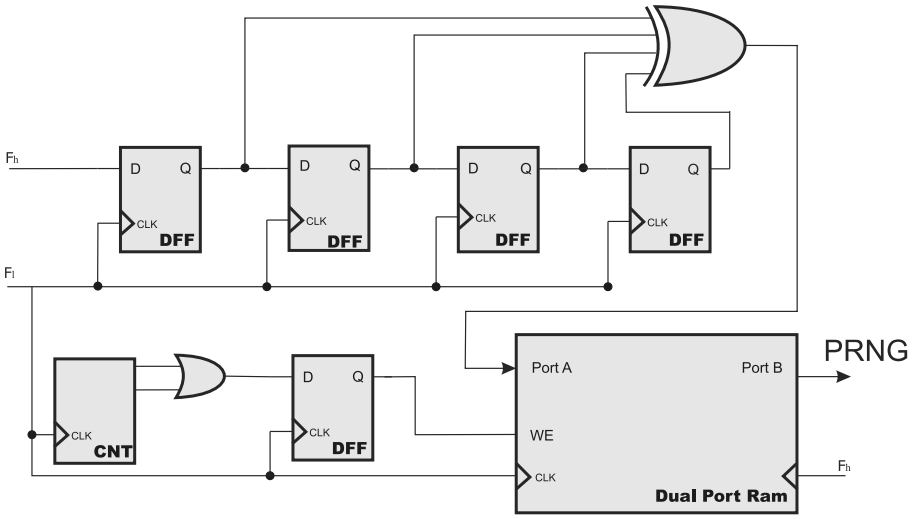

Figure 1.7 Principe de TRNG proposé dans [TLL03] où apparaissent l'échantillonnage et le filtre de parité d'ordre 4.

d'être simple et de reposer sur un principe relativement bien connu, bien que pas très détaillé dans cet article. Le rapport entre F_h et F_l étant important, la suite aléatoire obtenue est de bonne qualité. Les auteurs n'évaluent pas cette dernière et se contentent de lui appliquer un correcteur pour uniformiser la probabilité d'avoir des 1 ou des 0 en sortie (*i.e.* le *biais*). Le correcteur étant relativement simple c'est plutôt une bonne approche. Le principal défaut est que le signal qui sert de source d'aléa est réalisé grâce à des composants externes au FPGA et donc son utilisation dans un système cryptographique n'est pas recommandée.

4.1.2 TLL07

Le principe décrit dans [TLL07] reprend le coeur du TRNG décrit dans [TLL03] qui est l'échantillonnage par une bascule DFF d'un signal haute fréquence par un signal basse fréquence. Dans cette nouvelle version, le problème majeur de [TLL03] est corrigé, c'est à dire que la nouvelle structure ne comporte plus d'éléments externes au FPGA. Le TRNG est donc beaucoup plus adapté à un système cryptographique. Le signal F_h est maintenant généré par un oscillateur en anneau comportant 3 éléments (le fonctionnement détaillé des oscillateurs en anneau sera donné dans le Chapitre 2), judicieusement réalisé pour occuper une seule *slice* dans le circuit FPGA et maximiser ainsi la fréquence obtenue. L'implantation de cet oscillateur est présentée sur la Figure 1.8. En ce qui concerne le signal d'échantillonnage basse fréquence F_l, il est obtenu par

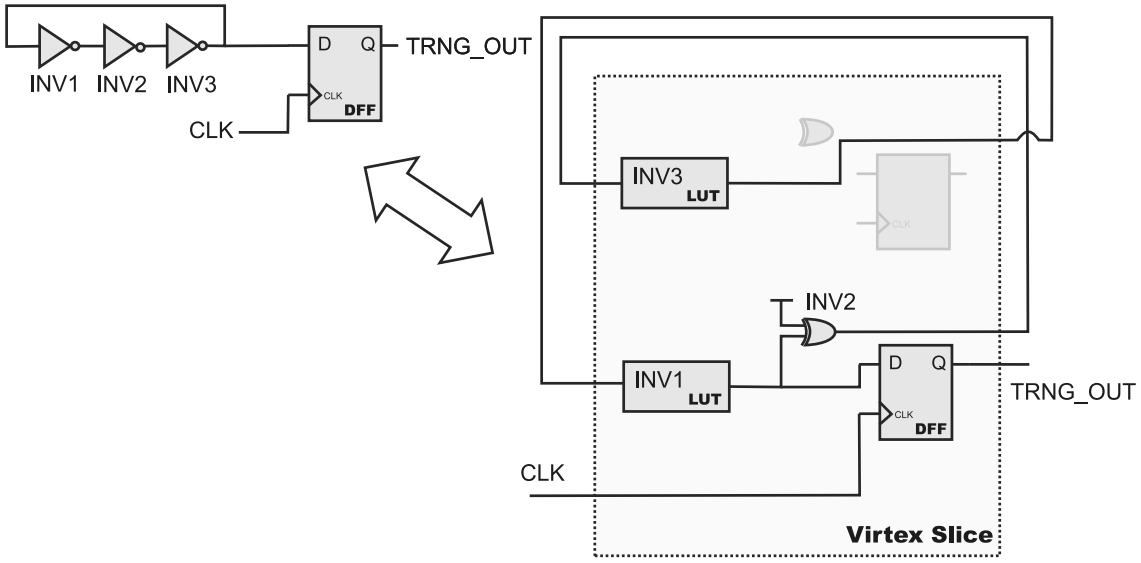

Figure 1.8 Structure logique interne de l'oscillateur en anneau dans [TLL07].

division de fréquence du signal d'horloge système qui est de 133Mhz dans les cartes utilisées par les auteurs. Il est à noter que ce signal est externe mais puisqu'il est utilisé par ailleurs comme signal d'horloge commune à tout le circuit FPGA, on peut avoir une certaine confiance dans son intégrité.

Les auteurs introduisent un concept intéressant pour augmenter la fréquence du signal issu de l'oscillateur en anneau F_h présenté à la Figure 1.9. Ce dispositif permet de doubler la fréquence de l'oscillateur et ainsi avoir un rapport plus important entre F_h et F_l. Malheureusement son effet n'est pas aussi important que l'on pourrait le vouloir. Le TRNG complet est présenté à la Figure 1.10 (sans le doubleur de fréquence). On

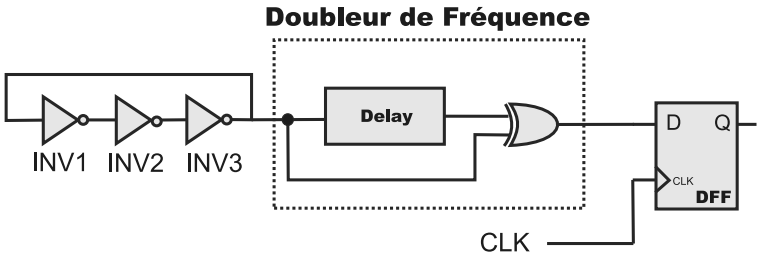

Figure 1.9 Doubleur de fréquence de l'oscillateur en anneau.

peut constater que contrairement à [TLL03] ce générateur est un TRNG hybride, c'est à dire qu'il combine un TRNG et un PRNG qui sont cascadés. La sortie est évaluée après le PRNG et non à la sortie du TRNG. Les auteurs ont d'ailleurs essayé d'appliquer un seul des tests de la suite FIPS-140-2 (le *poker test*) directement à la sortie du TRNG, sans succès. Bien que le doubleur de fréquence améliore les résultats de ce test, il ne permet pas de le passer.

Les auteurs évaluent donc les suites obtenues à la sortie du PRNG (deux LFSR formant un circuit appelé *Alternating Step Generator*) avec les suites de tests statistiques NIST SP800-22, DIEHARD et TestU01. Tous les tests passent avec succès avec un débit de 133 Mbs. Les auteurs n'abordent pas dans cet article non plus la qualité de la source d'aléa, et ne précisent pas où elle se situe. Est-ce le signal F_h ou bien F_l qui contient la source d'aléa ?

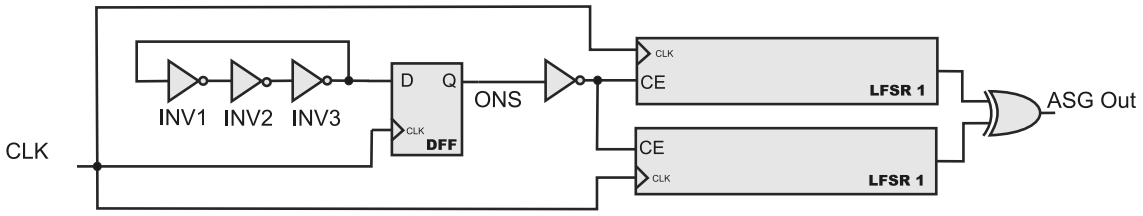

Figure 1.10 TRNG (hybride) complet décrit dans [TLL07].

4.2 TRNG proposé par Fischer et Drutarovsky

Historiquement, le générateur proposé par Fischer et Drutarovsky [FD03], [FDS+04] est l'un des premiers TRNG conçus spécialement pour des applications cryptographiques embarqués dans des cibles FPGA. Son principe de fonctionnement est présenté à la Figure 1.11. Il utilise comme source d'aléa le jitter présent dans un signal d'horloge

généré par une boucle à verrouillage de phase ou PLL (*Phase Locked Loop*). L'extraction se faisant par une bascule DFF classique. La PLL garantit un rapport rationnel entre les deux signaux T_{CLK} et T_{CLJ} ce qui permet d'arriver à un échantillonnage virtuel assez fin de T_{CLJ} et donc de détecter assez précisément le jitter. Ce générateur fournit un flux aléatoire d'assez bonne qualité et n'a pas besoin de post-traitement ou de correcteurs pour compenser le biais statistique des probabilités des '1' et des '0'.
Le coût matériel est faible, il est néanmoins obligatoire de disposer d'une PLL dans le circuit cible choisi. Cette limitation a été largement remarquée par la communauté lors de sa publication. De nos jours les trois principaux fabricants de FPGA proposent des circuits avec des PLL, avec plusieurs sorties différentes ce qui diminue d'autant le coût matériel pour ce TRNG.

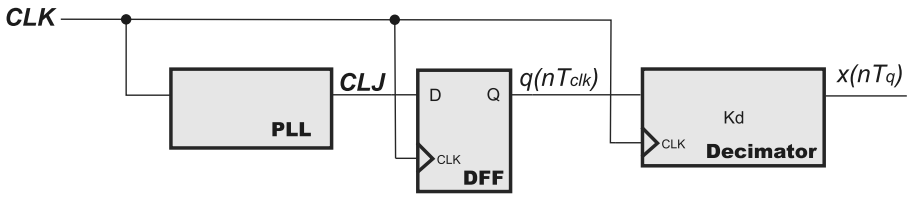

Figure 1.11 TRNG proposé par Fischer et Drutarovsky basé sur la détection du jitter d'un signal généré par une PLL.

4.3 TRNG proposé par Tkacik

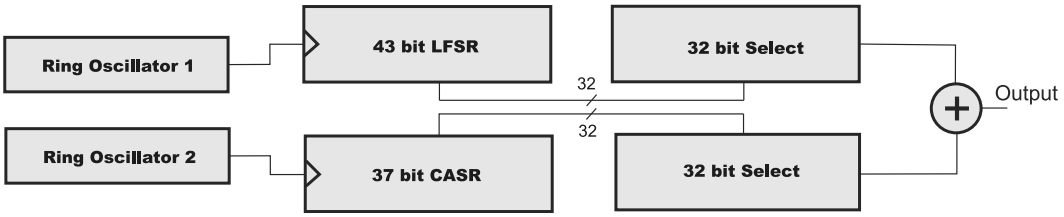

Figure 1.12 TRNG hybride proposé par Tkacik.

Tkacik dans [Tka03] présente un TRNG facilement implantable dans les circuits logiques FPGA, bien qu'originellement implanté dans un ASIC. La source d'aléa est ici également le jitter, bien que de manière non traditionnelle. Il faut noter ici que c'est un générateur hybride, dans la mesure où il utilise deux machines d'état complètement déterministes qui sont cadencés par deux signaux jittés issus de deux oscillateurs en anneau. Le principe de fonctionnement est présenté à la Figure 1.12.

Deux PRNG sont utilisés, un LFSR d'une longueur de 43 bits et un automate cellulaire CASR d'une taille de 37 bits. Chacun de ces PRNG est cadencé par un oscillateur en anneau différent dont la fréquence est sujette à des fluctuations (jitter).

Trente deux bits sont sélectionnés dans chacun des PRNG et sont ensuite ajoutés ensemble modulo 2 (XOR). L'auteur ne spécifie pas comment sont choisis ces 32 bits. Le TRNG fournit ainsi un mot de 32 bits sur demande.

Une particularité de ce générateur est qu'il n'est pas conçu pour fonctionner à une cadence fixe mais uniquement lorsque un mot est demandé. L'auteur nous indique par ailleurs qu'entre deux mots successifs, les deux PRNG doivent tourner au moins deux fois leurs longueurs respectives. C'est à dire que le LFSR doit fonctionner au moins pendant 86 périodes d'horloge avant de pouvoir fournir un mot de 32 bits.

Les auteurs évaluent la qualité statistique de ce générateur grâce à la suite DIEHARD, en affirmant que les tests passent sans toutefois donner des détails sur le débit obtenu. Au contraire ils parlent d'échantillonnage variable.

L'absence de toute modélisation ainsi que l'absence de caractérisation de la source d'aléa de ce générateur en font un générateur faible. Dichtl dans [Dic03] montre comment on pourrait hypothétiquement attaquer ce générateur et prévoir sa sortie. L'attaque est théorique mais illustre bien les problèmes liés à ce TRNG.

4.4 TRNG proposé par Kohlebrenner et Gaj

Kohlebrenner et Gaj dans [KG04] présentent un TRNG utilisant le jitter dans les oscillateurs en anneau comme source d'aléa d'une manière nouvelle. Au lieu d'utiliser un oscillateur rapide échantillonné par un oscillateur lent, Kohlebrenner et Gaj proposent d'utiliser deux oscillateurs dont la période est très proche. De cette manière, à la sortie de la bascule D on retrouvera une suite de bits à l'état haut suivie d'une suite de bits à l'état bas, c'est le signal S_0. Le principe du TRNG est présenté à la Figure 1.13 : le bloc d'échantillonnage qui est le coeur du générateur à la Figure 1.13 et le chronogramme de fonctionnement est donné à la Figure 1.14. Le signal S_0 présente une fréquence beaucoup plus faible que les signaux Clk_0 et Clk_1. Elle est directement proportionnelle à la différence des périodes des signaux Clk_0 et Clk_1 : plus cette différence est faible plus la fréquence du signal S_0 est faible. Le jitter présent dans les deux signaux aura pour effet de modifier la période du signal S_0 et donc la longueur de la suite de bits à '1' et à '0'. C'est cette longueur variable qui donnera un bit aléatoire à chaque transition '0'-'1' du signal S_0. En effet la longueur de la période de S_0 est compté modulo 2 grâce au signal C_0. Cette longueur sera toujours un multiple entier de période du signal Clk_1 à condition que la différence entre Clk_0 et Clk_1 ne soit pas inférieure à une certaine valeur critique. Si la différence devient trop petite alors le signal S_0 peut ne plus être qu'une succession de bit à '1' et à '0' de longueur variable mais présenter des transitions transitoires courtes fortement corrélés. C'est pour éviter ces transitions de longueur courte que les auteurs ont prévu une logique de contrôle (le bloc contrôleur dans la Figure 1.13).

Le principe de fonctionnement repose entièrement sur la différence de périodes des deux signaux Clk_0 et Clk_1. Dans [KG04] ces signaux sont obtenus à partir d'os-

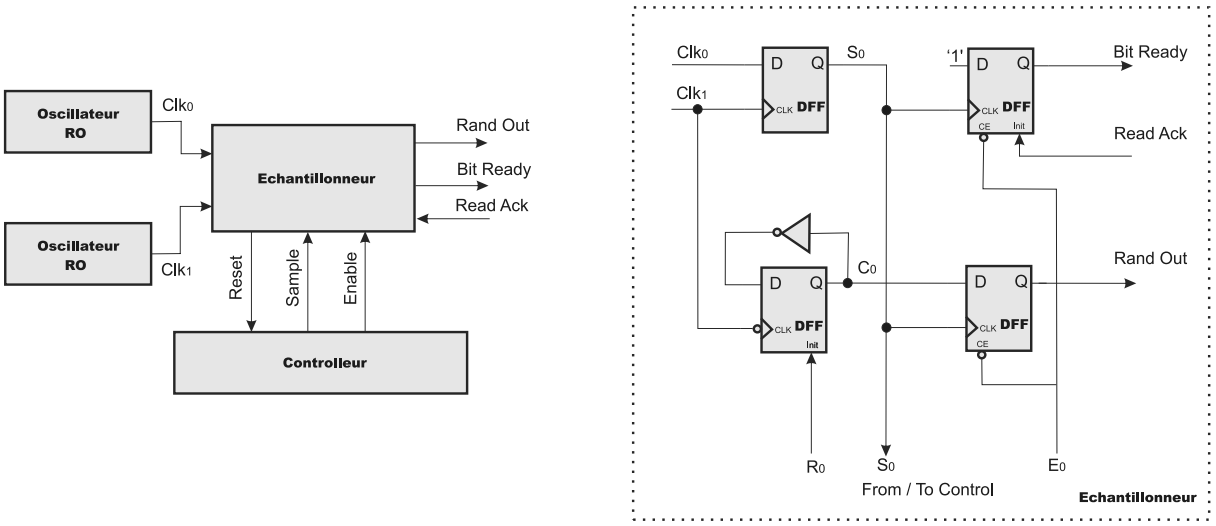

Figure 1.13 TRNG proposé par Kohlebrenner et Gaj utilisant le jitter de deux oscillateurs en anneau de fréquence très proche comme source d'aléa.

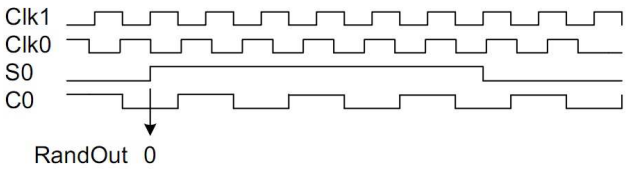

Figure 1.14 Chronogramme de fonctionnement du générateur proposé par Kohlebrenner et Gaj.

cillateurs en anneau. Pour garantir le fonctionnement de ce TRNG, une implantation précise des oscillateurs doit être réalisée. Les auteurs proposent de réaliser des oscillateurs qui tiennent dans un seul *CLB* (*Configurable Logic Block*) dans le FPGA. Ils remarquent qu'au sein d'une même puce (Virtex XCV1000) des variations de fréquence des oscillateurs importantes, de l'ordre de 7%, peuvent survenir à cause des variations des temps de propagation des éléments de l'oscillateur. Ces variations sont dues en partie à la dispersion des paramètres de la technologie lors de la fabrication, mais aussi à cause des variations locales de la température [KG04]. Puisque le débit et la qualité de la suite aléatoire obtenue dépendent de cette différence, une intervention manuelle est donc nécessaire pour trouver un placement fonctionnel des oscillateurs. C'est le principal défaut de ce générateur.

Les auteurs donnent des résultats d'implantation de ce générateur encourageant, néanmoins le signal présente un *biais* statistique, qui est corrigé par un correcteur qui effectue l'opération XOR sur deux bits successifs diminuant ainsi le débit par 2. C'est un filtre de parité d'ordre 2. Il est à noter ici que ce post-traitement est effectué en logiciel avant la validation par la suite de tests statistiques NIST. Un débit de 295 Kbs est obtenu qui passe tous les tests de la suite NIST avec 1 Gbits aléatoires testés.

Nous allons étudier ce générateur plus en détail dans le Chapitre 3 et présenter une série d'améliorations.

4.5 TRNG proposé par Sunar *et al.*

Sunar *et al.* dans [SMS07] proposent un TRNG destiné aux cibles FPGA, entièrement numérique, basé sur un grand nombre d'oscillateurs en anneau et qui présente un débit relativement important. Le principe est présenté à la Figure 1.15. Le principe est similaire à l'extraction historique de [FMC85] mais une idée nouvelle importante est introduite. Au lieu d'échantillonner un seul oscillateur en anneau, Sunar *et al.* proposent d'échantillonner un signal qui est le XOR d'un grand nombre d'oscillateurs qui contiennent du jitter. En faisant l'hypothèse que le jitter dans les oscillateurs est Gaussien et suit donc une loi Normale $N(\mu, \sigma^2)$, Sunar et *et al.* proposent un modèle statistique pour calculer le nombre minimal d'oscillateurs nécessaires pour garantir que l'échantillonnage se fera dans une zone dominée par le jitter. Le nombre ainsi obtenu, pour un σ donné, est de 114 oscillateurs de 13 éléments chacun. Ensuite, disposant de ce modèle les auteurs proposent un correcteur qui est une fonction cryptographique (*resilient function cf.* Section 6.5 de ce chapitre) basé sur un code BCH (256,16,113). Le débit obtenu par les auteurs est de 2.5 Mbs. Les oscillateurs sont implantés dans une cible FPGA Xilinx. Avec 13 éléments, leur période est mesurée à 25 ns et leur jitter RMS à 0.5 ns.

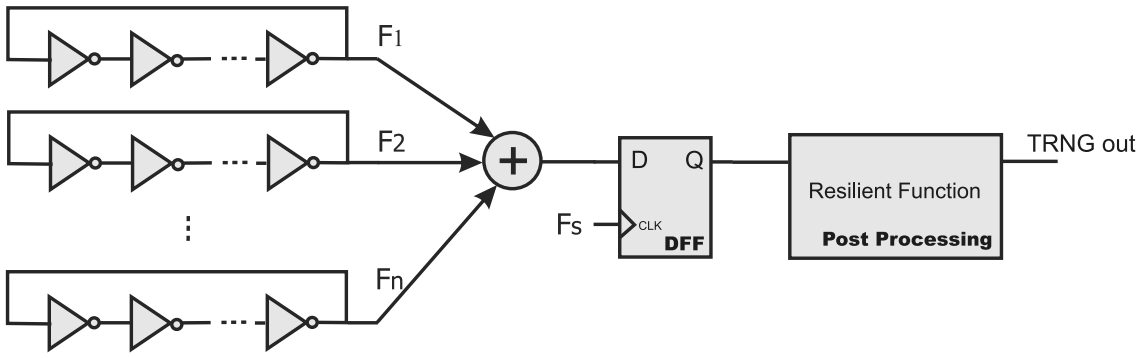

Figure 1.15 TRNG proposé par Sunar *et al.* basé sur l'échantillonnage d'un grand nombre de signaux avec du jitter.

Ce principe a été largement (peut être trop) critiqué dans la communauté scientifique en particulier car les auteurs avaient annoncé que leur TRNG était *provably secure* alors que leur preuve reposait en fait sur des suppositions (hypothèses) faibles. Dichtl avance le premier des critiques sur ce principe [DG07]. Nous relevons également plusieurs objections sur la preuve de sécurité proposée.

Tout d'abord le modèle statistique proposé sous-entend l'indépendance des oscillateurs, fait non prouvé dans l'article, largement contesté par la communauté. De plus

le jitter est supposé suivre une loi Normale idéale, nous montrerons dans le cadre de cette thèse que cela est en partie faux. Enfin, la mesure du jitter donnée par les auteurs, au regard de nos propres résultats, nous semble fortement exagérée. Nous montrerons dans le Chapitre 2 que la méthode de mesure, si elle est effectuée en externe, a une importance capitale sur la taille du jitter mesuré.

Le principal défaut soulevé par Dichtl est le fait qu'une porte XOR peut difficilement fonctionner avec 114 signaux de 40Mhz. C'est pourquoi le signal qui a été modélisé n'est plus le signal réel que l'on peut obtenir dans un vrai FPGA. Même si le signal à l'issue du XOR était correctement obtenu il serait difficilement échantillonable par la bascule car avec un nombre important de transitions il violerait trop souvent les temps de *setup* et *hold* de la bascule. Également la consommation de ce générateur, qui n'est pas évaluée dans cet article, est trop importante avec les 114 oscillateurs qui fonctionnent en même temps à 40Mhz.

Ce générateur est de loin le plus commenté par la communauté. D'un coté l'implantation est aisée et entièrement numérique, généralisable à une large gamme de cibles FPGA avec un débit important. D'un autre coté, la consommation importante, le modèle non réaliste et l'aléa masqué par une fonction de brouillage forte en font un mauvais candidat pour un TRNG dont l'aléa serait maîtrisé.

Il est intéressant de noter que dans [CS94] ce principe de génération avait déja été discuté.

4.5.1 Améliorations proposées par Schellekens *et al.*

Schellekens *et al.* dans [SPV06] proposent une étude et une nouvelle implantation du générateur proposé par Sunar *et al.* Ils proposent une implantation différente avec seulement 110 oscillateurs à 3 éléments ainsi qu'une version plus robuste avec 210 oscillateurs à 3 éléments.

4.5.2 Améliorations proposées par Yoo *et al.*

Yoo *et al.* dans [YSKB07] présentent une étude intéressante sur l'implantation et le routage des oscillateurs en anneau. Une étude intéressante sur l'influence de la température et la tension d'alimentation est proposée. En effet les auteurs remarquent que la fréquence des oscillateurs est fortement dépendante de ces paramètres. Afin d'améliorer la robustesse face à des attaques non intrusives les auteurs proposent une architecture avec deux différents types d'oscillateurs en anneau. Ils atteignent un débit de 67 Mbs pour une consommation de 0,3W ce qui est considérable.

4.5.3 Améliorations proposées par Wold et Tan

Wold et Tan dans [WT08] proposent une amélioration notable au principe original. Afin de palier au problème soulevé par Dichtl en ce qui concerne le signal à la sortie

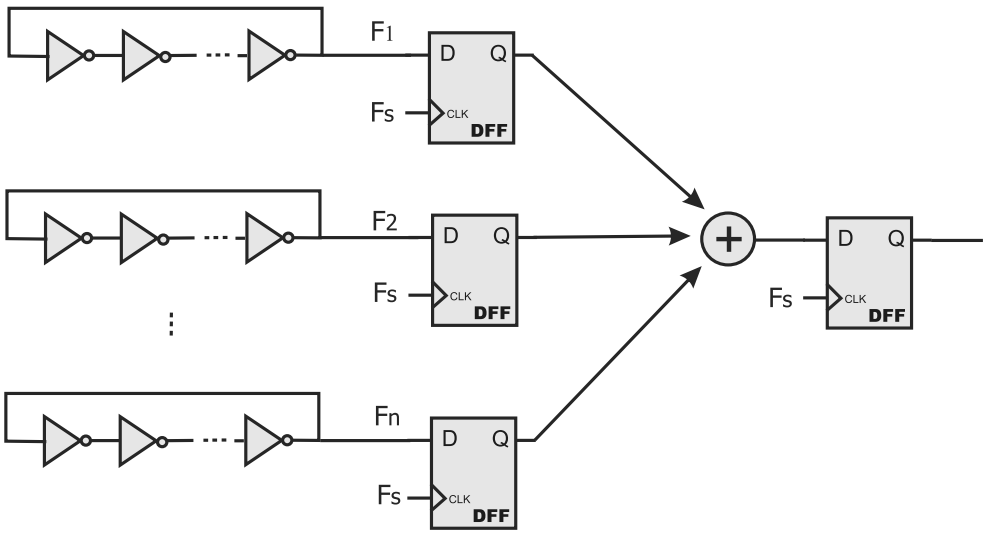

Figure 1.16 Amélioration du TRNG de Sunar *et al.* proposée par Wold *et al.*

du XOR, Wold et Tan ont eu l'idée de synchroniser le signal avant le XOR à l'aide de bascules qui sont synchrones avec le signal d'échantillonnage. Le schéma proposé est décrit à la Figure 1.16. Ceci résout effectivement le problème. Wold et Tan proposent en plus de réduire le nombre des oscillateurs à 50 (à 3 éléments chacun). Leur implantation permet de passer les tests statistiques NIST SP800-22 et DIEHARD avec succès tout en obtenant un débit de 100Mbs.

Nos travaux présentés dans [BBFV10] et [BBF09] remarquent que sur un plan théorique et idéal les deux principes, l'original proposé par Sunar *et al.* et l'amélioration de Wold et Tan, sont identiques. Par contre dans le matériel, les deux principes se comportent différemment montrant par ce fait que le principe proposé par Sunar a bien un problème avec le signal à la sortie du XOR. L'amélioration de Wold est donc effective.

Cependant, un autre problème très intéressant lié au générateur de Sunar est soulevé dans [BBFV10], [BBF09]. Nous montrons en effet, par la simulation, que même si aucun jitter n'était présent dans les signaux des oscillateurs, à partir d'un nombre déterminé d'oscillateurs les tests statistiques passent. Ce problème est inhérent à tous les générateurs qui mélangent le vrai aléa à du pseudo-aléa et qui ensuite brouillent encore plus le résultat avec une fonction cryptographique forte. De cette façon nous ne savons plus si le générateur nous fournit une suite vraiment aléatoire ou une suite déterministe ou plus probablement un mélange des deux mais dont on ignore la proportion.

4.6 TRNG proposé par Kwork *et al.*

Le principe de TRNG décrit dans [KL06] utilise comme source d'aléa le jitter présent dans une DFS de la famille Xilinx. Le principe reste le même que dans [FMC85], un signal jitté basse fréquence va échantillonner un signal haute fréquence. Le signal haute

fréquence est obtenu par un signal externe de 270 Mhz (Figure 1.17). Le signal basse fréquence est lui généré par une DFS avec des paramètres de multiplication et division $m = 31$, $d = 32$ de façon à maximiser le jitter.

Les auteurs ne discutent pas du type de jitter présent dans la DFS. De plus le rapport entre F_l et F_h n'est pas discuté mais donné empiriquement. Nous montrerons dans le Chapitre 2 que le jitter issu d'une DFS peut présenter des composantes déterministes importantes.
A nos yeux, le signal obtenu à la sortie de la bascule contenant l'aléa et dont une évaluation est donnée dans l'article présente trop peu de variations (*c.f.* Table 1. Distribution of period of F_l) dans [KL06].

C'est pourquoi les auteurs proposent d'utiliser comme dans [FMC85] un post-traitement correcteur. Les paramètres retenus par les auteurs sont un filtre de parité d'ordre 2 et une fonction de hachage MD5 avec un taux de compression de 2, ce qui donne un débit de 6.05 Mbs. Les tests statistiques (NIST SP800-22) sont appliqués après la sortie de la fonction de hachage et sont passés avec succès. Etant donné le prix à payer pour la fonction MD5 les auteurs suggèrent l'utilisation d'un filtre de parité d'ordre beaucoup plus élevé, ce qui se traduit par une diminution du débit. De plus, une évaluation en termes de ressources matérielles de cette solution n'est pas fournie.

En conclusion on peut remarquer que c'est le même principe historique que dans [FMC85], où une source d'aléa faible est brouillée de manière déterministe pour acquérir des qualités statistiques nécessaires mais au prix d'une fonction de hachage coûteuse.

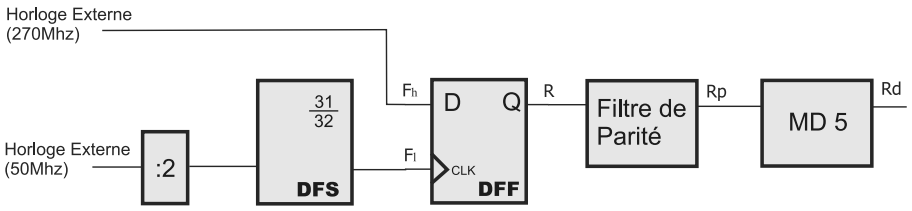

Figure 1.17 TRNG proposé par Kwork utilisant un conditionneur de fréquence DFS.

4.7 TRNG proposé par Golic

Golic dans [Gol06] propose un générateur basé sur des circuits asynchrones contre-réactionnés (*asynchronous circuits with feedback*). Ce sont des registres à décalages avec rétroaction linéaire modifiés (*LFSR*). En effet au lieu d'utiliser des bascules synchrones, Golic propose d'utiliser des inverseurs dans une structure auto oscillante (asynchrone) proche d'un oscillateur en anneau et qui respecte l'architecture d'un LFSR. Le principe de ces nouvelles structures est représenté à la Figure 1.18. Deux structures sont proposées *Fibonacci Ring Oscillator - FIRO* et *Galois Ring Oscillator - GARO* qui reprennent les structures des LFSR de même nom. Les FIRO et GARO sont caractérisés par leur

polynôme qui définit la boucle de retour. Comme on peut le voir à la Figure 1.18 à chaque inverseur correspond un interrupteur f_i qui peut être soit fermé $f_i = 1$ soit ouvert $f_i = 0$. Le comportement est donc défini par le polynôme donné par l'équation 1.2 avec la condition donnée par l'équation 1.3 pour que la boucle soit effective.

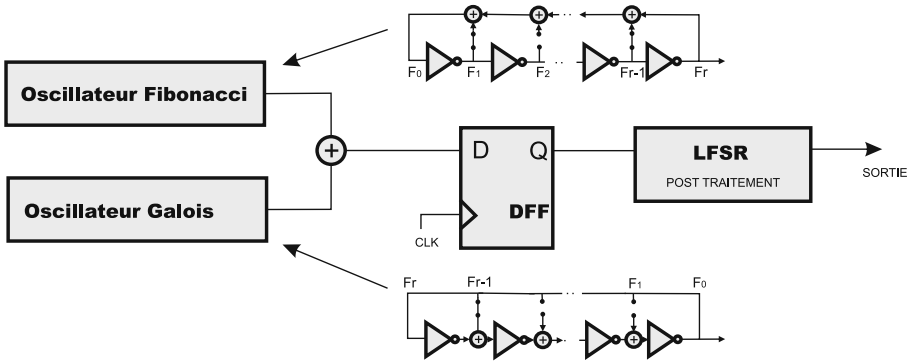

Figure 1.18 TRNG proposé par Golic basé sur des registres à décalages non linéaires modifiés.

$$f(x) = \sum_{i=0}^{N} f_i x^i \tag{1.2}$$

$$f(0) = f(r) = 1 \tag{1.3}$$

Golic dans [Gol06] donne les conditions et les preuves mathématiques sur les polynômes pour éviter que les oscillateurs restent indéfiniment dans un même état avec des sorties constantes ce qui serait catastrophique pour le générateur.

La sortie de ces structures auto-oscillantes présente ainsi un mélange de pseudo aléa et de vrai aléa. En effet si aucune source de bruit n'était présente, les délais des inverseurs seraient constants et invariables dans le temps et un comportement totalement déterministe en résulterait, identiquement à un LFSR de même polynôme. Mais puisque les délais ne sont pas constants, à cause des mêmes phénomènes qui donnent naissance au jitter un comportement aléatoire est vite atteint, ce que montre l'auteur en comparant plusieurs redémarrages à partir des mêmes conditions initiales. En effet très rapidement les signaux sont différents d'un redémarrage à l'autre.

Les deux signaux issus des GARO et FIRO sont additionnés modulo 2 avant d'être échantillonnés par un oscillateur. La sortie de cet échantillonneur présente malheureusement et étonnamment une assez mauvaise qualité avec un fort *biais*. Un post-traitement à base de LFSR est donc nécessaire pour la correction de ce *biais*. L'origine de ce *biais* n'est pas expliqué dans cet article, et sa présence est surprenante car le signal issu d'une structure GARO et FIRO devrait présenter un taux équilibré de bits à l'état haut et bas.

4.7.1 Améliorations proposées par Dichtl et Golic

Dans [DG07] Dicht et Golic reprennent le générateur présenté dans [Gol06] en y ajoutant quelques modifications. Le générateur est redémarré dans les mêmes conditions initiales avant chaque nouveau bit généré. Les chances d'obtenir une suite statistiquement indépendante sont ainsi augmentées.

L'extraction de l'aléa est également modifiée, une bascule est interposée entre la sortie du GARO et la bascule d'échantillonnage. Elle est cablée en mode *toggle* ou comme diviseur par deux ou comme compteur modulo 2. Elle compte les transitions 0-1 effectuées par le signal oscillant du GARO pendant un cycle du signal d'échantillonnage *Clk*. Son but est de diminuer un éventuel biais.

Les deux structures de générateurs proposés dans [Gol06] et [DG07] sont très intéressantes car elle proposent une solution élégante pour obtenir un signal aléatoire à la limite du signal analogique dans un FPGA. Cependant le fait que le vrai aléa est mixé à du pseudo-aléa rend très difficile l'évaluation de ce générateur au regard de la quantité de vrai aléa collectée. Aucune description mathématique n'est fournie par les auteurs, ce qui diminue l'impact scientifique de ce principe.

4.8 TRNG proposés par Cret *et al.*

Cret *et al.* ont récemment proposé plusieurs principes de TRNG destinés à des cibles FPGA. Cependant mis à part [GCS09] ils n'introduisent pas de nouveau principe de génération mais proposent des améliorations et des modifications aux principes de fonctionnement existants.
Dans [KCS08] ils proposent un TRNG basé sur des oscillateurs en anneau qui reprends des notions introduites par [SMS07].
Dans [GCS09] ils proposent un TRNG implanté dans une cible Xilinx où l'aléa provient d'un conflit d'écriture dans une mémoire RAM embarquée à deux entrées (*Dual Port*).

Dans [CSG08] les auteurs reprennent le principe proposé par Kohlebrenner et Gaj et proposent de palier au problème du besoin de placement manuel en utilisant un post-traitement robuste mais déterministe.

Dans [ISC09] les auteurs proposent toujours un TRNG basé sur le jitter mais cette fois un système de calibration automatique est proposé.

4.9 TRNG proposé par Danger *et al.*

Danger *et al.* proposent dans [DGH07] un TRNG qui exploite la métastabilité comme source d'aléa. Le principe de fonctionnement est représenté à la Figure 1.19 L'état métastable d'une bascule *latch* est obtenu en injectant un signal et sa version retardée dans une bascule D-latch implantée dans un élément LUT du FPGA. Le signal

d est séparé et empreinte deux chemins différents : un chemin d'horloge et un chemin de données. Le signal empruntant le chemin de données rentre dans un réseau de délais en série constitués par les délais intrinsèques des fils d'interconnexion. Le réseau de délais est constitué de n éléments de retards et chaque sortie est échantillonnée par le signal d. En mettant un grand nombre n de délais et de bascules il est probable qu'un certain nombre de ces bascules rentrerons dans un état métastable et produiront de l'aléa. Les n sorties des bascules sont ensuite "xorées" et échantillonnées par deux bascules pour éviter que l'état métastable ne se propage.

Pour que ce TRNG fonctionne, il faut garantir que la sortie ne reste pas indéfiniment dans le même état. Il faut donc qu'à chaque cycle de d au moins une des n bascules rentre dans un état métastable et ce quelque soient les variations de température, de tension ou de dispersion des paramètres physiques du circuit cible. En effet les valeurs des délais d'interconnexions sont fortement dépendants de ces variations.

Une version plus évoluée est présentée dans [DGH09] dont la description est donnée à la Figure 1.20. Dans cette version, afin de garantir l'occurrence de l'état métastable d'au moins une des bascules, un calcul est appliqué au contrôle dynamique d'un pré délai comme présenté à la Figure 1.20.

Cependant ce principe présente un biais dans la suite aléatoire obtenue. L'utilisation d'un correcteur est ici aussi nécessaire. Danger *et al.* utilisent un correcteur de Von-Neumann et obtiennent un débit important de 20Mbs. Les suites aléatoires obtenues à ce débit passent les tests statistiques NIST SP800-22.

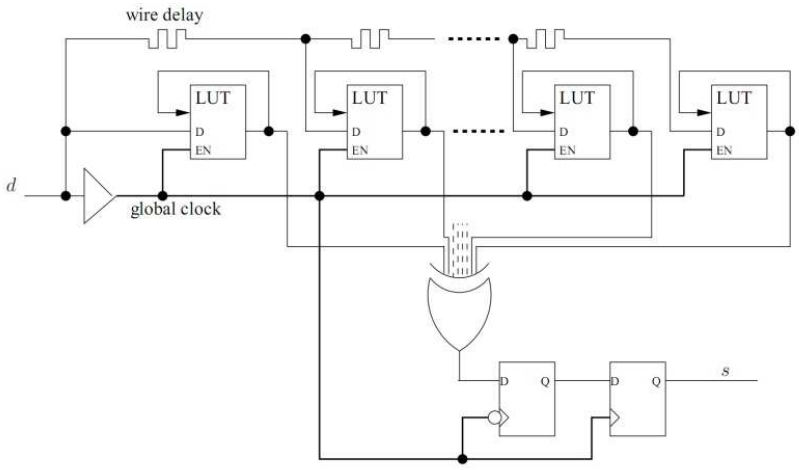

Figure 1.19 TRNG proposé par Danger basé sur la métastabilité de bascules Latch réalisées dans des LUT.

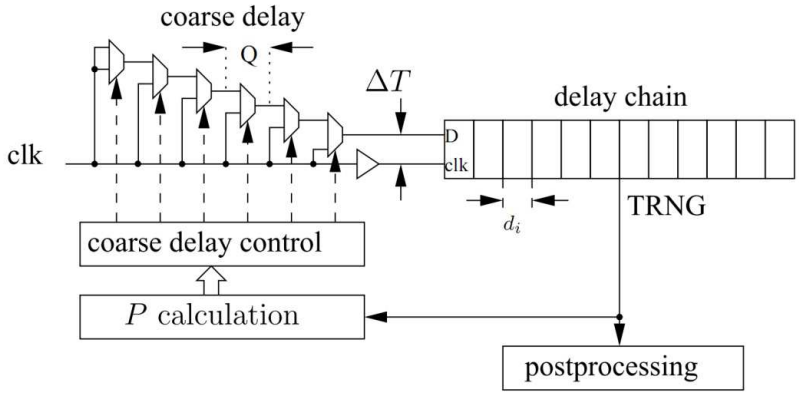

Figure 1.20 Variation sur le TRNG de Danger basé sur une boucle ouverte.

4.10 TRNG proposé par Vasyltsov *et al.*

Vasyltsov *et al.* dans [VHKK08] proposent un TRNG innovant utilisant non pas le jitter mais l'état métastable provoqué volontairement dans un oscillateur en anneau. Les auteurs proposent d'utiliser le terme de META-RO. L'état métastable se caractérise en électronique par un état transitoire de nature aléatoire et dont la durée est indéterminée. L'échantillonnage d'un signal dans un état métastable serait donc intéressant pour la génération d'aléa. Le principe de fonctionnement ainsi que les chronogrammes sont

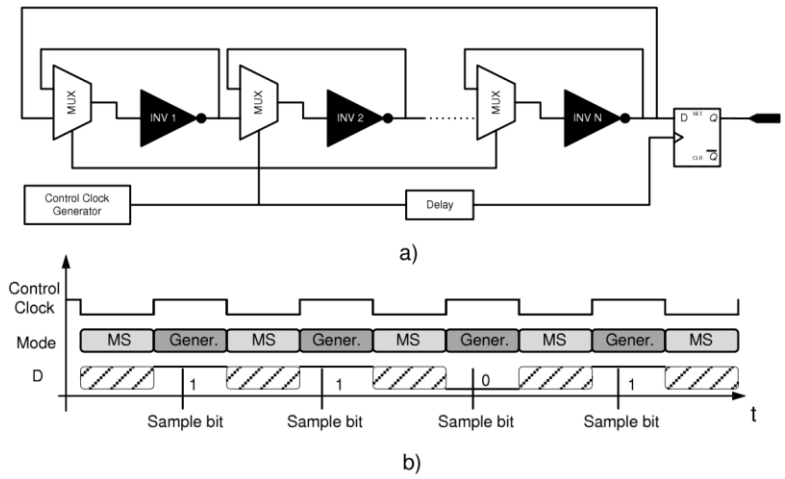

Figure 1.21 Principe de TRNG proposé par Vasyltsov basé sur des oscillateurs en anneau forcés en état métastable.

présentés à la Figure 1.21. Deux états sont provoqués de manière cyclique par un signal de contrôle *Clk*. Lorsque ce signal est à l'état bas, les inverseurs sont maintenus déconnectés les uns des autres et leurs sorties sont rebouclées sur eux mêmes (état *MS*). D'après les auteurs après un certain temps dans cet état la sortie de chaque oscillateur converge vers un état métastable. Ainsi d'après [VHKK08] chaque inverseur

forme une source de bruit indépendante. Dans un second temps lorsque le signal *Clk* est à '1', les inverseurs sont reconnectés en boucle et forment de nouveau un oscillateur en anneau normal. Cependant les oscillations reprennent après un certain temps transitoire qui dépend des conditions initiales. Puisque tous les inverseurs étaient dans un état métastable, ce temps transitoire est aléatoire et différent à chaque nouvel état. L'échantillonnage final se fait ensuite après un délai lorsque le régime d'oscillation est stabilisé. Puisque le temps transitoire est différent à chaque fois, le signal à la sortie sera aléatoire.

Ce principe de TRNG a été évalué par les auteurs dans des cibles ASIC et FPGA. Cependant très peu de détails sont fournis quant à son implantation FPGA. Les auteurs évaluent les suites aléatoires obtenues avec et sans post-traitement. On s'aperçoit que sans post-traitement les tests statistiques FIPS 140-2 et AIS 31 ne passent pas systématiquement, témoignant d'une mauvaise qualité de la suite aléatoire. Un correcteur de Von Neumann est donc utilisé. Les auteurs remarquent également l'extrême sensibilité de ce générateur aux conditions environnementales comme la température par exemple. Les débits potentiellement obtenables semblent être élevés, cependant dans cet article, seulement des valeurs correspondantes à des implantations ASIC sont discutées.
Ce générateur reprend en partie des idées présentées par Epstein *et al.* dans [EHK+03] quelques années auparavant. L'utilisation de l'état métastable et le fonctionnement du TRNG en deux phases. Cependant dans [EHK+03] aucun détail d'implantation n'est proposé.

4.11 Autres générateurs

Dans [DJ00] les auteurs présentent une étude interessante sur la terminologie utilisée dans le domaine de recherche appliqué aux TRNG, ainsi qu'une évaluation du TRNG utilisé par la société *Infineon*.
Dans [TAH06] est présenté un autre générateur basé sur les PLL dans les FPGA assez similaire à celui présenté par Fischer et Drutarovsky dans [FD03].

5 Comparaison des TRNG

La génération d'aléa dans les circuits logiques reconfigurables est en constante évolution. En effet de plus en plus de nouveaux principes sont proposés. Afin de montrer les influences mutuelles au cours du temps des différents principes, nous avons synthétisé l'évolution de la génération d'aléa au cours du temps. Ces influences sont représentées à la Figure 1.22 dans l'ordre chronologique croissant de gauche à droite. Lorsqu'un nouveau principe reprend directement un principe existant nous avons montré le lien entre les deux générateurs avec un trait plein. Lorsqu'un générateur a des éléments de similitude avec un principe existant même s'il est différent (même source d'aléa et même extraction) nous avons représenté l'influence par un trait en pointillé.

La Figure 1.22 nous montre plusieurs choses. D'abord le fait que la majorité

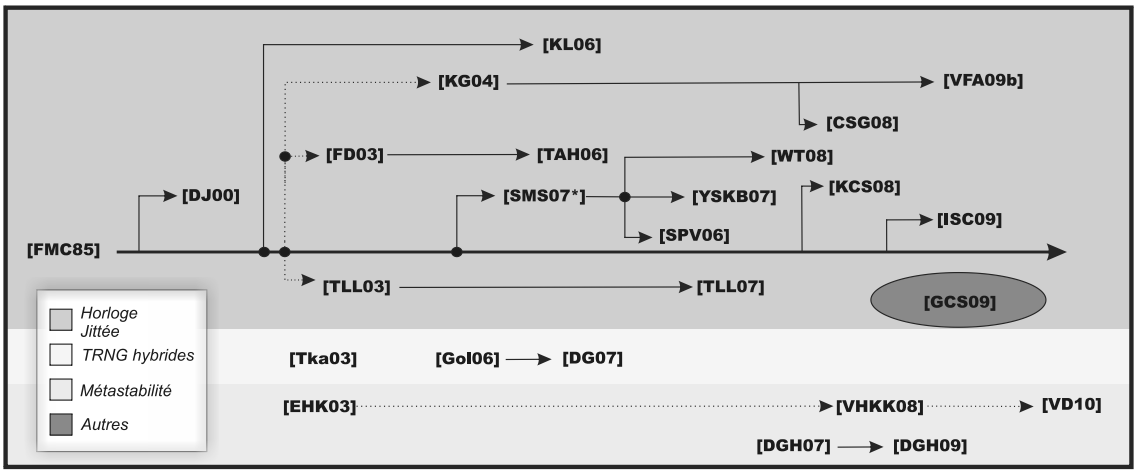

Figure 1.22 Classification des TRNG selon leur influences historiques et selon la source d'aléa utilisée.

des principes sont basés sur l'échantillonnage de signaux périodiques jittés. Ils sont directement inspirés du principe historique [FMC85]. Ce fait est intéressant car avec un tel nombre d'articles d'auteurs différents, ce principe est de mieux en mieux compris et documenté. Deuxièmement on voit que depuis quelques années de plus en plus de TRNG utilisent une source alternative d'aléa, peut être sous-estimée jusqu'à présent - la métastabilité, créant ainsi un groupe parallèle de TRNG.

Comme nous l'avons montré il existe un grand nombre de TRNG, avec chacun des avantages et des inconvénients. Il n'existe pas à ce jour de générateur parfait, et peut être il n'en existera jamais. La comparaison entre différents TRNG n'est pas aisée car les implantations sont présentées dans des cartes différentes et sur des puces différentes. Une évaluation des différents TRNG est hors du sujet de cette thèse, nous nous limiterons à présenter une vue d'ensemble sur le plus de principes existants pour mieux cerner les niches à explorer pour faire progresser la recherche dans le domaine de la génération d'aléa. De plus une évaluation n'est possible que si les différents TRNG à évaluer sont implantés sur la même puce avec le même environnement immédiat (alimentation, carte PCB, rayonnement électromagnétique environnant, etc...).

Nous avons ainsi synthétisé un comparatif des principaux TRNG étudiés dans cet état de l'art dans le Tableau 1.2 selon différents critères qui à nos yeux sont les plus importants bien que d'autres critères peuvent être aussi ajoutés (état interne ou pas, possibilité de tests en ligne, coût matériel, consommation, robustesse face aux attaques, etc...) :

Source d'aléa utilisée Il est essentiel de connaître la source d'aléa utilisée dans un TRNG afin de pouvoir en déduire le comportement du générateur ainsi que sa robustesse

face à d'éventuelles attaques. Si la source d'aléa n'est pas clairement identifiée cela rendra une future modélisation quasi impossible.

Caractérisation de la source d'aléa Nous nous sommes intéressés si les concepteurs avaient proposé une caractérisation de la source d'aléa, et si oui si elle était pertinente au regard des travaux exposés dans le Chapitre 2. En effet nous montrerons dans ce chapitre qu'il est facile de surestimer la source d'aléa si celle-ci est le jitter par exemple. Ce point est également essentiel pour une future modélisation du TRNG.

Qualité de la suite aléatoire avant post-traitement A nos yeux ce paramètre est essentiel. En effet une mauvaise qualité avant post-traitement, combiné avec un post-traitement fort donnera une fausse impression de qualité du TRNG. En effet il sera très difficile dans ce cas d'évaluer la quantité de vrai aléa produit par le TRNG.

Post-traitement utilisé Nous présenterons les différents post-traitements en détails dans la Section 6 de ce chapitre.

Débit Le débit est un paramètre dont l'importance peut être variable selon l'application envisagée. Cependant il faut être conscient qu'il dépend fortement de la technologie utilisée pour la réalisation du TRNG.

Facilité d'implantation Pour une production industrielle, il est impératif d'avoir une implantation qui dépende le moins possible du type de FPGA utilisé et qui demande le moins possible, voire pas du tout une intervention manuelle de la part du concepteur, afin de pouvoir proposer des IP commercialisables.

Existence d'un modèle mathématique L'existence d'un modèle mathématique est à nos yeux très importante. En effet ce modèle garanti une compréhension des mécanismes mis en place pour collecter l'aléa. De cette façon on peut prédire le fonctionnement du TRNG et ne pas le considérer comme une boîte noire.

Modèle mathématique possible Certains TRNG bien que n'étant pas décrits originellement avec un modèle mathématique sont modélisables avec plus ou moins de facilité. D'autres semblent difficilement modélisables.

TRNG	Source d'aléa	Caractérisation	Qualité avant post-traitement	Post-traitement	Débit	implantation	Model existant	Model possible
FMC85	jitter osc. externe	oui	faible	filtre de parité	faible	facile	oui	oui
TLL03	jitter + pseudo aléa osc. externe	non	faible	filtre de parité	Kbs	facile	non	oui
TLL07	jitter + osc. anneau	non	pas discutée	LFSR	hybride	facile	non	oui
FD03	jitter PLL	oui	bonne	sans/XOR Corrector	Kbs	facile - PLL	oui	oui
Tka03	jitter osc.anneau + pseudo aléa	non	LFSR + CASR	TRNG hybride	variable	facile	non	oui
KG04	jitter osc. anneau	non	moyenne	Von Neumann	591Kbs	complexe	non	oui
SMS07	jitter osc. anneau	non	moyenne	fonction résiliente	Mbs	facile	oui (contesté)	difficile
KL06	jitter DFS	non	très faible			DFS	non	difficile
Gol06	jitter osc. anneau + pseudo aléa	non	sans	TRNG hybride		facile	non	oui
DGH07	métastabilité	non	non discutée	XOR Corrector/LFSR	Mbs	très complexe	non	très difficile
VHKK08	métastabilité	non	moyenne	oui	140Mbs	complexe	non	très difficile

Tableau 1.2 Comparaison des différents principes de TRNG présentés en détail dans le présent état de l'art.

6 Correcteurs et post-traitement

Souvent dans la réalité le signal numérique aléatoire est imparfait. C'est une suite de bits vraiment aléatoires (qui dépendent d'une source physique) mais dont la probabilité des bits à '1' et '0' n'est pas équilibrée, *i.e.* elle présente un *biais*. De plus plusieurs bits peuvent être corrélés. Or pour pouvoir utiliser l'aléa contenu dans cette suite à des fins cryptographiques, on voudrait tendre vers un biais nul et vers une distribution statistiquement indifférentiable de la distribution uniforme [BST03] (pour des mots de longueurs fixés). Pour arriver à ce résultat, plusieurs techniques existent dont les principales et les plus souvent utilisées sont présentées.

D'un point de vue théorique, il existe une vaste littérature scientifique qui traite du domaine de l'extraction d'aléa aléatoires à partir de suites imparfaites *randomness extractors*. Cependant on peut s'étonner de constater que ces méthodes existantes sont soit trop complexes, et donc coûteuses en termes de ressources matérielles, soit simplement ignorés par les concepteurs de TRNG. De fait, dans la pratique on retrouve très souvent les mêmes correcteurs. Citons un état de l'art très complet des extracteurs logiques [NTS99] ou encore [JJSH00].

Dans le cadre de cette thèse nous ferons la différence entre un extracteur d'aléa (*randomness extractor*) logique, qui opère sur des suites binaires, d'un *extracteur physique* d'aléa qui aura pour fonction de collecter l'aléa d'une source physique et la transformer en une suite binaire.

6.1 Correcteur de Von Neumann

Von Neumann est un des premiers à s'intéresser à la correction de suites aléatoires. Le principe présenté dans [VN51] est connu sous le nom d' *extracteur de Von Neumann* ou *correcteur de Von Neumann*. Le principe consiste à considérer des paires de bits consécutifs X_{2n} et X_{2n+1}. Si les deux bits de la paire n sont différents on produit à la sortie du correcteur X_{2n}. Si les bits de la paire sont identiques on ne produit rien à la sortie et on passe à la paire suivante. De cette manière des longues suites de '0' ou de '1' sont éliminés.

Le correcteur de Von Neumann est simple et peu coûteux en termes de ressources matérielles. Cependant il corrige effectivement le biais uniquement à condition que les bits, bien que biaisés soient indépendants. Or souvent dans les suites aléatoires imparfaites des corrélations peuvent survenir. L'autre désavantage lié à l'utilisation de ce correcteur est le fait que le débit n'est pas constant, ce qui peut être contraignant dans un système embarqué à cause du surcoût matériel pour la logique de contrôle (registres, états etc...).

On peut montrer que pour une suite $b_1, b_2, ..., b_n$ de n bits en entrée avec $p = Pr(b_i = 1) = 1 - Pr(b_i = 0)$ on peut obtenir N bits avec un *biais* proche

de 0, comme le montre l'équation 1.4.

$$N = np(1-p) \tag{1.4}$$

6.2 Filtre de parité

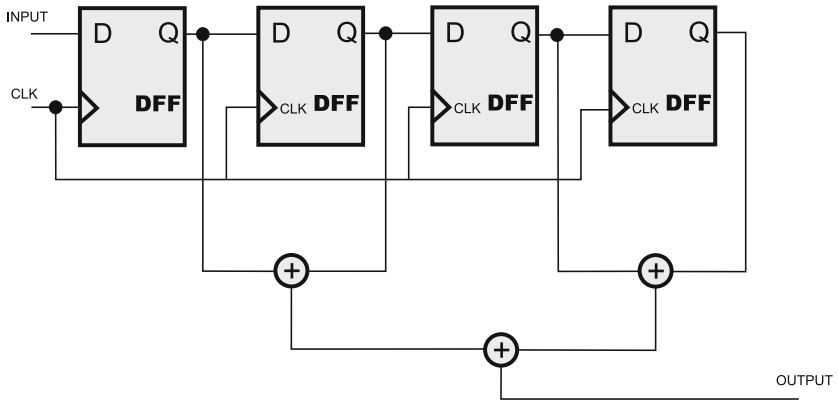

Figure 1.23 Filtre de parité pour corriger le biais statistique de suites aléatoires utilisé dans [FMC85].

Le principe de fonctionnement est décrit à la Figure 1.23. La suite de bits qui présente un biais, est passée par une suite de n bascules D (filtre d'ordre n) et les bits issues de ces bascules sont additionnés modulo 2 (XOR) pour fournir un seul bit à la sortie. Puisque les bits intermédiaires ne sont pas utilisés, le débit obtenu est divisé par n par rapport au débit initial à corriger.

On peut montrer que si $p = Pr(b_i = 1) = 1 - Pr(b_i = 0)$ à la sortie de ce filtre on aura :

$$p_0 = Pr(b_i = 0) = 0.5 + 2^{n-1}(p + 0.5)^n \tag{1.5}$$

$$p_1 = Pr(b_i = 1) = 0.5 - 2^{n-1}(p + 0.5)^n \tag{1.6}$$

Comme on peut le voir sur la Figure 1.24 un ordre 2 suffit pour corriger des biais relativement importants, alors que pour des biais très importants un ordre supérieur est nécessaire. Comme pour le correcteur de Von Neumann pour avoir une correction effective l'indépendance des bits est nécessaire. Une description plus détaillée est donnée dans [Dav02], le cas des bits corrélés est aussi traité.

6.3 Fonctions cryptographiques

Certaines implantations de TRNG utilisent des blocs cryptographiques connus pour produire un flux aléatoire uniforme en sortie. On utilise souvent des fonctions de hachage comme MD5 ou SHA-2. Il est également possible d'utiliser un chiffrement en bloc

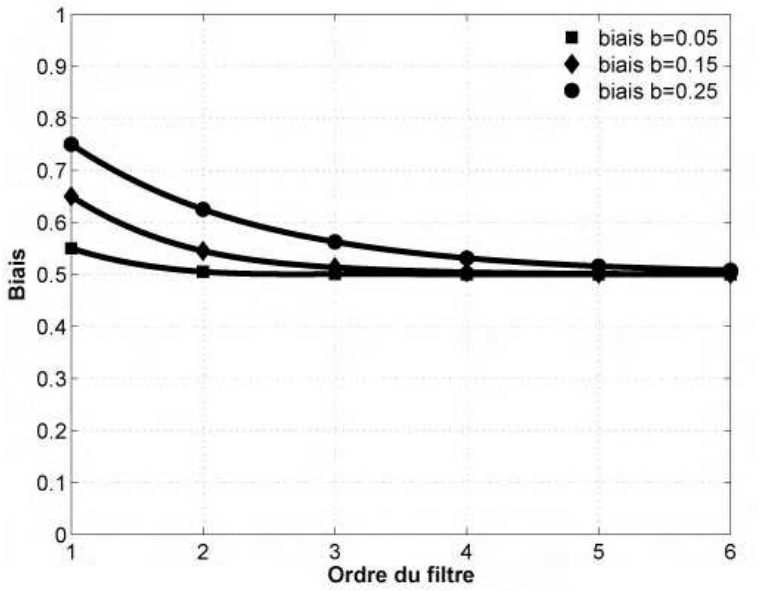

Figure 1.24 Correction du biais en fonction de l'ordre du filtre de parité.

(AES, DES) pour chiffrer la suite aléatoire à corriger, ainsi les propriétés de diffusion et de confusion des algorithmes de chiffrement par blocs sont utilisés. Cependant ces primitives cryptographiques sont coûteuses en termes de ressources matérielles et leur utilisation est *de facto* limitée.

Une solution consisterait à partager les ressources matérielles de la fonction cryptographique entre l'étape de chiffrement et la génération d'aléa.

6.4 LFSR

Comme nous l'avons vu dans la Section 4 de ce chapitre, plusieurs TRNG utilisent un post-traitement basé sur l'utilisation de registres à décalage à rétroaction linéaire. Les LFSR ont l'avantage d'être facilement implantables dans le matériel à un coût matériel faible. De plus leur fonctionnement est bien compris et peut être facilement modélisé. Cependant comme tous les post-traitements il masque completement la qualité de la source en amont.

6.5 Fonctions résilientes

Les fonction résilientes ou *resilient functions* en anglais sont des fonctions souvent utilisées en cryptographie et qui dérivent de la théorie des codes. Elles sont dérivées de la famille plus vaste des *fonction booléennes*. Elles jouent un rôle important dans la conception des algorithmes de chiffrement à clé symétrique. Nous conseillons la lecture de [Fon98] pour une étude théorique de ces fonctions.

Sunar *et al.* utilisent une telle fonction résiliente dans [SMS07]. D'après les auteurs, l'utilisation d'une telle fonction est particulièrement bien adaptée au post-traitement car la connaissance de m bits en entrée de la fonction ne permet pas de faire une meilleure prédiction de la sortie que de deviner.

Les fonctions résilientes opèrent par blocs et prennent en entrée des blocs de taille plus importante qu'elles ne fournissent en sortie, c'est pourquoi elles induisent également une forte diminution du débit.

6.6 Autres fonctions de post-traitement

Dichtl dans [Dic07] propose une méthode de post-traitement pour corriger le biais qui est très intéressante car elle est simple, puissante et peu coûteuse en matériel. Elle est également basée sur une opération XOR et prend un bloc de 16 bits en entrée et fournit 8 bits en sortie. Réduisant ainsi le débit uniquement de 2. Le principe de fonctionnement est présenté à la Figure 1.25.

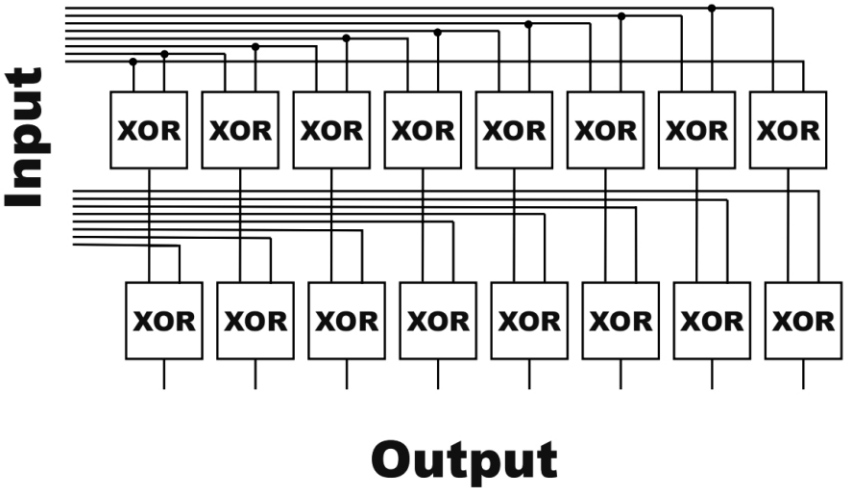

Figure 1.25 Post-traitement destiné à la correction du biais statistique proposé par Dichtl basé sur l'utilisation de portes XOR.

7 Attaques contre les TRNG

Comme tous les composants d'une chaîne cryptographique, les TRNG peuvent être sujets à des attaques visant à compromettre la sécurité du module cryptographique dans lequel ils ont été intégrés. Pour les TRNG implantés dans du matériel, ces attaques peuvent potentiellement être très dangereuses car les générateurs utilisent un processus aléatoire physique pour fournir des suites aléatoires. Or ces processus aléatoires physiques sont par définition très sensibles aux conditions de l'environnement : température,

tensions d'alimentation, environnement électromagnétique, etc... Un attaquant, qui dispose du module à attaquer, peut imposer des variations contrôlées de l'environnement pour modifier le comportement nominal du TRNG. Le but recherché est clairement la diminution de l'entropie de la suite aléatoire pour prévoir (ou même induire) les clés de chiffrement ou les challenges etc. Étonnamment il existe beaucoup moins d'attaques connues sur les TRNG que sur le reste de primitives de chiffrement. En effet, une littérature abondante existe sur le sujet de *l'injection de fautes* ainsi que sur les attaques *par canaux cachés* concernant des dispositifs cryptographiques embarqués. Nous citerons ici une attaque récente et particulièrement révélatrice [MM09]. Les auteurs réalisent une injection par le biais de l'alimentation d'un signal d'une fréquence perturbatrice d'un oscillateur embarqué dans une carte à puce. Ce signal a pour effet de verrouiller les oscillateurs en anneau qui sont utilisés comme source d'aléa. Une fois le verrouillage effectué, l'entropie est tellement faible qu'une attaque exhaustive de la clé devient triviale.

Dichtl dans [Dic03] propose une attaque sur le générateur de Tkacik, cependant elle reste théorique. A notre connaissance il n'existe pas de travaux scientifiques qui traitent du sujet de l'injection de fautes (par laser ou par *glitch*) sur les TRNG comme c'est le cas pour les primitives de chiffrement embarquées.

8 Méthodologie de test et certification des TRNG

Plusieurs normes existent pour évaluer et certifier la qualité de générateurs de nombres aléatoires pour des équipements cryptographiques. Généralement ces normes sont gouvernementales et définissent des classes d'utilisation avec des critères spécifiques pour chacune de ces classes. Cependant toutes les normes existantes et les méthodologies de tests proposés, à l'exception de la norme AIS31, ne sont pas spécialement conçus pour des générateurs vraiment aléatoires matériels mais pour des suites aléatoires pouvant être produites par des PRNG. C'est le cas des normes FIPS 140-2, NIST SP800-22 et DIEHARD. Ainsi ce qui est testé est la suite aléatoire finale obtenue, la méthode d'obtention elle n'est pas évaluée. Ce qui laisse la possibilité d'utiliser un générateur vraiment aléatoire faible ou de mauvaise qualité et de cacher ces défauts par une fonction déterministe complexe.

C'est pour cette raison que nous présenterons en premier la norme AIS31 Allemande qui s'applique spécialement aux générateurs vraiment aléatoires bien que dans la pratique les suites de tests statistiques NISTSP800-22 et FIPS 140-2 soient utilisées plus souvent pour valider la qualité des suites aléatoires. Dans un second temps nous présenterons les autres suites de tests proposés dans les normes principalement américaines NIST SP800-22 et FIPS 140-1 et 140-2.

8.1 AIS31

Le gouvernement Allemand, ou plus précisément le *BSI - Bundesamt für Sicherheit in der Informationstechnik* propose une méthodologie de tests et une certification de générateurs vraiment aléatoires embarqués dans du matériel cryptographique. Il est important de noter que c'est la première méthodologie de tests conçue spécialement pour des générateurs vraiment aléatoires contrairement aux suites NIST ou DIEHARD qui ont été développés pour valider des suites pseudo aléatoires et peuvent être utilisés pour des suites vraiment aléatoires. Plusieurs concepts novateurs sont donc proposés. La description complète de la norme AIS31 est présentée dans [KS01], [SK03].
Cette norme définit deux classes, $P1$ et $P2$ (et plusieurs sous classes qui ne se seront pas présentées ici). Pour chaque classe les applications sont définies et les qualités requises pour l'appartenance. La classe $P1$ est destinée entre autres à la fabrication de challenges et vecteurs d'initialisations, alors que la génération de clés confidentielles se fait par des équipements qui appartiennent à la classe $P2$. Pour chaque classe, neuf tests statistiques paramétriques (T_0 à T_8) sont définis. Les tests sont appliqués aux bits avant post-traitement et après avec des jeux de paramètres différents.
Des notions importantes sont définies dans la spécification de la norme. Ainsi les

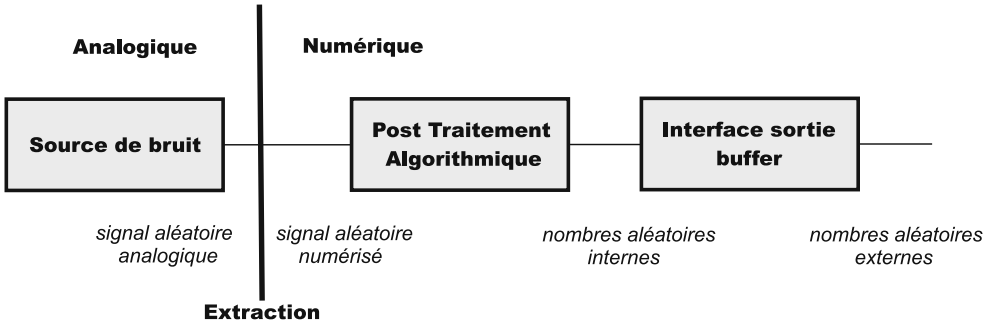

Figure 1.26 Formalisme et notions concernant les TRNG introduites par Killman et Schindler dans la description de la norme AIS31.

auteurs proposent une décomposition du TRNG en éléments comme présenté à la Figure 1.26. Au coeur du TRNG nous avons un signal analogique qui est de nature aléatoire et qui dépend d'un phénomène physique, ce signal est appelé *noise source*. Par un procédé d'extraction, le signal analogique est transformé en signal numérique appelé signal DAS ou *Digitalized Noise Signal*. C'est un signal numérique ou une suite de bits possédant une fréquence fixe et une certaine entropie. Si les qualités statistiques de ce signal ne sont pas satisfaisantes alors on lui applique une transformation déterministe qui est le post-traitement pour obtenir le signal aléatoire interne ou *Internal Random Numbers*. Ce dernier peut éventuellement traverser une circuiterie d'entrée sortie (buffer, fifo, etc...) pour devenir un signal aléatoire de qualité statistique cryptographique que l'on nomme *External Random Numbers*.

AIS 31 introduit également l'obligation de tests en ligne. Trois tests sont ainsi spécifiés : *Tot Test* qui est un test total et qui doit parvenir à identifier un arrêt éventuel

de la source d'aléa, *Startup Test* qui doit garantir l'intégrité et la qualité de la suite aléatoire au démarrage et *Online Test* qui doit garantir la qualité de la suite généré à la demande pendant le fonctionnement. La structure des tests n'est pas définie elle est laissée au concepteur du TRNG.

Une autre obligation pour avoir la certification AIS31 est la disposition d'une modélisation ou description mathématique du TRNG. Cette description doit fournir une description *plausible* de la source d'aléa et une modélisation des qualités statistiques de la suite obtenue. C'est par ce point que la norme AIS31 se différencie des normes NIST et FIPS.

Notons ici que la DCSSI française (La Direction Centrale de la Sécurité des Systèmes d'Information) a émis une note d'information intitulée *Utilisation de la méthode AIS31* dans laquelle elle préconise l'utilisation systématique de mécanismes de retraitement déterministes (PRNG) par dessus du TRNG. Ne faisant ainsi pas entièrement confiance à la modélisation de la source d'aléa qui d'après le DCSSI est délicate et dans la plupart des cas reposerait sur des hypothèses difficilement vérifiables.

8.2 Suites de tests statistiques

Les tests statistiques ont pour objectif de mesurer la qualité d'une suite aléatoire. Si, et seulement si, cette suite est issue d'un générateur vraiment aléatoire et qu'elle satisfait les tests statistiques alors nous pouvons conclure que cette suite est vraiment aléatoire et de bonne qualité.

Quelques problèmes sont tout de même rencontrés. De même qu'il est impossible de générer du vrai aléa par un procédé algorithmique, un test statistique ne peut en aucune façon garantir qu'une suite donnée est aléatoire. La seule information qu'un test statistique peut fournir c'est que la suite parait aléatoire. Or nous connaissons beaucoup de moyens simples de générer des suites qui ont l'air aléatoires (LFSR, CASR).

Une autre difficulté vient du fait que les TRNG sont souvent évalués par des batteries de tests statistiques après le post-traitement. Ceci principalement à des fins de marketing pour garantir une certaine norme. Cette évaluation post-traitement est certes nécessaire mais elle n'est pas adaptée pour vraiment évaluer la qualité d'un TRNG. Souvent la fonction de post-traitement est tellement forte que même si à l'entrée un signal complètement déterministe était présent, un compteur par exemple, à la sortie du post-traitement nous aurions une suite qui ressemble à une suite vraiment aléatoire. C'est pourquoi dans le cadre de cette thèse nous nous efforcerons à appliquer les tests statistiques avant tout post-traitement.

8.2.1 FIPS 140-1 et FIPS 140-2

Les normes FIPS 140-1 et FIPS 140-2 [Lab94] [Lab01] sont des normes gouvernementales américaines (crées par le *NIST - Nationnal Institude of Standards and Technology*) pour certifier le niveau de sécurité de modules cryptographiques, leur nom complet est d'ailleurs : *security requierements for cryptographic modules*. Elles ont été crées en 1994 et mises à jour en 2003. Elles définissent 4 classes ou niveaux de sécurité auxquelles peuvent prétendre des équipements cryptographiques ainsi que les conditions d'appartenance respectives et les applications autorisées par chaque classe. Nous nous intéresserons ici, uniquement aux 4 tests statistiques proposés pour les générateurs aléatoires. La norme FIPS 140-2 modifie plus sévèrement les critères ou les seuils numériques de passage des tests statistiques. Ces 4 tests, bien que pas très complets vis à vis de l'évaluation de la qualité statistique, sont un très bon point de départ lors de l'évaluation d'une suite aléatoire. En effet si ces 4 tests ne passent pas alors on peut être sur que la suite n'a pas les qualités requis et il ne sert à rien de poursuivre avec d'autres tests. Historiquement ces tests dérivent des travaux décrits dans [KS38].

Les 4 tests prennent en entrée 20 000 bits consécutifs et donnent en sortie un résultat binaire : test passé avec succès ou test raté. Il n'y a pas ici d'hypothèse à valider comme dans la suite NIST. Il faut que la suite de 20 000 bits passe les 4 tests en même temps pour garantir la qualité statistique. Si un seul des 4 tests n'est pas passé avec succès alors on peut conclure que la suite n'est pas de bonne qualité statistique. Ces quatre tests sont : *Monobit* ou *frequency test*, *poker test*, *runs test*, *long run test* dont nous donnons une brève description :

Monobit test Le *Monobit test* est le plus simple et un des plus importants. Il évalue simplement le *biais* statistique de la suite. Le *biais* est la déviation de la probabilité idéale d'obtenir un '1' ou un '0'. En effet une suite aléatoire de bonne qualité doit produire autant de bits à '0' que de bits à '1'. Ainsi la probabilité d'obtenir un bit à '1' doit être égale à la probabilité d'obtenir un bit à '0'. Ces deux probabilités doivent être idéalement égales à 0,5. Toute déviation dans ces probabilités de 0,5 est considéré comme un *biais*.

Puisque nous travaillons avec des processus aléatoires, une valeur de 0.5 finie n'a que peu de sens, ce qui importe c'est une valeur suffisamment faible du *biais*, c'est à dire un *biais* qui tends vers 0.

La *monobit test* compte simplement le nombre de bits à '1' n_1 et ce nombre doit être compris dans l'intervalle [9726 10274] pour la norme FIPS 140-2 et dans l'intervalle [9654 10346] pour la norme FIPS 140-1. Soit un *biais* inférieur à 0.0137 pour le cas de FIPS 140-2.

Le *biais* est un facteur très important pour les TRNG, car si un générateur fournit des suites aléatoires avec une probabilité plus importante de '1' ou de '0' et si ces

l	FIPS 140-1	FIPS 140-2
1	2267 - 2733	2343 - 2657
2	1079 - 1421	1135 - 1365
3	502 - 748	542 - 708
4	223 - 402	251 - 373
5	90 - 223	111 - 201
≥ 6	90 - 223	111 - 201

Tableau 1.3 Fourchettes de valeurs des *Runs* dans les normes FIPS 140-1 et 140-2.

suites sont utilisées pour générer des clés de chiffrement sécurisées alors une attaque par force brute de ces clés s'en trouve d'autant plus facilitée.

Poker test Le *Poker test* est un test d'indépendance des bits. Il est basé sur un test de χ^2.

Le séquence initiale de 20 000 bits est divisée en sous séquences de 4 bits. Ainsi 5000 séquences sont obtenues. Il existe 16 possibilités de combinaison de 4 bits. Soit n_i le nombre d'occurrences du bloc i avec $i \in \{0, 1, ..., 15\}$ le résultat donné par χ^2_{15}

$$X^2_{15} = \frac{2^4}{5 \times 10^3} \left(\sum_{i=0}^{15} n_i^2 \right) - 5 \times 10^3 \tag{1.7}$$

doit suivre une loi de χ^2 à 15 degrés de liberté et doit donc être compris dans l'intervalle]2.1646.17[pour satisfaire le *poker test*.

Run test Ce test fixe des seuils pour les suites consécutives (*runs*) de bits identiques. Le test est passé avec succès si le nombre de ces suites dans la séquence initiale de 20 000 bits est dans la fourchette fixée. Soit l la longueur d'une séquence de l bits identiques consécutifs. Les *runs* doivent alors satisfaire le tableau suivant :

Long Runs Ce test vérifie simplement qu'une suite de 26 bits identiques et consécutifs (*runs* de $l = 26$) n'existe pas dans la séquence initiale de 20 000 bits.

L'implantation des tests FIPS dans le matériel en vue d'un test en ligne est étudié dans [HKUK03], [VD09] et [SSR09]. Le test le plus coûteux en ressources matérielles est le *poker test* car il exige la somme de 16 élévations au carré. Il est montré dans [VD09] ainsi que dans [SSR09] une version qui fonctionne en nombres entiers pour ce test et éviter ainsi des représentations à virgule dans le matériel. Cependant nous noterons ici que les trois papiers cités ci-dessus travaillent sur des blocs de 20 000 bits consécutifs en mode bloc par bloc et non pas en continu. Leur but étant de tester la qualité des suites en ligne afin de pouvoir alerter un éventuel dysfonctionnement dans le

TRNG et ne pas utiliser la suite corrompue en tant que clé de chiffrement. Cependant avec ces implantations, il faudra attendre 20 000 bits pour dire si une suite est utilisable ou pas. Une implantation en fenêtre flottante serai préférable pour pouvoir garantir que chaque nouveau bit est issu d'une population de 20 000 bits conformes aux normes FIPS.

L'implantation des tests FIPS en logiciel est triviale et ne nécessite pas de compétences particulières en programmation ou en mathématique, et son exécution est immédiate sur un ordinateur moderne. Cela en fait un outil précieux pour une première validation de suites aléatoires.

8.2.2 Batterie de tests statistiques NIST SP800-22

Une section du NIST, la *Random Number Generation Technical Working Group (RNG-TWG)* travaille depuis 1997 à la description d'une batterie de tests statistiques. Ces tests sont au nombre de 16 et sont décrits dans [RSN+01]. Actuellement ce sont les tests statistiques d'aléa les plus évolués qui existent. Une version mise à jour en 2001 comporte uniquement 15 tests. Certains de ces tests sont paramétrables d'autre n'ont pas de paramètres. Contrairement aux tests FIPS, une méthodologie basée sur le formalisme du test d'hypothèses est proposée. Ces tests ne permettent en aucune façon de garantir ou de prouver qu'une suite est *vraiment* aléatoire, comme dans FIPS le mieux que l'on puisse déduire c'est que la suite en question *parait aléatoire*, c'est à dire qu'elle est de bonne qualité.

Un niveau de confiance est choisi (par exemple 99%) et un test d'hypothèse est effectué avec les données de la suite examinée. L'*hypothèse nulle* (H_0) est que la suite examinée *est* aléatoire et l'hypothèse inverse ou hypothèse alternative H_1 que la suite n'est pas aléatoire. Ces tests nous permettent ensuite de valider ou d'infirmer l'hypothèse nulle. Puisque nous travaillons avec des suites aléatoires, deux types d'erreur peuvent se produire : on peut s'attendre à ce qu'une suite réellement produite par un générateur de bonne qualité n'ait pas l'air aléatoire en vue des tests statistiques, de même qu'une suite issue d'un générateur non aléatoire ait les qualités nécessaires pour valider l'hypothèse H_0. Nous appellerons ces erreurs, erreurs d'ordre 1 et d'ordre 2, respectivement α pour l'erreur d'ordre 1 et β pour l'erreur d'ordre 2. C'est à cause de ces erreurs qu'un niveau de confiance est attribué. Pour chaque test statistique une $p - value$ est calculée. La $p - value$ est la probabilité qu'un générateur parfait ait généré une suite qui ait l'air moins aléatoire que la suite examinée par le test. Si la $p - value$ est supérieure à α l'*Hypothèse nulle* est acceptée, si elle est inférieure à α elle est rejetée. Pour des applications cryptographiques α est généralement choisie dans l'intervalle [0.001, 0.01]. Ainsi à un niveau de confiance α de 0.001 on s'attendrait à ce qu'une séquence seulement parmi 1000 soit reconnue comme n'ayant pas l'air aléatoire alors que en fait elle est aléatoire. Pour une $p - value > \alpha = 0.001$ on peut conclure, à $\alpha^{-1} = 99.9\%$ de chances, que la séquence à l'air aléatoire alors que si la $p - value$ est $\leq \alpha = 0.0001$ alors à $\alpha^{-1} = 99.9\%$ la suite n'a pas l'air aléatoire.

Le choix de α est important car il va déterminer le nombre de suites à tester et donc la taille des données en entrée de la batterie de tests. Chaque test prend en entrée des séquences de longueur différente, notée m. Ainsi nous aurons besoin d'au moins $n = \alpha^{-1}$ séquences de m bits pour les tests. Plus le nombre et la taille des séquences sont importantes et plus le temps d'exécution des tests est important.

Aussi une $p - value > \alpha$ n'est pas suffisant pour valider l'hypothèse nulle : il faut de plus que les valeurs des $p - values > \alpha$ soit uniformément réparties dans l'intervalle $]\alpha, 1]$ Le Tableau 8.2.2 regroupe les informations sur les tests statistiques utilisés. Pour chaque test le NIST donne des suggestions pour la longueur des séquences ainsi que sur les paramètres éventuels du test. Le test qui demandes les séquences les plus longues est le *Maurer Universal Test* qui requiert des suites individuelles (au nombre de α^{-1}) dont la longueur peut atteindre 10^9 bits. Ce test à initialement été décrit dans [Mau92] par Maurer.

Nom du test	
The Frequency (Monobit) Test	Non paramétrique
Frequency Test within a Block	Paramétrique
The Runs Test	Non paramétrique
Tests for the Longest-Run-of-Ones in a Block	Non paramétrique
The Binary Matrix Rank Test	Non paramétrique
The Discrete Fourier Transform (Spectral) Test	Non paramétrique
The Non-overlapping Template Matching Test	Paramétrique
The Overlapping Template Matching Test	Paramétrique
Maurer's "Universal Statistical" Test	Paramétrique
The Linear Complexity Test	Paramétrique
The Serial Test	Paramétrique
The Approximate Entropy Test	Paramétrique
The Cumulative Sums (Cusums) Test	Non paramétrique
The Random Excursions Test	Non paramétrique
The Random Excursions Variant Test	Non paramétrique

Tableau 1.4 Liste des tests statistiques dans la suite NIST SP800-22.

Les tests statistiques NIST ont été utilisés pour valider la qualité de *diffusion* du standard de chiffrement AES. A cette occasion Kim *et al.* dans [KUH04] remarquent que les tests *Discrete Fourrier Transform* et *Lempel-Ziv* sont faux et sont donc entièrement repris.

Le NIST fournit une implantation logicielle de la suite de tests statistiques gratuitement sur internet. L'exécution de la batterie complète de tests peut prendre un certain temps sur des ordinateurs, même récents. Contrairement au code des tests FIPS, comprendre, refaire ou valider l'exactitude d'un code pour la suite NIST nécessite des connaissances approfondies en mathématiques et en informatique et peut demander un

temps considérable, c'est pourquoi la communauté utilise le code fourni par le NIST. Cependant celui-ci n'engagent aucune responsabilité sur la qualité de ce code et même ont signalé en 2009 que le code du test FFT est faux et que les résultats de ce test ne doivent pas être pris en compte.

Dans [Sot99] Soto *et al.* donnent une évaluation critique de la méthodologie d'évaluation statistique de suites aléatoires proposé par le NIST.

8.2.3 DIEHARD et DIEHARDER

DIEHARD est une suite de tests statistiques pour tester la qualité de suites aléatoires. Elle a été proposée par George Marsaglia en 1995. Avant la suite NIST c'était la suite la plus avancée dans ce domaine. DIEHARDER est une libraire en C qui implémente la suite de tests. Cette suite consiste en 12 tests statistiques dont la description est donnée dans [Mar96a]. Cette suite de tests a été popularisée par une distribution sous forme de CDROM avec le code et un logiciel pour tester les nombres aléatoires [Mar96b].

Les résultats des tests de cette suite fournissent également des $p - values$ en sortie dont la distribution doit être uniforme comme la suite NIST. Cependant contrairement à la suite NIST, les tests sont considérés comme passés avec succès si la $p - value$ est comprise dans l'intervalle $[0 + \frac{\alpha}{2}, 1 - \frac{\alpha}{2}]$ où α est le niveau de confiance choisis. Dans [MT02] Marsaglia *et al.* donnent la description de 3 tests statistiques qui sont particulièrement difficiles à passer et proposent ainsi de réduire le nombre de tests de la suite à 3.

8.2.4 Autres tests statistiques

Schindler dans [Sch01] donne une méthodologie de test de suites issues de TRNG. Dans cet article l'auteur étudie une nouvelle procédure de test en ligne, c'est-à-dire en continue.

Knuth dans son célèbre livre *The art of programmig* [Knu73] donne la description de plusieurs tests statistiques.

Pierre l'Ecuyer maintien une libraire en langage C pour le test d'uniformité de nombres aléatoires. Cette librairie s'appelle *TestU01*

9 Conclusion

Nous avons montré dans ce chapitre que la génération d'aléa dans les circuits logiques joue un rôle important dans les cryptosystèmes embarqués. Cette problématique qui est celle de notre thèse est un domaine actif de recherche pour la communauté scientifique. Nous avons présenté dans ce chapitre un état de l'art de la génération d'aléa dans les circuits logiques programmables FPGA. En effet les principaux articles

scientifiques disponibles dans la communauté ont été recueillis et les plus importants ont été brièvement décrits avec un apport critique sur leurs performances. Nous avons également décris certains principes qui ne sont pas implantables dans des circuits de type FPGA mais nous avons montré qu'ils jouent un rôle important dans la conception de TRNG car beaucoup de principes reposent sur ces architectures historiques.

Nous avons organisé la description des TRNG qui à nos yeux sont les plus pertinents dans un ordre chronologique dans un premier temps. Ce sont les TRNG qui sont directement implantables dans n'importe quel circuit FPGA. La présentation chronologique à été retenue car elle permet entre autre de montrer l'évolution des idées dans la communauté scientifique. Nous avons également montré les différentes influences que certains TRNG ont eus sur d'autres principes de génération. Cette organisation historique avec les liens entre les différents articles est à nos yeux originale. Nous y voyons clairement le fait que la majorité des TRNG décrits dans par la communauté scientifique utilisent le jitter comme source d'aléa.

Dans un second temps une confrontation entre les principes a été proposée sur différents critères. Il apparaît clairement que très peu de générateurs disposent de modélisations mathématiques qui pourtant, à nos yeux, sont d'une importance cruciale. La modélisation est d'ailleurs un critère important dans l'obtention de la certification allemande AIS31. Elle permet de considérer le TRNG comme un système entièrement décrit et compris, et non pas comme une boîte noire dont on ignore le fonctionnement. C'est pourquoi nous avons travaillé sur la modélisation des TRNG et des sources d'aléa dans le cadre de cette thèse.
Un autre fait transparaît dans la confrontation entre les différents principes de génération d'aléa. Très peu d'entre eux sont dotés d'une caractérisation de la source d'aléa utilisée. Certaines descriptions tiennent pour acquis qu'il existe un phénomène aléatoire qui est le jitter sans le montrer encore moins le caractériser. Or une caractérisation est essentielle pour une modélisation mathématique. Nous montrerons dans le Chapitre 2 que les différentes sources utilisées dans la communauté scientifique présentent chacune des particularités et qu'elles sont un mélange de phénomènes vraiment aléatoires et de phénomènes déterministes.

Une synthèse des différentes sources de signaux avec du jitter disponibles dans les FPGA a été présentée. En effet nous baserons nos travaux principalement sur le jitter comme source d'aléa.

Nous avons également présenté les différentes méthodes déterministes utilisées pour corriger les suites aléatoires de mauvaise qualité, c'est ce que nous appelons le post-traitement. Nous avons ainsi évalué si les TRNG proposés utilisent un tel post-traitement, et si oui, si la qualité de la suite aléatoire a été évaluée avant celui-ci. En effet un post-traitement est nécessaire pour plusieurs raisons mais la qualité de la suite aléatoire doit être testée avant celui ci. En effet certains post-traitements sont tellement forts que même avec un signal périodique en entrée (suite "aléatoire" avant

post-traitement) on retrouverait à leur sortie un signal qui à l'apparence aléatoire. Or nous constatons que beaucoup de TRNG sont présentés justement avec des post-traitement forts et la qualité de la suite aléatoire n'est pas évaluée en amont. C'est pourquoi dans le cadre de cette thèse nous nous appliquerons à évaluer la suite aléatoire obtenue avant de faire un post-traitement.

Chapitre 2

Caractérisation des sources d'aléa : cas particulier du jitter

Dans cette partie, la caractérisation du jitter comme source d'aléa pour la réalisation de TRNG sera étudiée. Le jitter est la source la plus souvent utilisée pour la réalisation de TRNG car son origine vient en très grande partie d'un phénomène physique aléatoire - le bruit intrinsèque à tout composant électronique. Contrairement aux systèmes analogiques, dans un système entièrement numérique tel qu'un FPGA, où seulement les états haut et bas sont pris en compte, ce bruit est uniquement perceptible sous sa forme temporelle de jitter. Cependant le jitter est aussi fortement dépendant des perturbations externes qui sont souvent de nature déterministe. Une caractérisation spécifique est donc nécessaire pour proposer des modèles pertinents de TRNG. Dans ce chapitre, une définition du jitter sera présentée ainsi qu'une étude de caractérisation qualitative et comparative de plusieurs sources de signaux jittés disponibles dans les circuits FPGA.

1 Introduction

Comme nous l'avons montré dans le Chapitre 1, la majorité des articles qui proposent des TRNG basés sur le jitter ne recherchent pas une caractérisation de cette source. Bien que le jitter est en très grande partie aléatoire, il peut néanmoins présenter des composantes déterministes et périodiques. Dans ce chapitre, nous allons montrer que les jitter dans les différents composants qui sont généralement utilisés pour générer des signaux jittés ont tous des caractéristiques différentes.

Dans un système analogique, les différentes blocs travaillent tous avec des signaux analogiques. Un signal analogique est un signal dont l'amplitude de la grandeur qui porte l'information peut prendre une infinité de valeurs ou d'états. Ainsi on peut concevoir des amplificateurs qui amplifient le niveau faible du bruit, intrinsèque à tout

composant électronique, jusqu'à des niveaux importants. Ce bruit étant de nature aléatoire, il est ainsi plus aisé de concevoir des générateurs d'aléa. Cependant dans une système numérique, l'amplitude de la grandeur porteuse d'information est limitée à seulement deux valeurs possibles : un état logique haut et un état logique bas. Un tel système numérique est une vue de l'esprit, car en fait il est construit à partir d'un signal analogique continu dont l'amplitude est limitée à deux états seulement. Pour déterminer si le signal numérique est dans état haut ou dans un état bas, il est comparé à un seuil fixe. S'il est supérieur à ce seuil c'est un état haut s'il est inférieur à ce seuil c'est un état bas. C'est pourquoi le bruit n'aura que peu d'influence sur l'information véhiculée par le signal numérique, car son amplitude étant faible elle ne peut généralement pas faire changer la valeur d'un état. Par ailleurs un signal numérique est caractérisé par une fréquence, ou période. En effet l'état du signal est évalué périodiquement pour savoir s'il se trouve dans un état haut ou dans un état bas. Le seuil entre l'état bas et l'état haut étant fixe, le bruit sur le signal va modifier le passage dans le temps du seuil en modifiant la longueur d'un état bas ou d'un état haut. C'est ce phénomène que l'on nomme *jitter* et dont une définition technique sera donnée dans la Section 2 de ce chapitre.

Le jitter est ainsi la manifestation la plus évidente du bruit que l'on peut trouver dans un système numérique et donc dans un FPGA. Or le bruit est en très grande partie de nature aléatoire. C'est ce qui explique le fait que la plupart des TRNG sont basés sur ce phénomène.

Pour les applications conventionnelles de l'électronique, plus la vitesse de transmission des données numérique augmente plus le jitter est un problème à prendre en considération. Ainsi pour garantir le bon fonctionnement des transmissions obéissants à des normes de transmission série à très haut débit (fibre optique, bus SATA, SerDes, ...) il faut respecter des contraintes sur le jitter maximal. Dans ce cas, le jitter est considéré comme un phénomène qui nuit aux performances et dont on cherche à réduire l'amplitude au minimum. Pour garantir le bon fonctionnement du système, on ne s'intéresse pas à son évolution dans le temps ni à ses propriétés intrinsèques mais à son amplitude maximale. Or pour la génération d'aléa le jitter est au contraire considéré comme la source même de l'aléa, le coeur du TRNG. Ainsi nous avons besoin de connaître les caractéristiques du jitter pour pouvoir modéliser le comportement du TRNG.

Les TRNG qui utilisent le jitter comme source d'aléa sont généralement basés sur des signaux périodiques issues de différents composants : oscillateurs en anneau, boucles à verrouillage de phase, oscillateurs RC, etc... Or comme nous l'avons constaté dans la Section 5 du Chapitre 1, ces sources de jitter ne sont généralement pas caractérisées par les concepteurs de TRNG. La présence du jitter est juste montrée empiriquement comme des variations d'un des signaux internes au TRNG, la qualité de ce dernier est ensuite validée par des suites de tests statistiques.

Nous montrerons dans ce chapitre que le jitter issu des différents composants disponibles dans un FPGA, présente des caractéristiques et des propriétés très différentes. Nous nous arrêterons en particulier sur les oscillateurs en anneau, les PLL et les blocs de synthèse de fréquence DFS disponibles sur les plate-formes Xilinx. En effet ces composants représentent la quasi totalité des sources d'aléa basés sur le jitter.

Nous présenterons des comparaisons qualitatives afin de pouvoir par la suite modéliser et simuler un TRNG complet de la source d'aléa jusqu'à la suite numérique aléatoire.

2 Notions générales sur le jitter

Le terme jitter, est un terme technique générique utilisé pour désigner l'instabilité de la période dans le temps d'un oscillateur. Il existe une grande variété de manières de quantifier cette instabilité (*phase jitter*, *jitter RMS*, *cycle to cycle jitter*, etc...), tant dans le domaine temporel que dans le domaine fréquentiel. Les principales seront évoquées dans ce chapitre. Les problèmes liés au jitter ont un aspect particulièrement important et gênant dans les systèmes de communication à très haut débit car à cause de l'instabilité du signal d'horloge dans le temps des erreurs de transmission peuvent se produire. D'où les efforts considérables pour le modéliser et le minimiser dans les circuits intégrés, efforts qui vont à l'encontre des possibilités de développement de générateurs embarqués dans les circuits entièrement numériques.

2.1 Définitions et formalisme mathématique

La définition la plus utilisée est celle donnée par l'ITU (*International Telecomunication Union*) qui définit le jitter comme la déviation du front d'un signal d'horloge pour ce qui nous intéresse, de sa position idéale. Ainsi un signal d'horloge idéal de période T_0 aura un front montant idéalement espacé dans le temps, aux instants $t = nT_0$. A cause du bruit, un signal réel présentera cependant une durée variable entre deux fronts montants successifs. Certaines fois ce front montant sera en avance, certaines fois il sera en retard par rapport au front idéal.

La caractérisation du jitter peut être abordée tant sur un plan temporel que sur un plan fréquentiel, on parle dans ce dernier cas de *bruit de phase*.

2.1.1 Analyse temporelle du jitter

Phase jitter* ou *absolute jitter Le *phase jitter* est une mesure discrète de l'avance ou de retard de phase des fronts montants et descendants par rapport à leur positions idéales. Si nous considérons un signal idéal avec une période idéale T_0 alors les fronts montants de celui-ci auront lieu tous les nT_0 exactement. Le signal jitté au contraire va être soit en avance soit en retard par rapport au temps idéal et aura son front montant

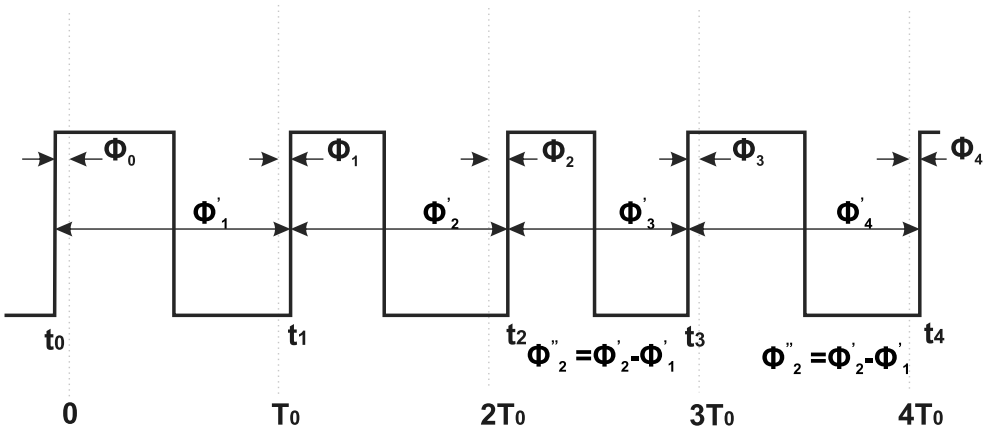

Figure 2.1 Définitions pour les différents types de jitter - *period jitter*, *cycle-to-cycle jitter* et *phase jitter*

aux instants :

$$t_n = nT_0 + \Phi_n \tag{2.1}$$

Ainsi le *phase jitter* est mathématiquement défini par l'équation, (Figure 2.1)

$$\Phi_n = t_n - nT_0 \tag{2.2}$$

Selon le type d'oscillateur, cette mesure du jitter peut être divergente si l'oscillateur est en oscillation libre ce qui rend sa quantification difficile voire inappropriée dans le cas d'un oscillateur libre.

Period jitter Le *period jitter* est également une mesure discrète. Elle est particulièrement importante et elle sera largement utilisée dans le cadre de cette thèse car c'est cette grandeur qui est la plus facilement mesurable en externe au FPGA grâce à l'oscilloscope numérique. On définit le *period jitter* comme la mesure successive dans le temps des périodes réelles du signal [Incb]. Elle peut être mathématiquement représentée par l'équation 2.3

$$\Phi'_n = t_{n+1} - t_n \tag{2.3}$$

Pour plus de cohésion mathématique, certains auteurs [Wed06] la présentent comme la différence entre la période réelle observée et la période idéale (2.4). Cela reviens à ajouter une constante T_0 dans l'équation 2.3. Cette définition est très pratique d'un point de vue mathématique, car elle permet de rattacher la mesure du période jitter à autres mesures du jitter et ne change rien aux observations réalisés dans la pratique à une constante près. Dans la pratique la période idéale est inconnue et peut seulement être estimée.

$$\Phi'_n = t_n - t_{n-1} - T_0 \tag{2.4}$$

Nous pouvons maintenant relier le *period jitter* au *phase jitter* en écrivant :

$$\Phi'_n = \Phi_n - \Phi_{n-1} \tag{2.5}$$

Cycle-to-Cycle jitter Une autre mesure du jitter est définie comme la différence entre deux périodes successives, comme le montre l'équation 2.6

$$\Phi_n^{''} = \Phi_n^{'} - \Phi_{n-1}^{'} \tag{2.6}$$

Comme on peut le remarquer ces différentes mesures du jitter sont reliés et expriment différents aspects du jitter. En effet le *period jitter* peut être interprété comme la vitesse de variation du jitter tandis que le *cycle-to-cycle* jitter comme l'accélération de celui-ci.

2.1.2 Analyse fréquentielle du jitter

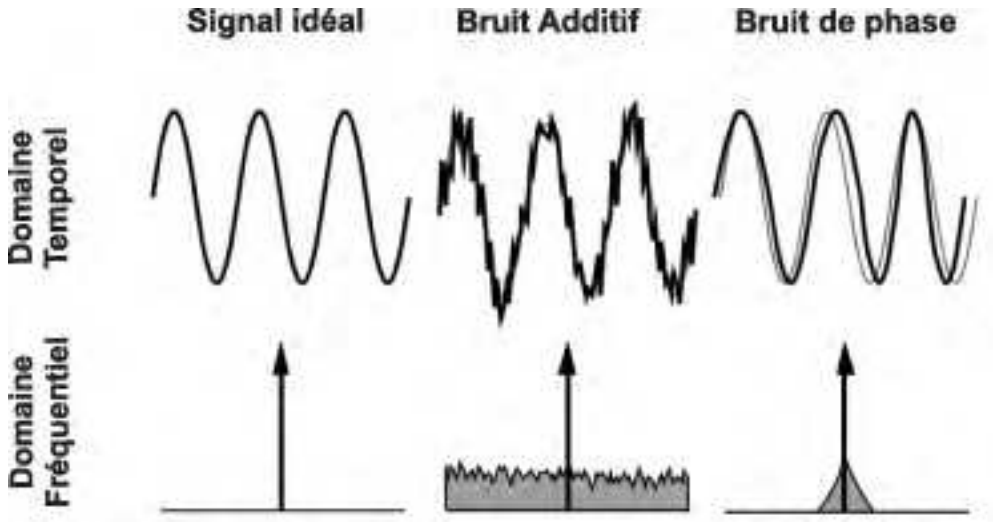

Figure 2.2 Illustration de l'effet du bruit sur le signal d'un oscillateur dans le domaine temporel et dans le domaine fréquentiel ; signal idéal (à gauche), bruit additif (au centre), bruit de phase (à droite)

Le jitter peut être également observé dans le domaine fréquentiel (Figure 2.2). On parle de bruit de phase ou de *phase noise* en anglais. En effet un signal idéal se traduirait dans le domaine fréquentiel par une raie unique si ce signal était sinusoïdal, ou une infinité de raies d'amplitude espacées de la fréquence fondamentale dans le cas d'un signal carré. Le jitter présent dans le signal se traduira par une modulation aléatoire de la fréquence entraînant une dispersion de la fréquence fondamentale. Le bruit de phase est mesuré en db à une fréquence d'offset de la fréquence fondamentale, donnant ainsi une métrique en dbc (*db relative to the carrer*). La métrique finale ainsi obtenue est le dbc/Hz.

Si le jitter est totalement aléatoire, la déviation de la fréquence fondamentale est caractérisée par une pente qui décroît avec l'écart de la fréquence centrale (Figure 2.3). Cette pente est caractéristique du type de bruit aléatoire : bruit blanc ou bruit 1/F.

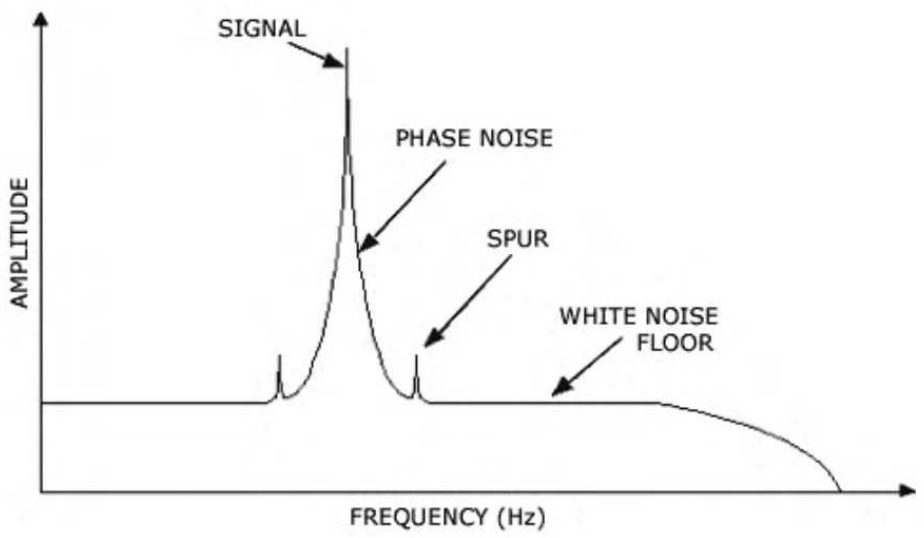

Figure 2.3 Représentation du bruit de phase dans le domaine fréquentiel

Cette représentation fréquentielle est intéressante car si un phénomène déterministe et périodique est présent dans le jitter il apparaîtrait comme un pic près de la fréquence fondamentale.

2.2 Causes, origines, types de jitter

Le jitter est un phénomène complexe dont les causes sont multiples. Ainsi on peut séparer le jitter en deux composantes principales : le jitter aléatoire (RJ *Random Jitter*) et le jitter déterministe (DJ *Deterministic Jitter*). Ce dernier peut à son tour se décomposer en plusieurs sous catégories selon les différents phénomènes qui lui donnent naissance. Cette séparation du jitter est illustrée à la Figure 2.4.

2.2.1 Types de jitter

Le jitter total observé dans un signal est souvent un mélange de jitter déterministe et de jitter aléatoire comme cela est montré à la Figure 2.4.

Partie aléatoire du jitter Le jitter est en partie crée par le bruit intrinsèque à tout composant électronique. Ce bruit étant de nature aléatoire, le jitter est également en grande partie de nature aléatoire. Le bruit est un phénomène normal dont on ne peut pas s'affranchir en électronique. Il peut être du à plusieurs causes et il existe plusieurs types de bruit dans les composants électroniques avec chacun des caractéristiques fréquentielles différentes. Le bruit thermique (*Johnson*) du au mouvement ou agitation aléatoire des électrons. Le bruit de grenailles (*Shot noise*) du aux fluctuations induites par les porteurs élémentaires de charges. Ces deux types de bruit sont intrinsèques à

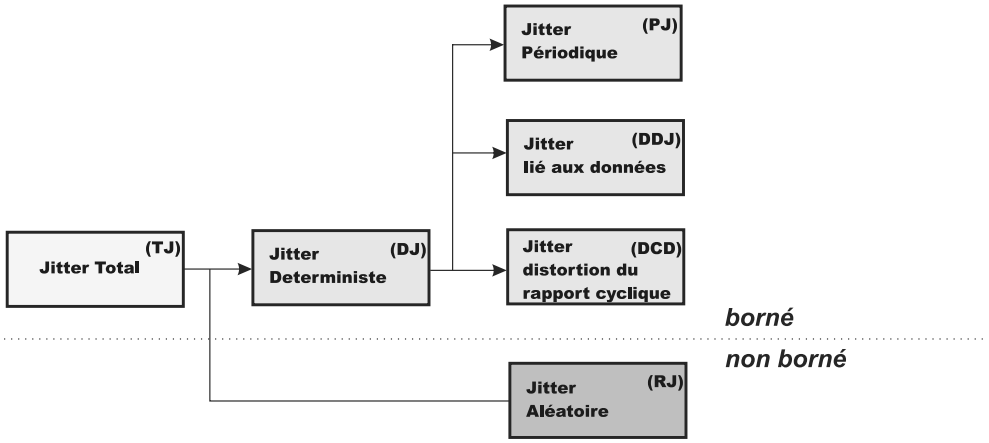

Figure 2.4 Types de jitter

tous les composants électroniques et on ne peut pas s'en affranchir. Leur caractéristique fréquentielle est généralement assimilée à du bruit blanc, dont l'amplitude est constante pour toutes les fréquences et dont la distribution statistique suit une loi *normale* ou loi Gaussienne.

Un autre type de bruit, très important en électronique est le bruit *flicker* ou *bruit 1/F*. Il se caractérise par son spectre de fréquences en 1/F ce qui se traduit par de grandes amplitudes aux fréquences basses. Son origine est peu connue et on pense qu'il résulte de l'accumulation d'un grand nombre de causes différentes , liées au facteur qualité du système électronique dans son ensemble. Son amplitude peut donc être réduite en améliorant la qualité des processus de fabrication des composants électroniques.

Partie déterministe du jitter Cependant d'autres perturbations déterministes (modulations, diaphonie, etc...) peuvent venir s'ajouter au jitter. Ainsi le jitter total observé sera dans la réalité un mélange de jitter aléatoire et de jitter déterministe. Le jitter déterministe peut être périodique ou bien corrélé aux données transmises. Dans tous les cas son amplitude est bornée contrairement au jitter purement aléatoire dont l'amplitude n'est pas bornée.

2.2.2 Quantification du jitter

A cause de la nature aléatoire et non déterministe du jitter, pour quantifier celui-ci, on utilise des grandeurs statistiques tels que les valeurs moyennes ou l'écart type de la population de mesure d'une des grandeurs définies dans la Section 2 de ce chapitre. Ces estimateurs sont obtenus en collectant un grand nombre de mesures et sont généralement extraites à partir des histogrammes de ces populations de mesures. Ces valeurs sont très bien adaptées pour caractériser le jitter aléatoire car leurs estimateurs sont très rapidement convergents. Par contre lorsque le jitter

aléatoire est mélangé à du jitter déterministe leur utilisation s'avère trompeuse. En effet dans [PL04] sont présentés plusieurs histogrammes différents contenant du jitter aléatoire et du jitter déterministe avec les mêmes moyennes et les même écarts types.

Les grandeurs les plus souvent utilisées pour la quantification du jitter sont l'écart-type (σ) et la valeur crête-à-crête (*pk-pk*) d'une population de mesures du jitter (cela peut être n'importe lequel des différents types de jitters decrits auparavent). L'écart type (σ) est particulièrement bien adapté pour des populations de mesures qui suivent une loi de distribution Gaussienne, en effet dans ce cas, et dans ce cas seulement, l'écart-type et la moyenne de la population estimés expérimentalement à partir des données caractérisent complètement le comportement du jitter. Il s'agit de la métrique RMS du jitter (*Root Mean Square*), par abus de langage, car l'écart-type est calculé selon la formule :

$$\sigma = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}\left(X_i - \bar{X}\right)^2} \tag{2.7}$$

avec

$$\bar{X} = \frac{1}{n}\sum_{i=1}^{n} X_i \tag{2.8}$$

Cette métrique RMS n'a de sens que si la loi de distribution du jitter suit rigoureusement une loi Gaussienne, ce qui dans la réalité n'est que rarement le cas. Dans tous les autres cas il s'agit uniquement d'une mesure de l'écart type d'une population de mesures qui ne renseigne pas directement sur la loi de distribution suivie, mais qui est néanmoins porteuse d'information. Dans ce cas une représentation de la population de mesures dans le temps ainsi qu'un histogramme sont nécessaires.

La grandeur crête-à-crête *pk-pk* (*peak to peak*) quant à elle est très utilisée pour quantifier la partie déterministe du jitter qui est par nature bornée. Il s'agit de la plus grande différence entre deux valeurs de la population de mesures expérimentales. Cependant l'utilisation de cette grandeur doit toujours s'accompagner de la taille de population de mesures car la partie aléatoire (qui n'est pas bornée) contenue dans celle-ci provoquera une augmentation de la valeur *pk-pk* avec la taille de la population. Cette valeur présente l'avantage de fournir un nombre facilement manipulable, par contre cette métrique présente l'inconvénient de ne pas caractériser complètement le comportement du jitter. Ainsi deux populations de mesures du jitter avec des lois de distribution différentes peuvent présenter la même valeur *pk-pk* [PL04].

Ainsi pour résumer, à nos yeux les deux grandeurs σ et *pk-pk* ne sont pas suffisants pour quantifier le jitter. La valeur *pk-pk* est fortement dépendante de la taille de la population de mesures tandis que la grandeur σ est rapidement convergente vers une valeur stable indépendamment de la taille de la population, cependant elle n'est pas suffisante pour caractériser complètement le comportement du jitter dans la réalité. C'est pourquoi nous avons opté pour la suite de cette thèse pour une représentation

graphique du jitter donnant les moyennes et les écarts-types du *period jitter* et du *Cycle to Cycle jitter* mais aussi les périodes réelles successives du signal étudié dans le temps ainsi qu'une FFT du signal échantillonné à 40GS obtenue directement sur l'oscilloscope Lecroy WavePro 7300. De cette façon différents comportements du jitter peuvent être comparés entre eux.

3 Description des méthodes de mesure

Quantifier précisément le jitter dans un système électronique n'est pas une tâche aisée. En effet le jitter est une manifestation de l'intégrité du signal et donc la dégradation du signal à mesurer à n'importe quel niveau (soudures, vias, PCB) peut affecter sa mesure.

Dans le cadre de cette thèse, une difficulté supplémentaire se présente quant à la mesure du jitter. En effet nous voulons utiliser ce dernier comme source d'aléa et nous avons besoin de connaître ses caractéristiques afin de modéliser et prédire le comportement du générateur. Cependant pour des raisons de sécurité les signaux jittés utilisés comme source d'aléa sont internes au FPGA. Nous devons donc être capable de caractériser un signal interne dans le FPGA. Une solution consisterai à sortir le signal sur une broche et d'effectuer une mesure en externe avec un oscilloscope à large bande passante. Cependant cette méthode présente l'inconvénient de modifier l'intégrité du signal réel interne et donc potentiellement de faire une erreur sur la quantification et la caractérisation de la source d'aléa.

Réaliser une mesure en interne d'un signal jitté n'est pas aisée. En effet la mesure en externe à l'aide de l'oscilloscope présente l'avantage de pouvoir mesurer précisément chaque période du signal en l'échantillonnant à une très grande fréquence de plusieurs GHz. Il est impossible d'embarquer des méthodes de mesures du jitter avec de tels performances en interne, car souvent les signaux qui nous intéressent sont les signaux les plus rapides disponibles dans le FPGA. Tout au mieux en interne nous pouvons estimer le jitter en observant son effet accumulé. Le prix à payer est toutefois très élevé car l'information précieuse de son comportement période par période est perdue.

La caractérisation du jitter comme source d'aléa présente donc un dilemme. Soit la mesure est effectuée en interne et le signal est identique à celui effectivement utilisé pour la génération d'aléa, mais dans ce cas nous ne disposons pas des informations du jitter période par période mais d'une estimation de son effet accumulé dans le temps. Soit la mesure est effectuée en externe, chaque période est précisément mesurée car échantillonnée à très haute fréquence, mais l'intégrité du signal est modifiée par rapport au signal réel interne et une erreur peut être commise sur l'amplitude du jitter. Dans le cadre de cette thèse, nous utiliserons la caractérisation externe principalement à des fins de comparaison.

En ce qui concerne la mesure en externe du jitter, plusieurs facteurs peuvent perturber l'intégrité du signal et fausser la mesure. Ainsi la qualité de la carte support (capacités de découplage, soudures, plans de masse, longueur des pistes...) peut directement dégrader le signal. Egalement le type de buffer de sortie au sein du FPGA peut modifier l'intégrité du signal interne. En effet il existe une grande variété de standards électroniques de transmission de l'information numérique (TTL, LVTTL, CMOS, LVCMOS, LVDS, etc...) et chaque standard possède ses propres caractéristiques (tensions, bande passante maximale, fan out, etc...). Ces standards peuvent être séparés en deux classes, les standards différentiels (*differential*) et les standards non différentiels (*single ended*). Les standards différentiels présentent justement l'avantage d'être insensibles aux perturbations externes. [Ma] Pour illustrer la sensibilité de la mesure du jitter en

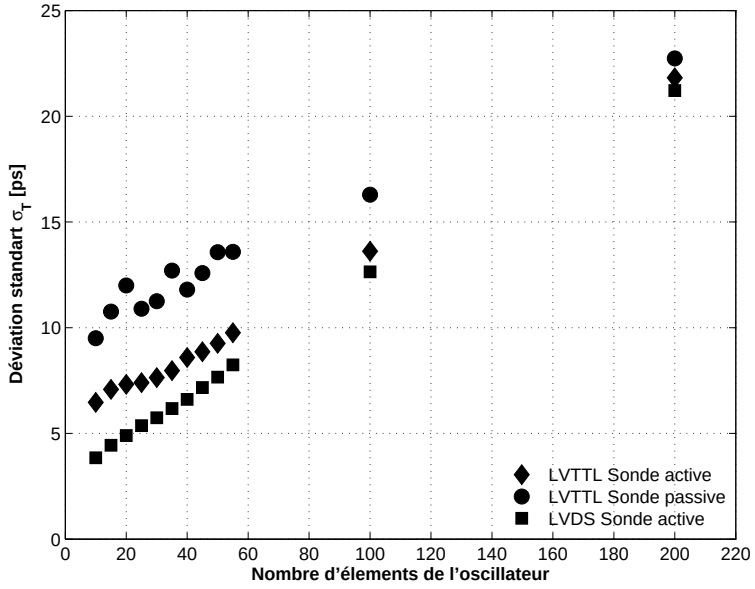

Figure 2.5 Impact des sondes et du standard de sortie électronique sur la mesure du jitter en externe. Carte d'évaluation Digilent Spartan 3

externe aux conditions de mesures, nous avons effectué la mesure du jitter dans plusieurs configurations. Un signal périodique de période variable a été obtenu dans un oscillateur en anneau en faisant varier le nombre d'éléments à retard. Le jitter a été évalué sur des signaux transmis à l'aide de différents standards de transmission électroniques LVTTL (*single ended*) et LVDS (*differentiel*). De plus l'acquisition a été effectuée avec deux types de sondes différentes : une sonde passive classique d'une bande passante de 500Mhz, et une sonde différentielle d'une bande passante de 4Ghz. Les résultats sont présentés à la Figure 2.5. On peut constater que pour toutes les fréquences observées l'erreur relative commise est importante. Celle-ci est d'autant plus importante que la fréquence du signal observé est grande (nombre d'éléments à retard faible). Le jitter semble être plus faible pour la mesure avec la sonde différentielle et avec le standard de transmission différentiel.

Ainsi, si les conditions de mesure en externe ne sont pas précisées il faut être très prudent sur la quantification du jitter qui peut être proposée dans certains papiers qui utilisent le jitter comme source d'aléa. Dans le cadre de cette thèse, nous nous limiterons exclusivement à la méthode de mesure externe avec une sonde différentielle et le standard de transmission différentiel LVDS. A nos yeux en effet vu les bandes passantes du standard LVDS et de la sonde différentielle il serait difficile d'imaginer que celle-ci filtre le bruit. Par contre il est plus plausible qu'une sonde passive et un standard de transmission lent introduisent du bruit. De plus la bande passante du standard LVDS est bien supérieure à celle des standards non différentiels. Ainsi de signaux de plusieurs centaines de Mhz ont pu être observés sans une dégradation minimale du signal ce qui n'est pas le cas avec les standards non differentiels LVCMOS et LVTTL.

Il est important de noter que jusqu'à présent tous les papiers de la communauté scientifique qui utilisent le jitter comme source d'aléa et qui traitent de la quantification du jitter en externe ([SMS07], [Abc04], [Cop05], [SPV06],...) ne précisent pas le type de standard de transmission et le type de sonde utilisée.

4 Jitter dans les oscillateurs en anneau

Le jitter présent dans les oscillateurs en anneau (*RO Ring Oscillators*) est de loin la source d'aléa la plus utilisée pour la réalisation de TRNG. La raison en est simple, les oscillateurs en anneau sont des oscillateurs basés sur des portes numériques facilement implantables dans toutes les familles de FPGA. La fréquence de fonctionnement de l'oscillateur peut être facilement réglée dans une large plage en modifiant le nombre d'éléments contenus dans l'oscillateur.

Pour modéliser les TRNG qui utilisent le jitter des oscillateurs en anneau comme source d'aléa, une quantification de son amplitude ainsi qu'une caractérisation poussée sera présentée dans cette section. Cependant nous présenterons d'abord le fonctionnement d'un tel oscillateur. Une modélisation du fonctionnement d'un tel oscillateur et du profil de jitter présent dans le signal à sa sortie sera présentée en détail dans le chapitre 4.

Les oscillateurs en anneaux sont omniprésents en conception de circuits intégrés. Une des raisons est que contrairement aux oscillateurs LC ou RC, ils peuvent facilement être intégrés dans des technologies de fabrication standard et ne demandent pas l'utilisation de substrats ou de matériaux exotiques (GaN, saphire, ...) à leur intégration sur silicium. Il existe un grand nombre de publications qui traitent du jitter dans les oscillateurs à anneaux, cependant la quasi totalité traite des oscillateurs en anneau réalisés dans les circuits ASIC et très peu traitent du phénomène du jitter dans les FPGA. Or bien que l'on puisse réaliser un oscillateur en anneau dans ces deux technologies, les outils et les modèles présentent très peu de similitudes [MS].

En effet dans un circuit intégré spécialisé, les oscillateurs sont généralement réalisés par le bouclage d'un nombre impair d'inverseur CMOS constitués uniquement de deux transistors MOS complémentaires. La simplicité et le fonctionnement de ces inverseurs sont le fondement de la micro-électronique et leur fonctionnement est relativement bien compris et modélisé. L'interconnexion est réalisée grâce à des pistes métallisés qui ne sont pas actives et dont les caractéristiques de bruits sont également bien comprises. Au contraire dans un FPGA, les éléments qui constituent l'oscillateur peuvent être de nature différente (LUT, portes AND, inverseurs, etc...) dont la structure micro-électronique est plus complexe que celle d'un inverseur CMOS et généralement complètement ignoré par le concepteur qui travaille à un niveau d'abstraction bien supérieur. De plus l'interconnexion dans un FPGA se fait non seulement à l'aide de pistes métalliques passives mais à l'aide d'éléments de commutation actifs *switch boxes*.

Ces différences font que les modèles proposés dans la littérature et qui sont destinés aux oscillateurs en anneau dans des technologies ASICs sont peu, voir pas du tout utilisables par les concepteurs FPGA travaillant au niveau au dessus. C'est pourquoi nous proposerons dans le chapitre 4 un modèle de haut niveau de comportement d'un oscillateur en anneau.

4.1 Fonctionnement d'un oscillateur en anneau

Le fonctionnement d'un oscillateur en anneau est présenté à la Figure 2.6. Deux cas y sont présentés, un oscillateur libre et un oscillateur commandé. Le signal EN servant à autoriser ou interdire les oscillations. L'oscillateur est composé d'un nombre de portes logiques en boucle ou en anneau. En effet chaque porte logique a un temps de propagation du signal entre l'entrée et la sortie intrinsèque (τ sur la Figure) et qui dépend de la technologie de réalisation. Traditionnellement pour la réalisation d'un oscillateur en anneau un nombre impair d'inverseur était utilisé. Un front (transition entre deux états logiques) se propage ainsi en boucle et si le nombre d'inverseurs est impair alors une oscillation se produit. La fréquence F de cette oscillation peut être exprimée par l'équation 2.9, où τ est le temps de propagation élémentaire d'une porte logique supposé ici identique et constant pour simplifier l'explication.

$$F = \frac{1}{2 \times \sum_{n=1}^{i} \tau_n} \tag{2.9}$$

Si le nombre d'éléments constituants est impair alors le pas des périodes que l'on peut obtenir est de 4τ seulement. Pour affiner la gamme des fréquences que l'on peut obtenir avec un tel oscillateur nous avons utilisé la structure commandée représentée à la Figure 2.6 à droite. Avec un seul élément inverseur - la porte NAND lorsque le signal EN est à l'état haut, et un nombre quelconque de portes non inverseuses pour ajuster la fréquence souhaitée. De cette manière l'intervalle entre deux périodes possibles est diminué à 2τ donnant une plus grande liberté au concepteur de TRNG ainsi que la possibilité d'arrêter l'oscillateur à tout moment.

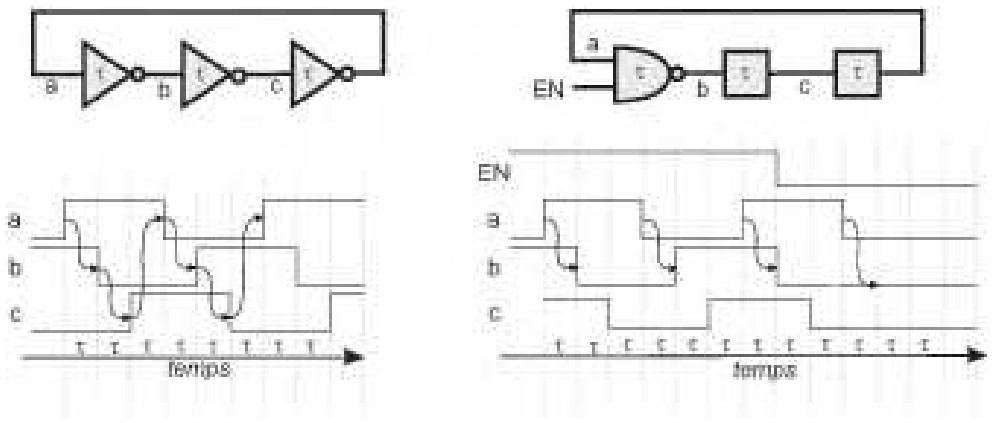

Figure 2.6 Chronogrammes de fonctionnement d'un oscillateur en anneau libre et commandé

4.2 Implantation dans les FPGA

L'implantation dans un FPGA d'un oscillateur en anneau ne pose pas de problèmes particuliers, et le concepteur dispose d'un large choix de possibilités pour les éléments à retards. Les différents constructeurs de FPGA proposent chacun une terminologie différente mais les architectures sont semblables, on peut y reconnaître des blocs contenant de la logique programmable (CLB, LAB, etc...) et des blocs d'interconnexions également programmables. De plus, il peut exister plusieurs types de lignes de transmissions entre les blocs d'interconnexion. Le contenu d'un bloc logique programmable diffère selon les fabricant et le type du FPGA mais on peut y reconnaître systématiquement une unité logique programmable indexée (LUT *Look-Up Table*) à plusieurs entrées et une sortie et une ou plusieurs bascules programmables.

Chaque porte logique, quelle qu'elle soit, présentera inévitablement un temps de propagation faisant ainsi un candidat idéal pour un élément à retard pour l'oscillateur en anneau. De plus la possibilité de disposer de plusieurs types d'interconnexions donne une souplesse supplémentaire pour la conception de l'oscillateur.

Dans le cadre de cette thèse, nous avons utilisé pour toutes les expériences, aussi bien pour la caractérisation que pour l'implantation de TRNG, les mêmes types d'éléments à retard. Ces différents éléments à retard sont présentés dans le Tableau 2.1. Ce choix trouve sa justification dans le fait que nous avons essayé d'implanter des TRNG et de caractériser les sources d'aléa dans plusieurs familles de FPGA différentes. Nous voulions donc avoir une architecture VHDL et matérielle aussi portable que possible.

Pendant la phase de caractérisation d'oscillateurs en anneau comme source d'aléa, nous avons remarqué l'importance du type d'interconnexion entre les différents éléments à retard. Chaque type d'interconnexion présente des temps de propagation différents. Pour évaluer son impact, nous avons implanté un oscillateur en anneaux avec un

Famille FPGA	Type de délai	temps de propagation	Datasheet
Xilinx Spartan 3	lut1	610 ps (Max value)	[Xil]
Actel Fusion 600	and2	630 ps (Standard value @70°)	[Acta]
Actel Igloo	and2	840 ps (Standard value @70°)	[Actb]
Altera Cyclone III	lcell		

Tableau 2.1 Synthèse et caractéristiques théoriques des différents types de portes logiques utilisés comme éléments à retard dans le cadre de cette thèse

nombre fixe d'éléments à retard et nous avons fait varier le type d'interconnexion entre ces éléments à retard. La cible choisie était Xilinx Spartan3. Dans ce FPGA il existe quatre types d'interconnexions : *Direct Lines* (DL), *Double Lines* (DoL), *Hex Lines* (HL), *Long Lines* (LL). Pour chaque type d'interconnexion, nous avons mesuré la période moyenne obtenue et l'écart type du *period jitter*. Les résultats sont présentés à la Figure 2.7. On peut constater que l'interconnexion a une influence directe sur le période de l'oscillateur mais également sur le jitter. Cette observation n'est, à notre connaissance, pas reportée dans la littérature. Elle est importante car le jitter est utilisé comme source d'entropie et son amplitude doit être la plus grande possible.

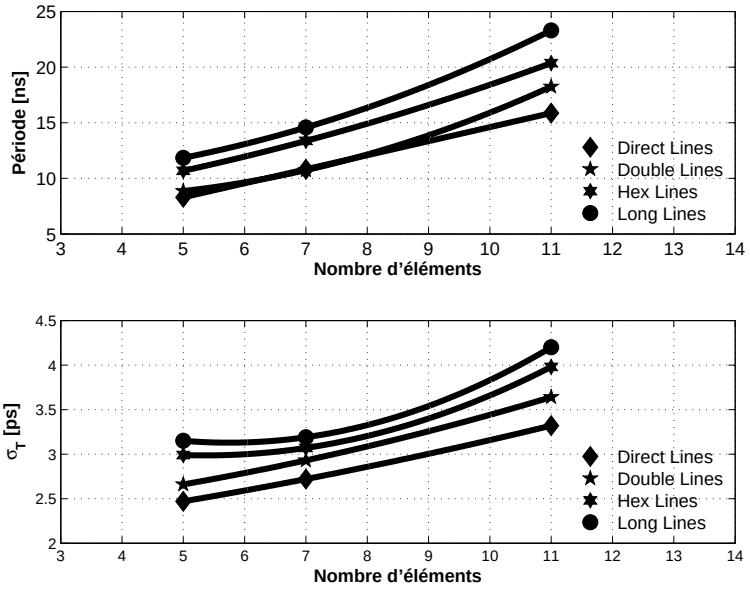

Figure 2.7 Impact du type d'interconnexion dans un oscillateur en anneau sur le jitter et la période d'oscillation

4.3 Susceptibilité aux conditions externes

Les oscillateurs en anneau sont très sensibles aux conditions externes tels que la tension d'alimentation ou la température. Dans [LBRPB] Lopez *et al.* proposent

d'estimer la température de fonctionnement de la puce en observant la fréquence d'un oscillateur en anneau. Cette sensibilité accrue s'explique par le fait que les oscillateurs en anneau implantés dans les FPGA ne sont pas des circuits à part entière étudiés et réglés par les fabricants de FPGA mais plutôt un détournement des fonctionnalités du FPGA. De plus les oscillateurs en anneau généralement utilisés pour la génération d'aléa sont relativement simples. En effet des structures oscillantes plus complexes existent et sont généralement utilisées dans les ASIC. Pour la génération d'horloges rapides. Citons les oscillateurs en anneau couplés ou les structures différentielles qui présentent une immunité aux bruits induits par l'alimentation. Une description de ces oscillateurs et de leur applications est donnée dans [MS]. Cependant il n'existe pas à notre connaissance de tels structures implantées dans les FPGA.

4.3.1 Effets des variations de l'alimentation

Nous avons évalué la dépendance de la période d'un oscillateur en anneau en fonction de la tension d'alimentation. Trois FPGA de trois familles ont ainsi été étudiés. Nous avons fait varier la tension du coeur du FPGA dans une plage allant de 0,95V à 1,6V. La tension nominale étant de 1,2V. Les résultats pour les 3 FPGA sont présentés aux Figures 2.8, 2.9 et 2.10. Nous pouvons noter que la dépendance n'est pas linéaire pour une large plage de variations. Cependant pour des variations suffisamment faibles autour du point de fonctionnement nominal de 1.2V, nous avons relevé les dépendances linéarisées pour comparer les 3 FPGA. Les résultats sont synthétisés dans le Tableau 2.2. Comme on peut le constater les dépendances autour du point de fonctionnement nominal sont pratiquement identiques pour les 3 familles de FPGA. Ces

FPGA	Delai nominal	Sensibilité du délai
Xilinx Spartan 3	624 ps	52.89 ps/V/élément
Actel Fusion 600	687 ps	900 ps/V/élément
Altera Cyclone III	229 ps	259 ps/V/élément

Tableau 2.2 Synthèse des sensibilités vis-à-vis de la tension d'alimentation des différentes familles de FPGA

dépendances sont conséquentes, car ils impliquent une dépendance directe du jitter aux bruits présents dans l'alimentation. Ainsi si l'alimentation présente des composantes déterministes répétitives, comme c'est le cas d'une alimentation à découpage par exemple (même de quelques millivolts) ils seront directement répercutés sur le jitter de l'oscillateur en anneau.

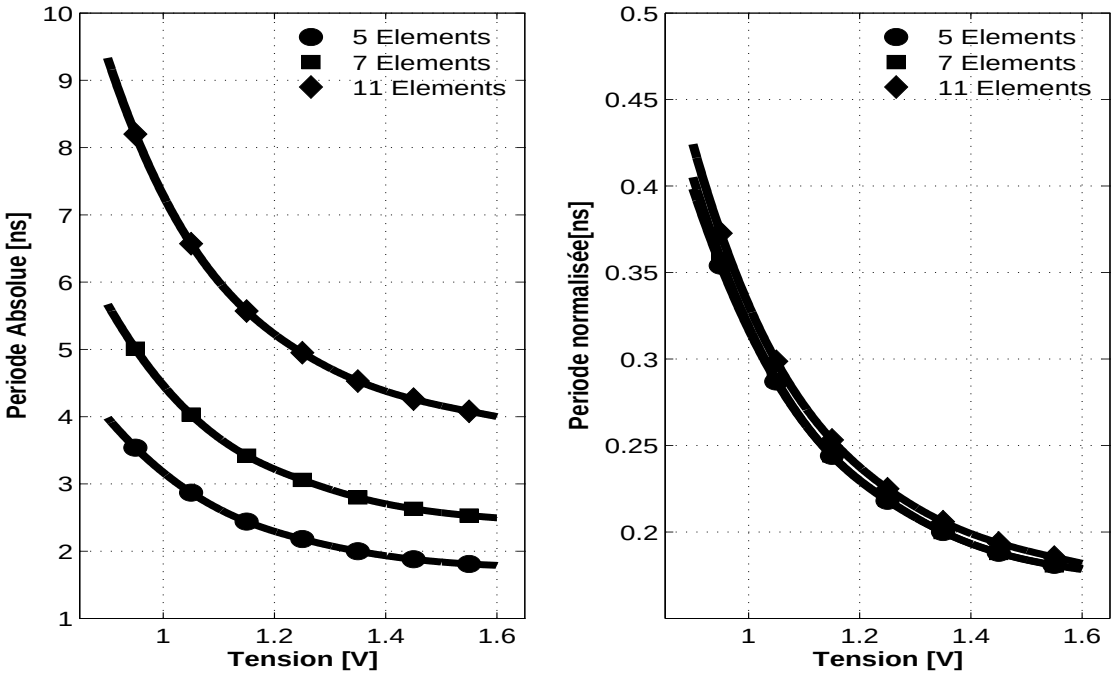

Figure 2.8 Effets de la variation de l'alimentation du coeur du FPGA sur la cible Altera Cyclone III

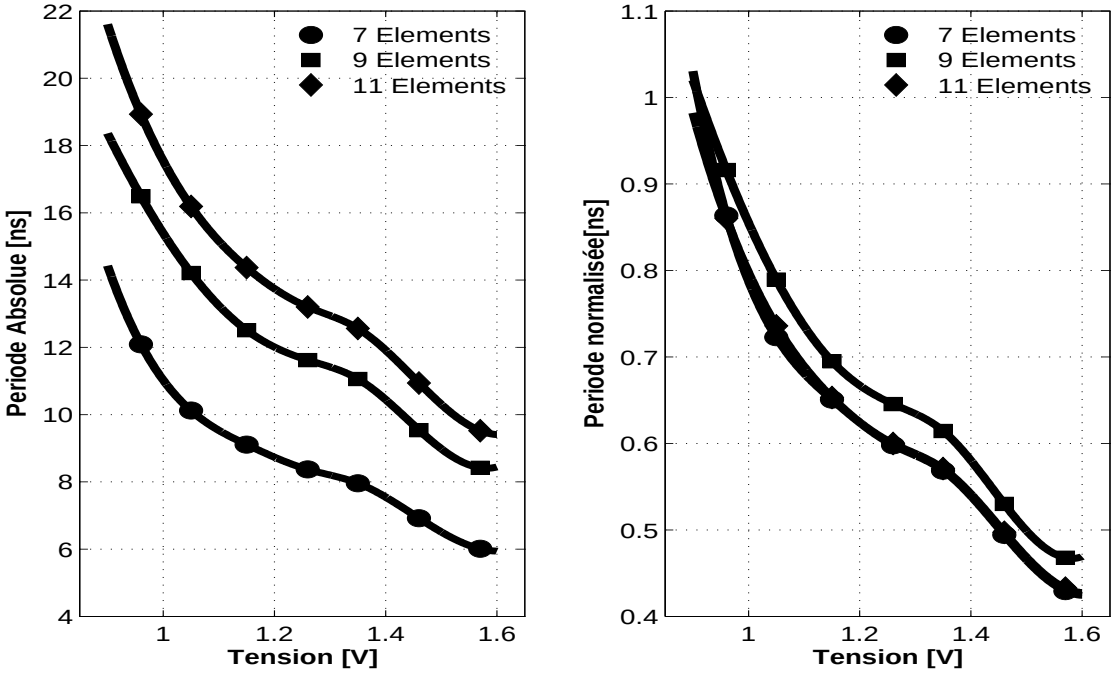

Figure 2.9 Effets de la variation de l'alimentation du coeur du FPGA sur la cible Xilinx Spartan 3e

Dans un contexte d'attaque matérielle, l'alimentation est particulièrement vulné-

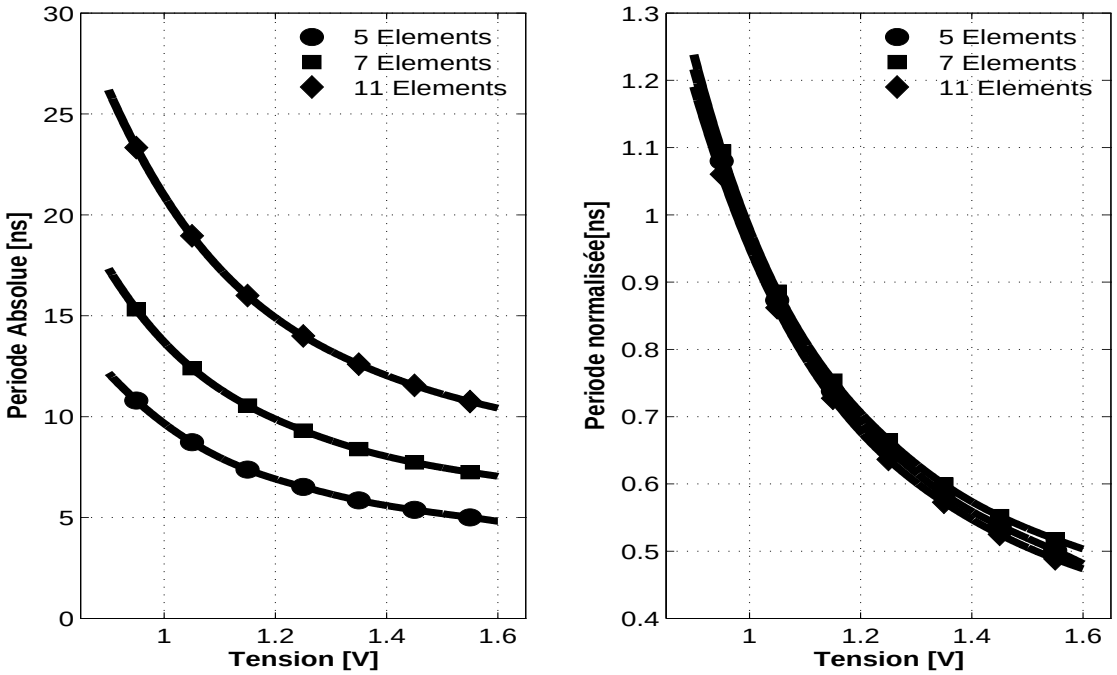

Figure 2.10 Effets de la variation de l'alimentation du coeur du FPGA sur la cible Actel Fusion 600

rable et souvent l'attaquant en a la maîtrise totale. Ainsi il faut être particulièrement vigilant à ces dépendances et bien évaluer leur impact sur la sécurité du TRNG.

Il faut noter également que ces perturbations, tout comme celles provoquées par la température affectent les oscillateurs en anneau d'une manière globale. Ainsi tous les oscillateurs seront sujets à ces perturbations en même temps.

4.3.2 Effets de la température

L'effet de la température sur la période de l'oscillateur en anneau a été étudié expérimentalement sur trois FPGA différents, un pour chaque famille. Nous avons fait varier la température de 30 °C à 60 °C par pas de 5 °C. Les résultats sont présentés sur la Figure 2.11 pour les trois familles, avec la même échelle pour les valeurs en ordonnée. Comme on peut le constater les trois familles présentent une sensibilité différente mais quasi linéaire. Ces Figures représentent la variation du délai moyen d'un élément à retard en fonction de la température. Ce délai est en fait constitué du délai de la porte et de l'interconnexion à la porte adjacente. Ainsi dans un oscillateur à 8 éléments il y a huit portes et huit interconnexions. Nous en tirerons également les valeurs des délais de ces éléments à température ambiante pour les trois familles, qui sont différentes.

Les courbes ont été approximées par une loi linéaire et la sensibilité en pico secondes par °C relative à un élément a été relevée et synthétisée dans le Tableau 2.3.

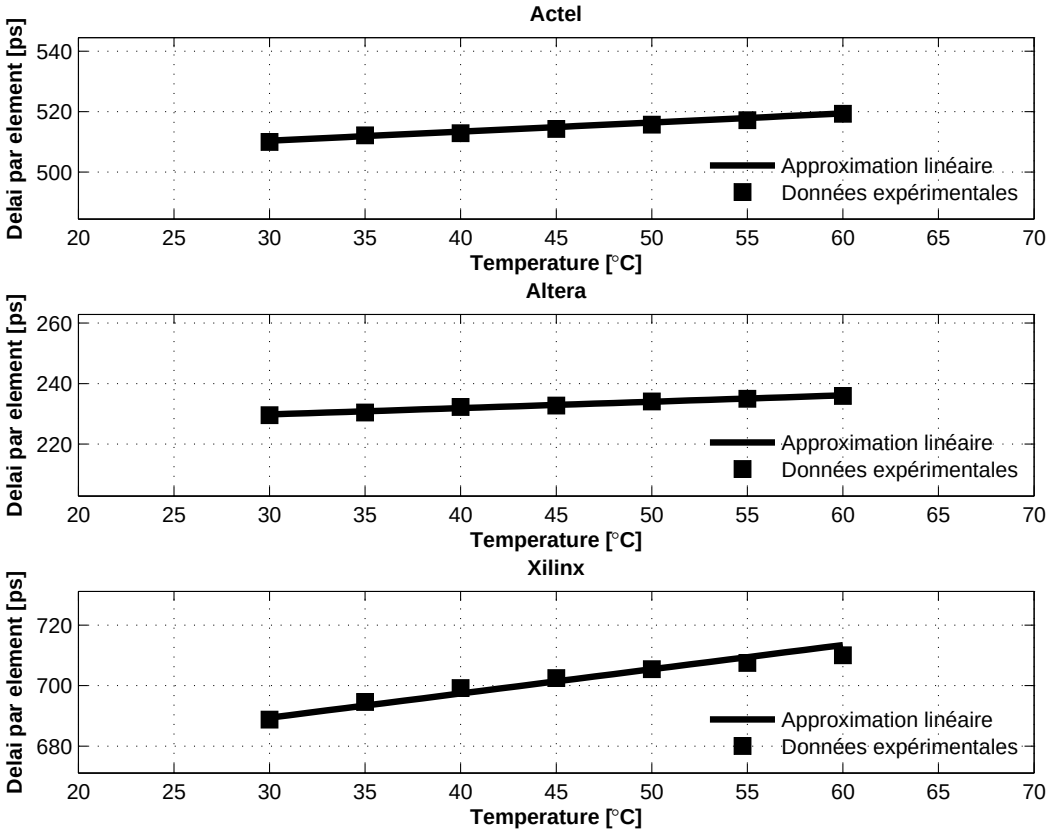

Figure 2.11 Impact de la variation de la température sur la période d'un oscillateur en anneau

Nous pouvons observer également que cette dépendance est sensiblement moins linéaire pour le FPGA Xilinx Spartan3 que pour les FPGA Cyclone III et Fusion. On peut

FPGA	Sensibilité du délai
Xilinx Spartan 3	0.3 ps/élément/°C
Actel Fusion 600	0.23 ps/élément/°C
Altera Cyclone III	0.83 ps/élément/°C

Tableau 2.3 Synthèse des sensibilités vis-à-vis de la température de fonctionnement des différentes familles de FPGA

constater que bien la température affecte sensiblement la période d'un oscillateur en anneau, son effet n'est pas aussi important que celui de la tension d'alimentation. De plus les variations de température peuvent être considérées comme quasi statiques comparées aux fréquences de fonctionnement des oscillateurs et du TRNG peuvent être de l'ordre du Mhz.

4.3.3 Effets de la logique environnante, fonctionnement en conditions non idéales

Dans le cadre de cette thèse, nous avons caractérisé les oscillateurs en anneau pendant un fonctionnement idéal, lorsque ces oscillateurs étaient implantés seuls dans tout le FPGA afin de quantifier la partie aléatoire du jitter. Cependant comme cela a été soulevé dans [San09], le fonctionnement de l'oscillateur et donc le jitter est fortement influencé par la logique environnante si celle-ci existe. En effet celle-ci va introduire des composantes déterministes très importantes. Le concepteur de TRNG doit être conscient de ce phénomène et le prendre en compte. Pour illustrer ce phénomène nous avons procédé à deux expériences. Nous avons étudié la dépendance de la période d'oscillation en fonction de la fréquence de fonctionnement d'une logique perturbante. Dans un second temps, nous avons analysé le comportement du jitter pendant un fonctionnement idéal et pendant le fonctionnement d'un chiffreur en même temps.

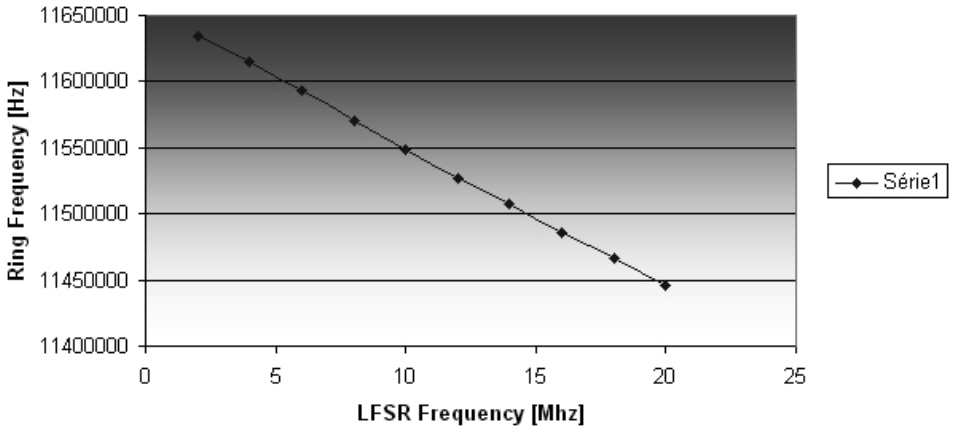

Figure 2.12 Fréquence d'oscillation d'un oscillateur en anneau en fonction de la fréquence de fonctionnement d'une logique perturbatrice environnante

Expérience 1 : Dépendance de la période de l'oscillateur en fonction de la fréquence d'une logique perturbatrice Nous avons observé l'extrême sensibilité de la fréquence d'un oscillateur en anneau à la fréquence de fonctionnement d'une logique environnante. En effet la fréquence de l'oscillateur dépend fortement de la fréquence de fonctionnement de la logique synchrone présente dans le FPGA dans un fonctionnement non idéal. Pour illustrer ce phénomène, nous avons observé la fréquence d'un oscillateur et nous avons implanté dans le FPGA des bascules que nous avons fait commuter à une fréquence croissante grâce à un signal externe. Pour avoir un grand nombre de bascules commutant en même temps, et simuler ainsi une forte activité logique, un registre à décalage à rétroaction linéaire de 2282 bits a été implanté. Le polynôme caractéristique était $x^{2281} + x^{715} + x^0$. Ce registre occupait ainsi 54 % des ressources logiques du FPGA. Les résultats sont présentés à la Figure 2.12 on peut

constater que la fréquence de l'oscillateur en anneau suit rigoureusement une loi linéaire de dépendance en fonction de la fréquence de la logique perturbatrice. Ce phénomène s'explique par le fait que la consommation est bien supérieure aux hautes fréquences qu'aux basses entraînant ainsi une chute de la tension d'alimentation.

Expérience 2 : Comportement d'un oscillateur en anneau avec un chiffreur AES perturbateur Nous avons effectué une seconde expérience plus proche des conditions de fonctionnement réels. Nous avons étudié l'influence sur le jitter du fonctionnement d'un bloc de chiffrement AES 128 bits cadencé à fréquence fixe de 50Mhz. Le texte en entrée était un compteur de 16 bits et l'occupation du chiffreur était de 16% des ressources logiques disponibles dans le FPGA. Les résultats de cette expérience fig.2.13 représentent l'allure du jitter pendant un fonctionnement idéal non perturbé par le chiffreur AES et l'allure du jitter perturbé pendant le fonctionnement d'AES. Comme on peut le voir par l'histogramme l'oscillateur est fortement perturbé dans le second cas, introduisant une forte composante déterministe de 0.2 ns crête-à-crête (sur une population de 27K mesures).

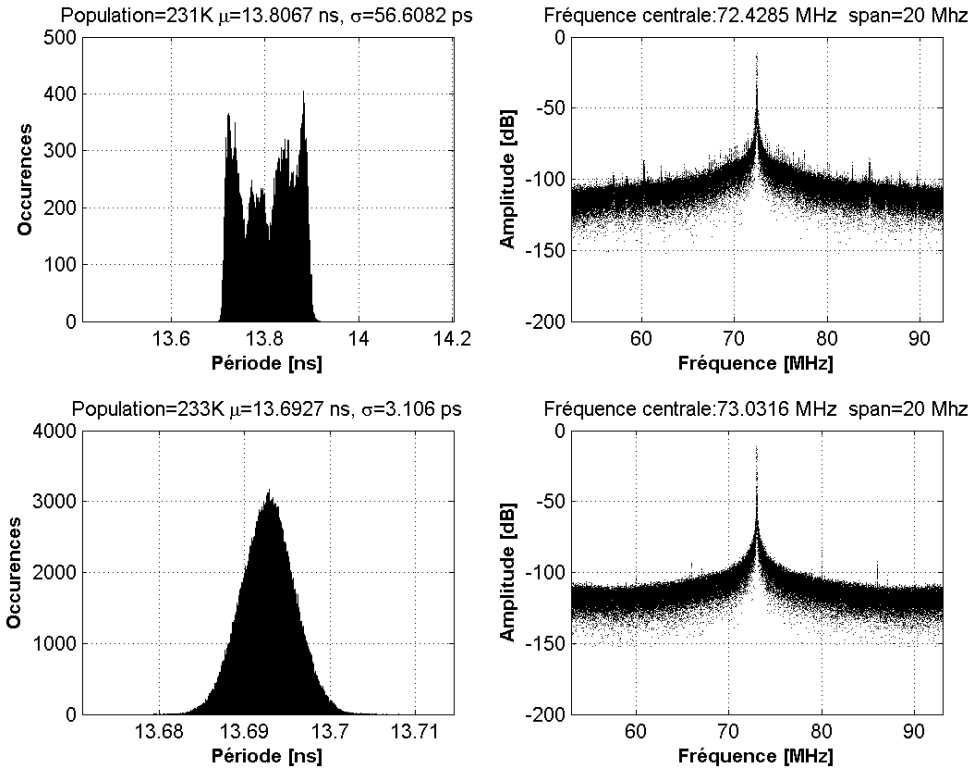

Figure 2.13 Perturbation du jitter pendant un fonctionnement non idéal simultanément avec un chiffrement AES à 50Mhz (en haut) et pendant un fonctionnement idéal (en bas), l'oscillateur fonctionnant seul dans le circuit

4.4 Phénomène de verrouillage des oscillateurs en anneau

Comme nous l'avons vu dans la Section 4.3.1 de ce chapitre, la période du signal issue d'un oscillateur en anneau est très sensible à la tension d'alimentation du FPGA. Ainsi un phénomène intéressant et potentiellement dangereux pour la génération d'aléa se produit lorsque deux oscillateurs en anneau de fréquences proches sont sujets à des variations de l'alimentation. Les deux oscillateurs possèdent le même nombre d'éléments mais ils ne sont pas totalement identiques. Ceci est du aux variations des paramètres physiques du circuit intégré lui même. Ainsi la variation de la tension d'alimentation aura un effet différent sur chaque oscillateur. L'effet de la variation de la tension peut être observé à la Figure 2.14 où nous avons les deux courbes correspondantes aux deux oscillateurs. On peut constater que leur valeurs se croisent pour une certaine valeur de la tension d'alimentation, impliquant une égalité des périodes.

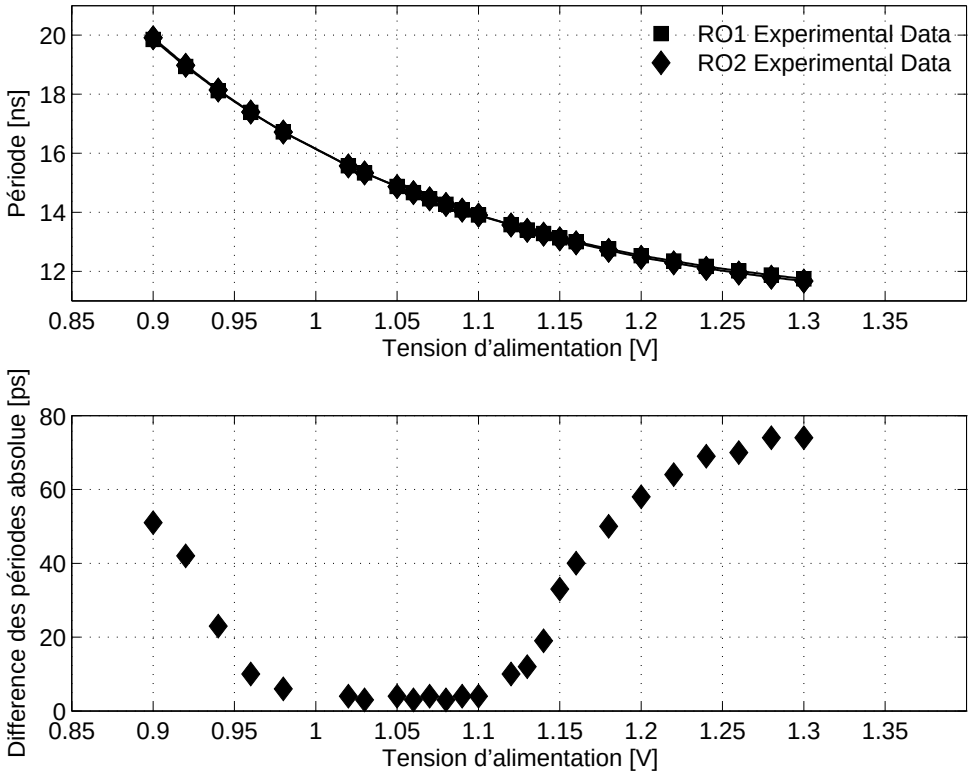

Figure 2.14 Evolution des périodes de deux oscillateurs en fonction de la tension d'alimentation du coeur (à gauche). Evolution de la différence des périodes en fonction de la tension - mise en évidence du phénomène de verrouillage

Pour mieux observer l'influence de la variation de la tension, nous avons également représenté à la Figure 2.14 la valeur absolue de la différence des deux périodes. Nous constatons un phénomène intéressant. En effet la différence reste très faible (pratiquement égale à zéro à l'erreur de mesure de l'appareil près) pendant une large

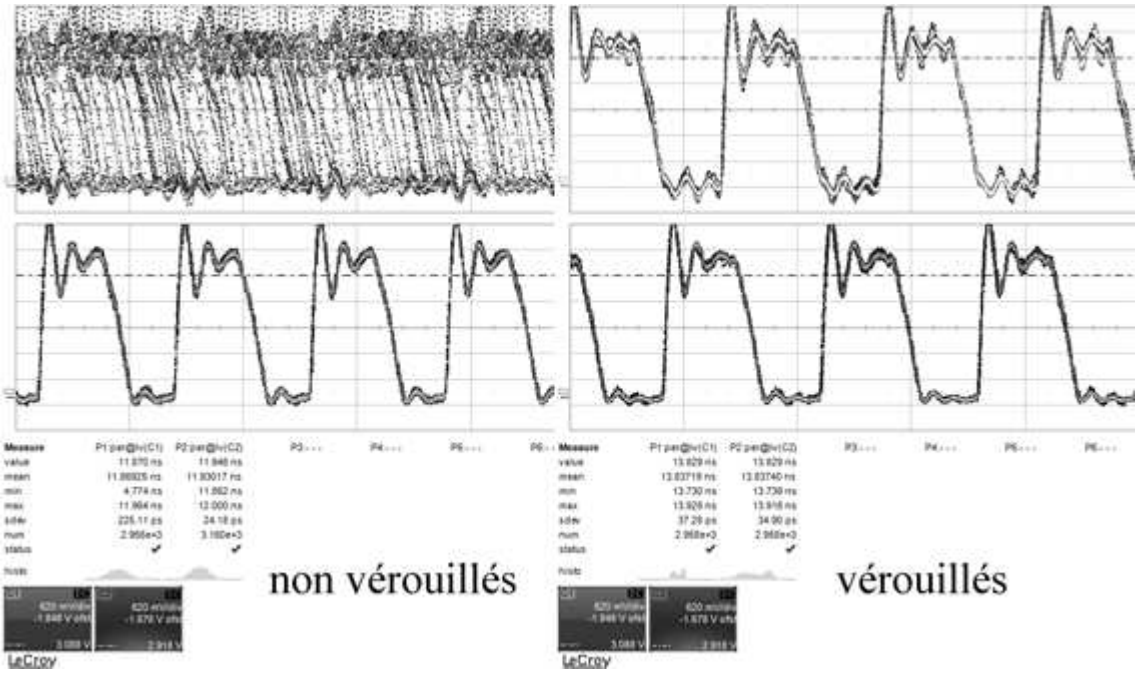

Figure 2.15 Mise en évidence du phénomène de verrouillage de la phase de deux oscillateurs grâce à l'oscilloscope. Celui-ci est déclenché sur le signal du bas. A gauche : à chaque *trigger* les deux signaux sont indépendants, à droite ils sont verrouillés et leur phase est constante dans le temps

plage de tensions et non pas comme on pourrait s'y attendre en un seul point. Ce qui se produit dans cette zone est un verrouillage des deux signaux. Dans cette zone, les deux signaux sont de même période et parfaitement en phase l'un par rapport à l'autre. Ce phénomène est illustré à la Figure 2.15 qui représente une capture d'écran d'oscilloscope synchronisée sur l'un des deux signaux et en mode de persistance infinie. On peut voir que lorsque les signaux ne sont pas verrouillés même si la différence des périodes est petite, les signaux sont en oscillation libre et les phases sont totalement désynchronisés. Par contre, dans la zone verrouillée, leur phase est rigoureusement identique. Ce phénomène est potentiellement dangereux car si ces signaux sont utilisés d'une manière ou d'une autre pour la génération d'aléa alors le comportement du TRNG peut devenir prévisible ou bien diminuer l'entropie de la suite aléatoire. Pour nous en convaincre nous avons évalué une suite aléatoire issue d'un TRNG conformément au principe proposé par Wold *et al.*, composé de 5 oscillateurs en anneaux à l'aide des 4 tests FIPS-140-1. Deux suites aléatoires de 20 000 bits ont été acquies. La première sans le phénomène de verrouillage et la seconde avec au moins 2 des 5 oscillateurs verrouillés. Les 4 tests sont passés avec succès dans le premier cas alors que dans le second le *Poker test* ainsi que le *Runs test* ne passent pas. L'entropie de la suite aléatoire a également chuté de 7,841 bits par caractère (octet) à 7,393 bits par caractère (octet).

Makettos et Moore dans [MM09], réalisent également une perturbation de plusieurs oscillateurs et proposent une attaque complète contre un TRNG embarqué dans

une carte à puce. Cependant la perturbation est obtenue par injection d'un signal externe et non pas en faisant varier statiquement la valeur de la tension d'alimentation du coeur du FPGA. Ils produisent des résultats très similaires à ceux présentés à la Figure 2.15, observant l'état verrouillé pour une plage de fréquences.

Nous avons également validé le fait que le verrouillage se produit en interne même si les signaux restent internes à la cible FPGA et ne sortent pas à l'extérieur à travers les *pads* de sortie. Pour effectuer cette validation nous avons effectué un XOR des deux signaux verrouillés et avons observé la sortie du XOR. Lorsque les signaux étaient verrouillés cette sortie restait stable dans le temps témoignant ainsi de l'état verrouillé des deux signaux.

4.5 Jitter dans les différentes familles de FPGA

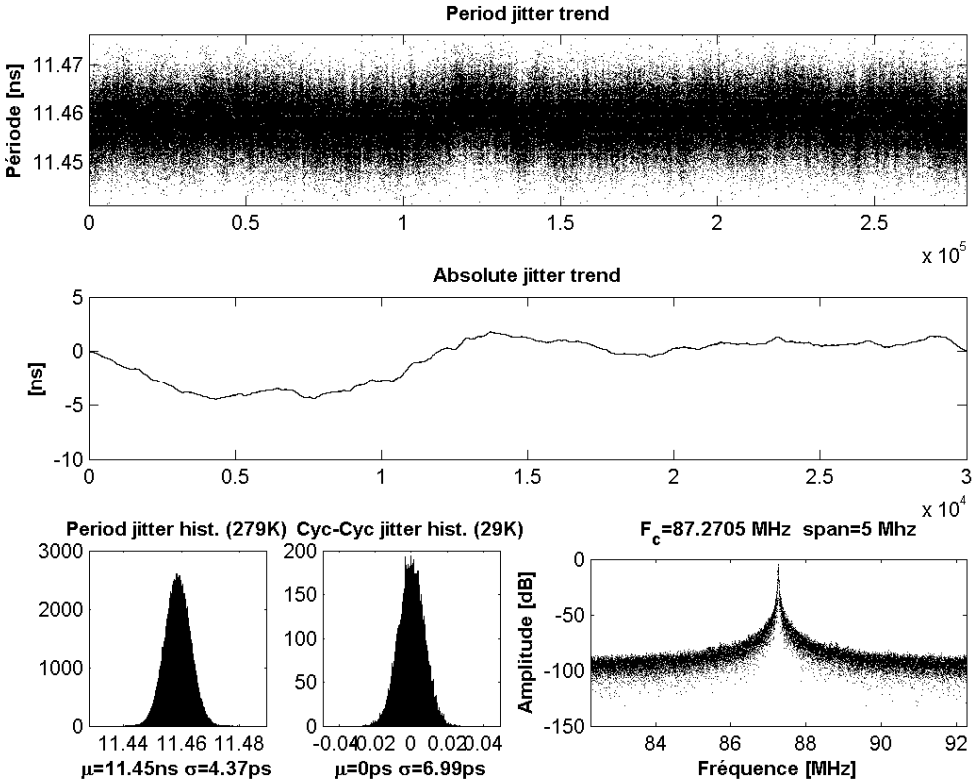

Figure 2.16 Jitter dans un oscillateur en anneau de 7 éléments dans Spartan 3 - XC3S200

Nous avons étudié les profils des jitters issus d'oscillateurs en anneau semblables dans les trois familles de FPGA. Les signaux ont été acquis en externe avec l'oscilloscope numérique LeCroy Wave Pro 7300 et une sonde différentielle Lecroy D320-SP d'une bande passante de 3,5Ghz. Le standard de transmission électronique utilisé est le LVDS retenu pour ses avantages présentés dans la Section 3 de ce chapitre. Le signal issu du FPGA est échantillonné à 40GHz. Ainsi une suite consécutive de mesures des

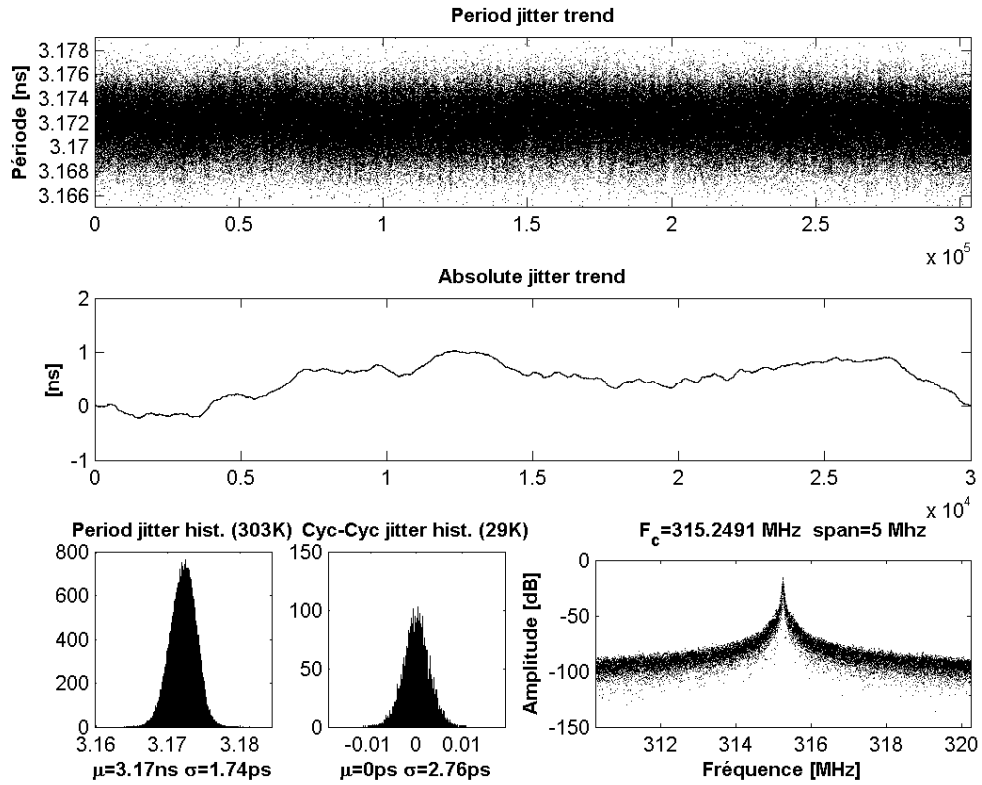

Figure 2.17 Jitter dans un oscillateur en anneau de 7 éléments dans Cyclone III - EP3C25F256

périodes est enregistrée dans la mémoire de l'oscilloscope et exportée dans un fichier Matlab pour analyse. Nous avons représenté aux Figures 2.16, 2.17, 2.18 les mesures consécutives des périodes dans le temps, ce qui revient à s'intéresser à la distribution du *period jitter*. L'accumulation du jitter ou encore le *phase jitter* (ou *absolute jitter*) ainsi que les histogrammes de ces mesures sont également présentés sur ces Figures. La FFT est obtenue directement sur l'oscilloscope et centrée sur la fréquence fondamentale de l'oscillateur (FFT Window VonHann).

Nous pouvons constater que ces 3 familles possèdent donc des jitters dans les oscillateurs très différents. Une synthèse est donnée dans le Tableau 2.4. Nous montrerons dans le Chapitre 4, en étudiant la modélisation des oscillateurs en anneau, que ces différences sont en partie dues aux différentes valeurs des délais introduits par les portes logiques constituants l'oscillateur. Ces valeurs ont été obtenues sur des cartes d'évaluation contenant les FPGA cités, cependant nous avons réalisé les mêmes expériences sur d'autres cartes qui embarquaient les mêmes puces et nous n'avons pas observé des différences entre les mesures. Ainsi ces quantités de jitter sont propres aux différentes puces.

Cette information comparative est importante pour la conception de TRNG, car un même principe d'extraction d'aléa basé sur le jitter devra être paramétré différemment selon la quantité de jitter dans la source.

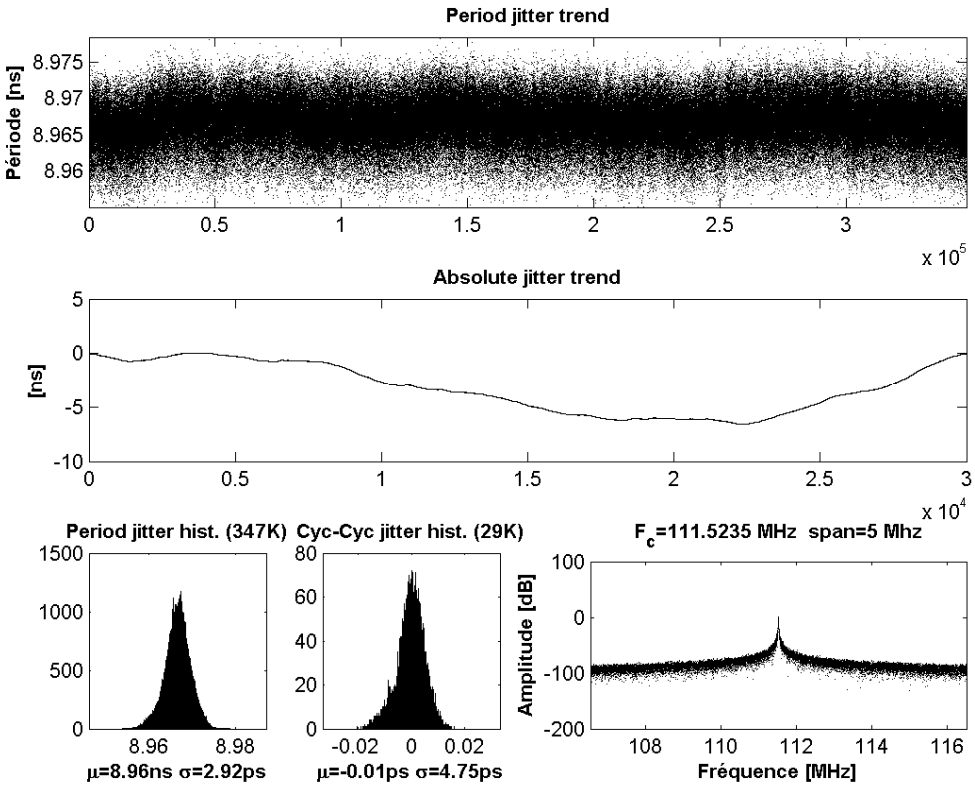

Figure 2.18 Jitter dans un oscillateur en anneau de 7 éléments dans Fusion 600 - M7AFS600FG484

FPGA	Période moyenne T_{moy}	*Period jitter* $\sigma_{per.}$	*Ratio* $\frac{\sigma_{per.}}{T_{moy}}$	*Ratio* $\frac{\sigma_{per.}}{element}$
Xilinx Spartan 3	11.4 ns	4.3 ps	3.85×10^{-4}	0.61 ps
Actel Fusion 600	8.96 ns	2.9 ps	3.18×10^{-4}	0.55 ps
Altera Cyclone III	3.17 ns	1.82 ps	5.74×10^{-4}	0.26 ps

Tableau 2.4 Comparatif des caractéristiques de jitter dans un oscillateur en anneau dans les différentes familles

4.6 Conclusion sur le jitter dans les oscillateurs en anneau

Les oscillateurs en anneau sont une source d'aléa importante pour la génération d'aléa. Dans des conditions idéales leur comportement est complètement aléatoire avec des composantes de bruits blanc et de bruit en 1/F. Ce qui en fait des sources d'aléa précieuses. Les différentes familles de FPGA cependant présentent des caractéristiques différentes en ce qui concerne la quantité de jitter.

Cependant nous avons remarqué que le jitter dans les oscillateurs est très sensible aux conditions externes. La tension d'alimentation ainsi que la température sont des paramètres importants qui agissent directement sur la fréquence des oscillateurs et ce de façon quasi linéaire. Nous avons identifié également deux phénomènes qui sont plus importants au regard de la génération d'aléa : le verrouillage des oscillateurs et l'extrême sensibilité des oscillateurs au fonctionnement de la logique environnante interne au FPGA.

Le verrouillage de deux oscillateurs au moins a été constaté pour une large plage de tensions d'alimentation du coeur du FPGA. Dans cette plage les deux signaux présentent rigoureusement la même fréquence et une phase constante, ce qui peut réduire considérablement l'entropie dans un générateur basé sur les oscillateurs en anneau.

5 Jitter dans les oscillateurs externes

Les circuits numériques ont tous besoin d'une horloge externe pour fonctionner. Cette horloge peut être utilisée comme horloge de référence pour la synthèse de fréquence à l'aide de PLL. Nous avons montré dans le chapitre 1 que certains principes de TRNG utilisent directement le jitter présent dans les signaux générés par ces oscillateurs externes [FMC85], [TLL03], bien que d'un point de vue de la sécurité ce soit un problème. D'autres principes [FD03], [KL06] utilisent ces oscillateurs comme signal de référence de la PLL pour la génération d'aléa

Nous avons analysé le jitter dans deux oscillateurs externes présents dans deux cartes d'évaluation. La carte d'évaluation Actel Fusion System Management avec un oscillateur ECS-2200BX et la carte d'évaluation Digilent Spartan 3 avec un oscillateur SG-8002JF. Les caractéristiques de ces oscillateurs sont présentés dans le Tableau 2.5. Ainsi que l'oscillateur de 16 Mhz Epson –98NG présent sur la carte Altera Cyclone III sur un module PCB fabriqué par la société Micronic.

Le jitter de ces oscillateurs est très différent de celui des oscillateurs en anneau ou des oscillateurs embarqués. En effet ces oscillateurs sont souvent des systèmes électroniques complets qui intègrent une logique de contrôle et de compensation. Dans le cas de l'oscillateur SG-8002JF c'est un composant basé sur une PLL qui produit un signal très stable en fréquence et qui présente la particularité d'avoir un *phase jitter* complètement borné et dont l'estimateur de l'écart type est convergent.

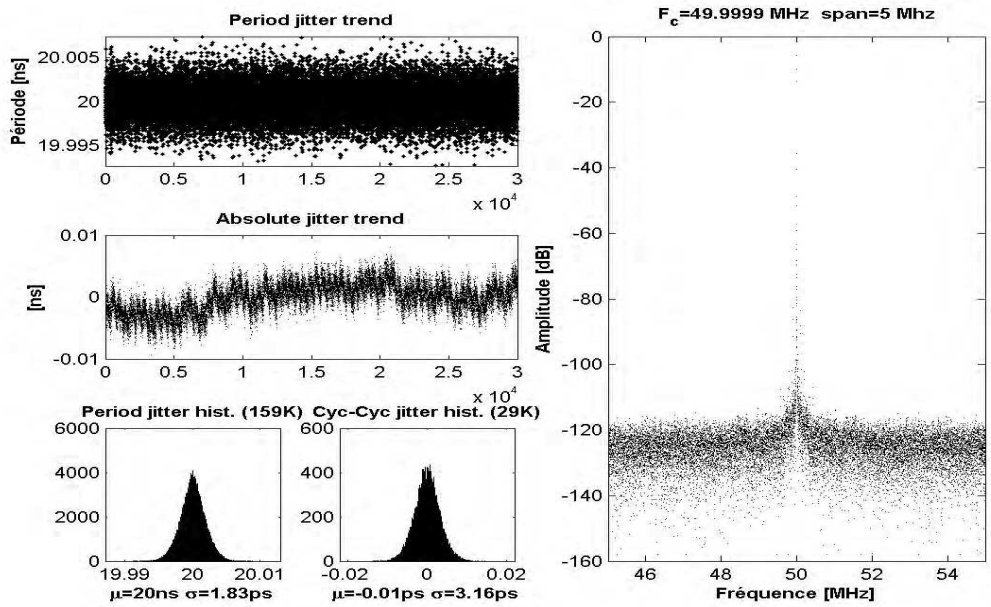

Figure 2.19 Jitter dans un oscillateur externe à quartz Epson SG-8002JF présent sur la carte Digilent Spartan 3 evaluation Kit

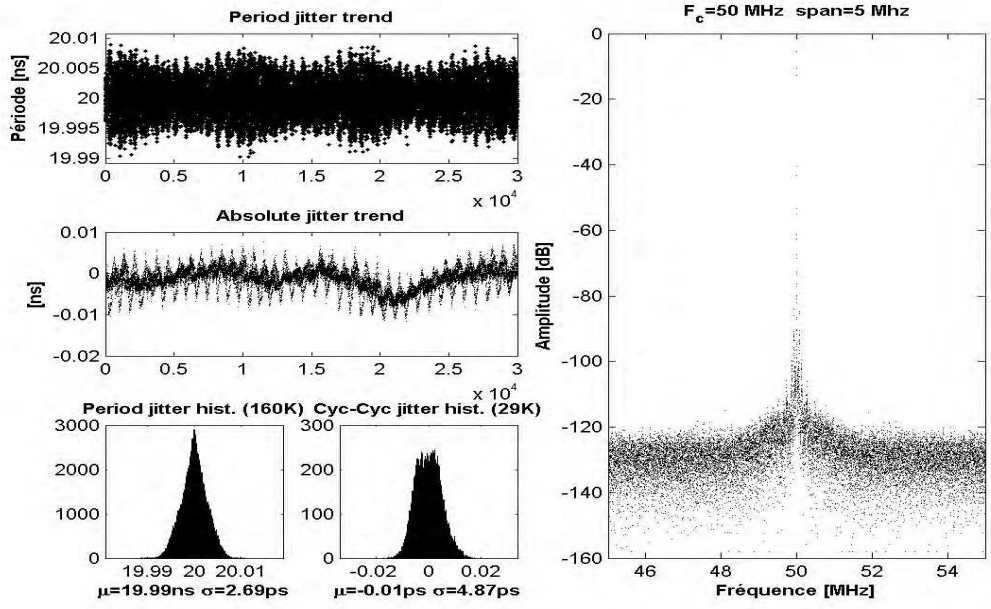

Figure 2.20 Jitter dans un oscillateur externe à quartz ECS-2200BX présent sur la carte System Management Actel Fusion evaluation board

Les données sont présentées aux Figures 2.19 - 2.21. On peut constater que dans les 3 cas l'amplitude du jitter est faible. De plus les trois oscillateurs étudiés présentent la caractéristique des oscillateurs contraints avec un jitter absolu borné et convergent.

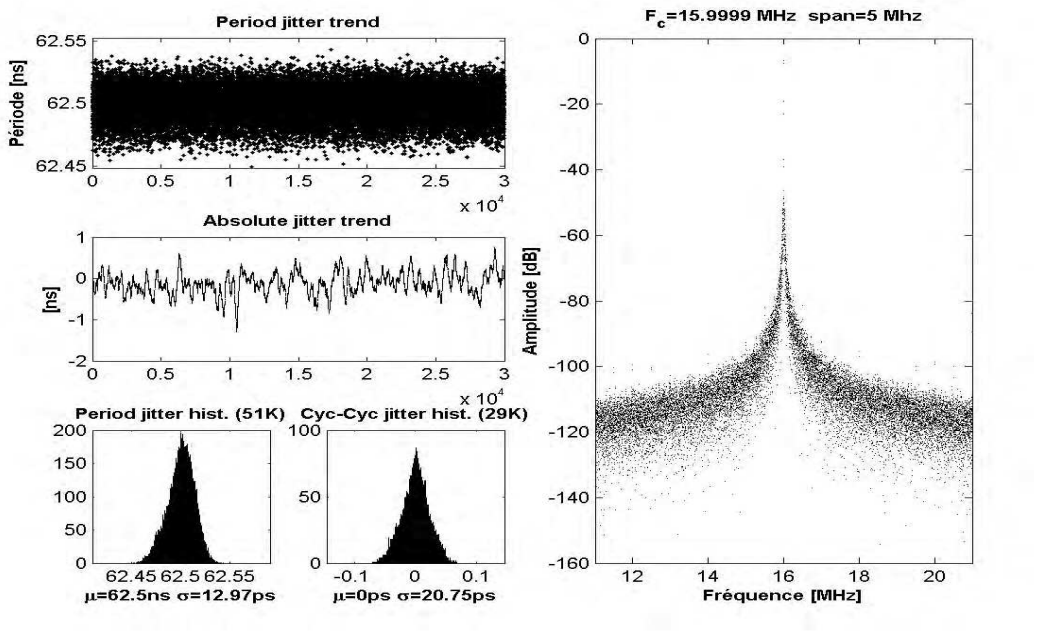

Figure 2.21 Jitter dans un oscillateur externe à quartz Epson –98NG présent sur la carte module Micronic Altera Cyclone III

Nom	Fabricant	Fréquence	Lien *Datasheet*
SG-8002JF	EPSON	50,00 Mhz	[Eps]
ECS-2200BX	ECS Inc. international	50,00 Mhz	[Inca]
–98NG	EPSON	16,00 Mhz	

Tableau 2.5 Les différents oscillateurs externes caractérisés dans le cadre de cette thèse et leur documentation technique

On remarque toutefois dans les trois cas des composantes périodiques importantes potentiellement néfastes pour la génération de bits aléatoires.

Ainsi ces particularités sont à prendre en considération lors de la modélisation de TRNG qui utilisent ces signaux comme source d'aléa.

6 Jitter dans les PLL

Les circuits logiques programmables FPGA intègrent des blocs dédiés à la gestion de l'horloge tels que des PLL (*Phase Lock Loop* ou boucle à verrouillage de phase) ou des DFS. Ces blocs permettent entre autre de synthétiser une fréquence dans une large gamme en fournissant une horloge de référence fixe. Ces blocs sont particulièrement utiles pour la génération d'aléa car ils fournissent une source de signaux contrôlables et moins sensibles que les oscillateurs en anneau aux conditions externes. Les signaux ainsi générés présentent également une part importante de jitter aléatoire.

Les PLL sont un des piliers de l'électronique analogique, ce sont des systèmes étudiés depuis très longtemps et il en existe beaucoup de variantes (PLL Analogique, PLL entièrement numérique, PLL mixte, etc...). C'est un domaine d'application industrielle et de recherche très vaste qui est en dehors du cadre de cette thèse. Nous nous bornerons à présenter une description très simplifiée du principe d'une PLL analogique. C'est ce type de composant que les fabricants de FPGA intègrent dans leurs circuits. Les PLL qui sont disponibles dans les FPGA sont toutes différentes selon le fabricant du FPGA. Elles sont paramétrables par une interface graphique propre à chaque fabricant et de plus les paramètres que l'on peut ajuster diffèrent également selon les fabricants. Nous présenterons les principales familles qui proposent des PLL par la suite mais nous présentons d'abord le fonctionnement simplifié d'une PLL qui est le même pour tous les FPGA.

6.1 Fonctionnement d'une PLL

Une PLL est un système électronique qui génère un signal dont la phase est lié à celle d'un signal de référence. Une PLL compare la phase du signal de référence avec celle dérivée d'un oscillateur interne et ajuste la fréquence de ce dernier (VCO *Voltage Controlled Oscillator*) pour maintenir les phases ajustés. Le principe de fonctionnement est décrit à la Figure 2.22. Asservir la phase d'un signal de sortie à celle d'un signal de

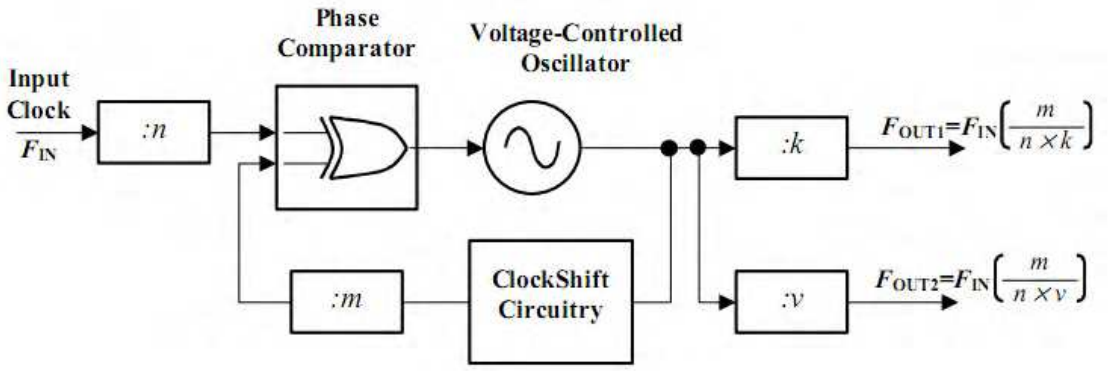

Figure 2.22 Fonctionnement et structure simplifiée d'une PLL

référence revient à verrouiller la fréquence de sortie à celle de référence. Par conséquence une PLL est capable de suivre précisément une fréquence de référence ou de générer une fréquence multiple de la fréquence de référence.

C'est cette dernière caractéristique qui est particulièrement intéressante pour la génération d'aléa car elle permet en intégrant des divisions par M et N avant le signal de référence et après le signal issu du VCO d'obtenir une fréquence fractionnaire de sortie ajustable par les paramètres M et N :

$$F_{out} = \frac{M}{D} F_{in} \tag{2.10}$$

La PLL met un certain temps avant de se verrouiller sur le signal de référence. C'est pourquoi généralement on trouve un signal de sortie qui devient actif lorsque la sortie de la PLL est effectivement verrouillée sur le signal de référence. Ce signal est important car si la PLL n'est pas verrouillée elle n'est en fonctionnement nominal.

6.2 Sources de jitter dans les PLL

Le jitter à la sortie de la PLL est sensiblement plus complexe que celui d'un oscillateur en anneau. En effet la PLL est un système qui intègre beaucoup de circuits actifs dont chacun présente des sources de bruit. Ces circuits sont ainsi sensibles aux bruits dans l'alimentation. Une alimentation propre est donc nécessaire pour en limiter l'impact sur le jitter [AC2]. Les PLL dans les FPGA sont intéressants car il permettent de séparer cette dernière des alimentations des entrées sorties et du coeur numérique.

6.3 PLL dans Actel Fusion

La famille Actel Fusion intègre plusieurs PLL dans la puce afin d'offrir des possibilités de gestion des horloges. Une des particularités de cette famille est d'offrir la possibilité de fournir une alimentation externe séparée limitant ainsi l'impact du jitter déterministe.

Une capture d'écran de la fenêtre de paramétrage de la PLL dans l'environnement de développement d'Actel est présentée à la Figure 2.23. Les paramètres que l'utilisateur peut indirectement modifier sont : M D et P, ainsi que plusieurs délais fixes réglables, donnant une relation pour le signal de sortie :

$$F_{GLA} = \frac{M \times P}{D} F_{CLKA} \tag{2.11}$$

Ces paramètres ainsi que la fréquence du VCO sont pré-calculés par l'outil de configuration, en accord avec les contraintes de fréquences d'entrée et de sortie fournies par l'utilisateur. Le principal inconvénient de la plate-forme Actel est que l'utilisateur ne peut pas modifier ou même connaître la fréquence de coupure du filtre passe bas après le comparateur de phase.

Le jitter à la sortie de la PLL dépend en partie de celui du signal de référence. Ainsi nous avons fait plusieurs expériences avec différents signaux de référence. De plus nous avons observé que le jitter à la sortie de la PLL dépend en grande partie des valeurs des paramètres de la PLL.

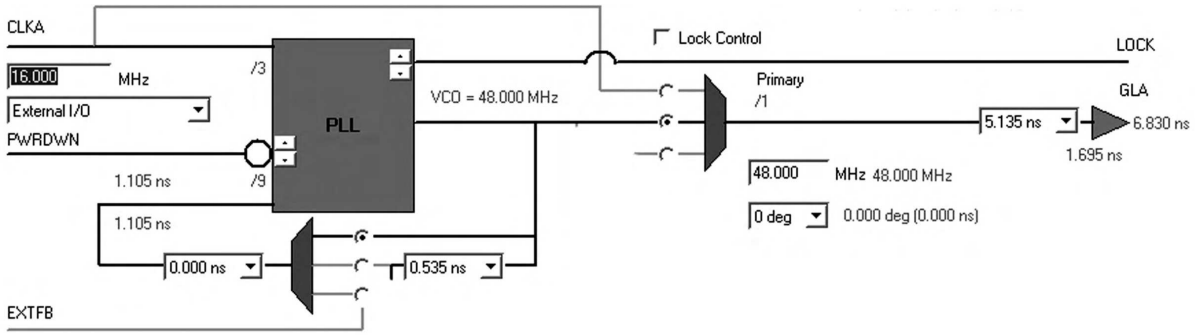

Figure 2.23 Interface graphique de configuration de la PLL dans la famille Actel Fusion

6.3.1 Oscillateur en anneau comme horloge de référence pour la PLL

Les oscillateurs en anneau présentent l'avantage de produire des signaux internes au FPGA c'est pourquoi leur utilisation comme signal de référence est particulièrement intéressante. Nous avons utilisé un signal produit par un oscillateur en anneau de 12 éléments comme signal de référence et nous avons analysé le jitter à la sortie de la PLL avec plusieurs jeux de paramètres différents. Les données expérimentales sont présentées sur les Figures 2.24-2.30 et une synthèse de ces expériences avec l'oscillateur en anneau de 12 éléments est fournie dans le Tableau 2.6.

La Figure 2.25 est particulièrement intéressante car dans ce cas le VCO de la PLL est configuré à la même fréquence de fonctionnement que l'oscillateur en anneau. On peut voir sur cette Figure que la PLL suit l'accumulation de jitter du signal de référence. Ainsi le *period jitter* des deux signaux en entrée et en sortie sont quasiment identiques. Les deux signaux présentent la même phase et la même fréquence. Cependant un phénomène intéressant se produit. En effet le jitter à la sortie de la PLL est supérieur à celui du signal de référence. Ce phénomène est bien visible sur la même Figure dans le plan fréquentiel avec un bruit de phase plus large à la sortie de la PLL. Nous expliquons ce phénomène par le fait que même si l'oscillateur de référence présente en entrée un jitter faible ($\sigma_{per.}$ de 4,09 ps à la figure 2.24), la partie aléatoire du jitter intrinsèque du VCO peut être plus importante. Ainsi cette paire de signaux est particulièrement intéressante pour la génération d'aléa car les deux signaux suivent la même tendance (jitter déterministe) mais avec une partie aléatoire différente. Sur les autres Figures 2.26-2.30 nous montrons l'impact sur le jitter des paramètres de la PLL. Ainsi nous pouvons constater dans certains cas l'apparition de pics (ou *spurs*) dans le domaine fréquentiel à la sortie de la PLL. Ces composantes déterministes sont très gênants pour la génération d'aléa car ils peuvent induire des bits répétitifs à la sortie du TRNG.

6.3.2 Quartz comme horloge de référence

Nous avons procédé à la mesure externe du jitter à la sortie de la PLL lorsque le signal de référence est le signal généré par l'oscillateur externe à quartz ECS-2200BX de la cible Actel Fusion Eval. C'est l'oscillateur dont le jitter a été présenté à la Figure

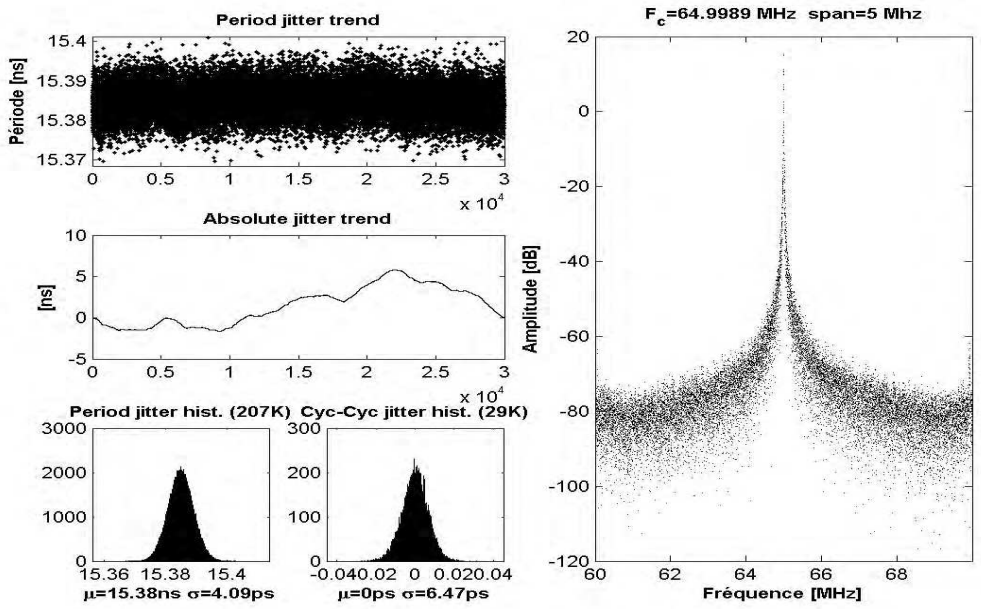

Figure 2.24 Jitter dans un oscillateur en anneau de 12 éléments utilisé comme signal de référence dans une PLL

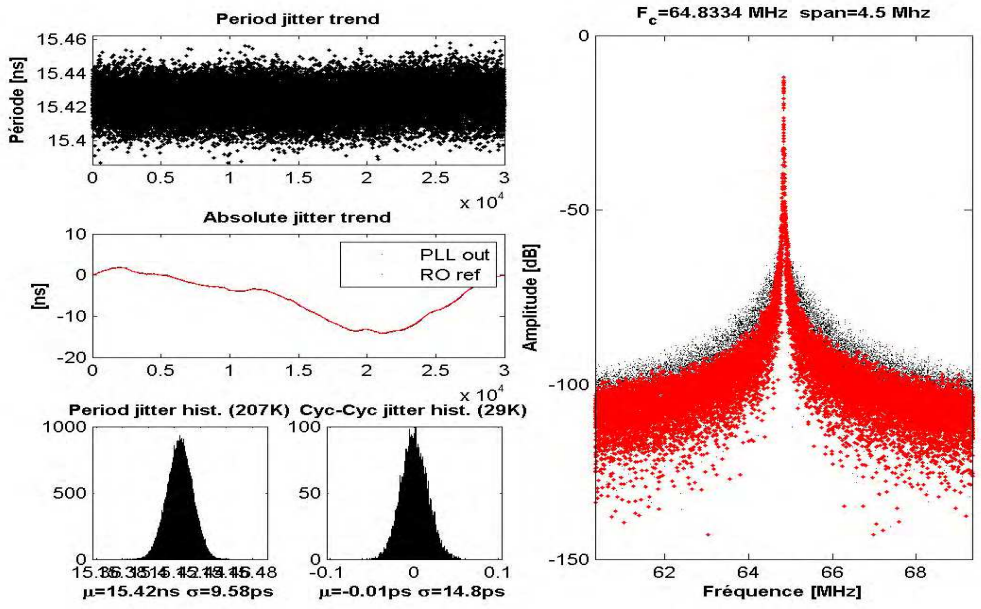

Figure 2.25 Jitter à la sortie d'une PLL dans Actel Fusion M7AFS600M, avec un oscillateur en anneau de 12 éléments comme signal de référence. Paramètres : M=12, D=12, P=1, Lock=1, VCO=65.040Mhz

2.20. Nous pouvons constater que comme dans le cas d'un signal issu d'oscillateur en anneau comme référence, le jitter à la sortie de la PLL est largement plus important

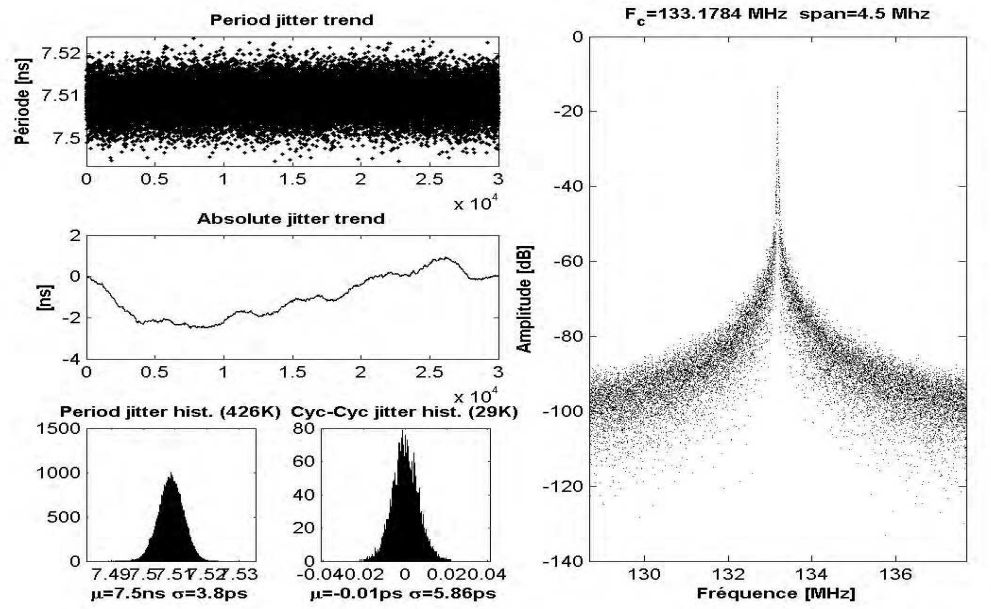

Figure 2.26 Jitter à la sortie d'une PLL dans Actel Fusion M7AFS600M, avec un oscillateur en anneau de 12 éléments comme signal de référence. Paramètres : M=12, D=24, P=1, Lock=1 VCO=130.080Mhz

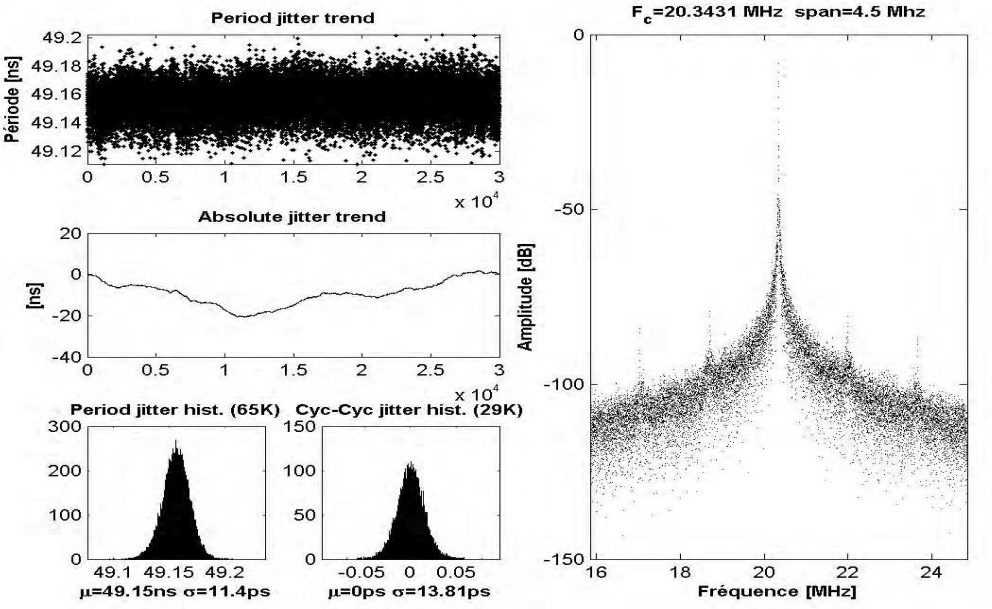

Figure 2.27 Jitter à la sortie d'une PLL dans Actel Fusion M7AFS600M, avec un oscillateur en anneau de 12 éléments comme signal de référence. Paramètres : M=40, D=123, P=10, Lock=1 VCO=199.998Mhz

que celui à son entrée (sauf pour les paramètres de configuration présentés à la ligne

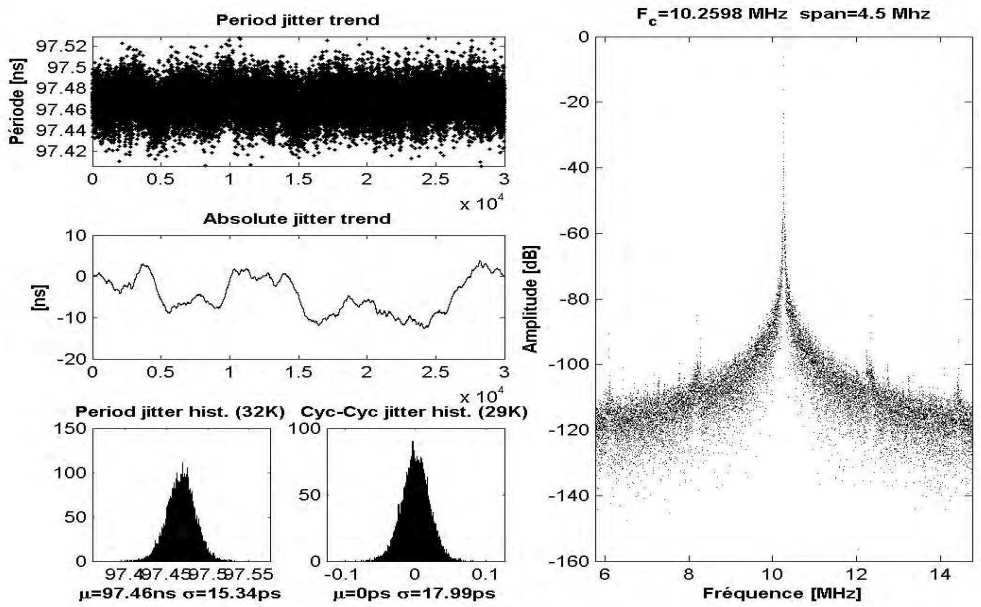

Figure 2.28 Jitter à la sortie d'une PLL dans Actel Fusion M7AFS600M, avec un oscillateur en anneau de 12 éléments comme signal de référence. Paramètres : M=32, D=123, P=15, Lock=1, VCO=249.998Mhz

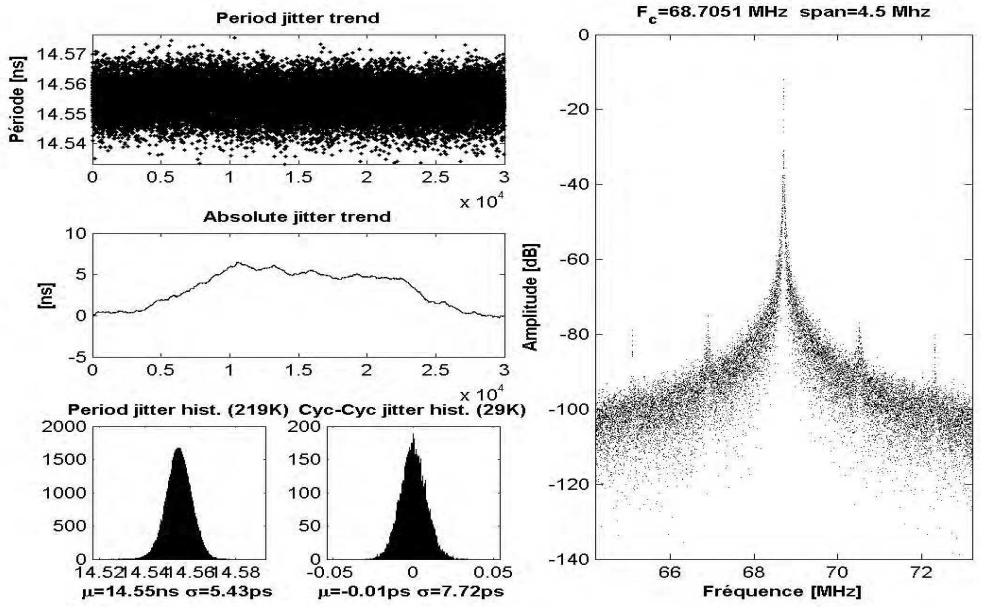

Figure 2.29 Jitter à la sortie d'une PLL dans Actel Fusion M7AFS600M, avec un oscillateur en anneau de 12 éléments comme signal de référence. Paramètres : M=37, D=38, P=1, Lock=1, VCO=66.798Mhz

3 du Tableau 2.7). Cette caractéristique est propre aux PLL de la famille Actel. Nous

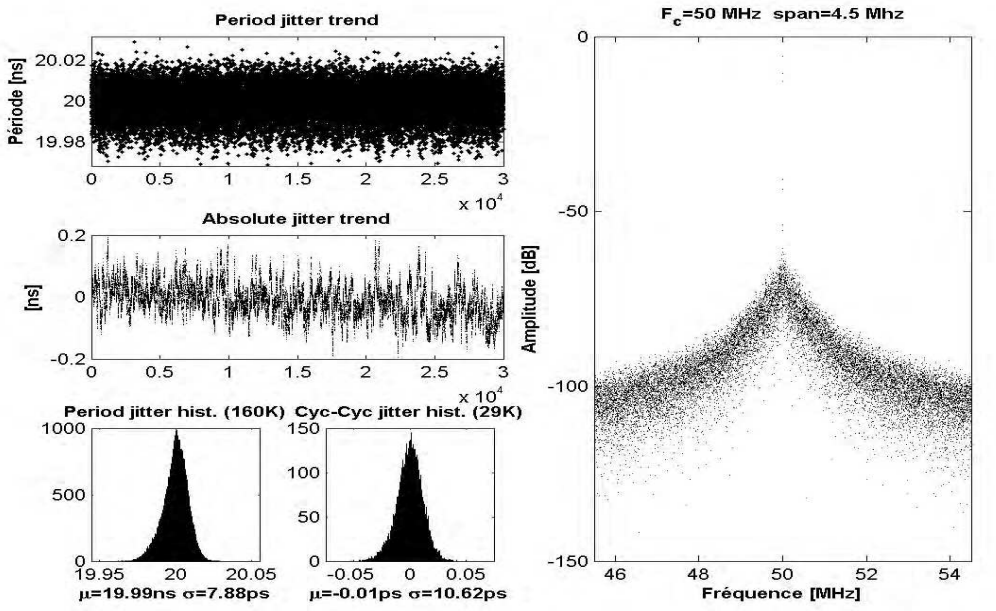

Figure 2.30 Jitter à la sortie d'une PLL dans Actel Fusion M7AFS600M, avec un oscillateur à quartz ECS-2200BX de 50Mhz comme signal de référence. Paramètres : M=10, D=10, P=1, Lock=1, VCO=50Mhz

pouvons également constater que le *phase jitter* ici est borné comme celui de l'oscillateur à quartz utilisé comme signal de référence.

M	D	P	lock	F_{VCO}	F_{out}	σ_{RMS}	σ_{C-C}	notes
					65.04 Mhz	4.09 ps	6.49 ps	osc. en anneau seul
12	12	1	oui	65.04Mhz	65.04Mhz	9.58 ps	14.66 ps	augmentation du jitter
12	24	1	oui	130.08Mhz	130.08Mhz	3.8 ps	5.81 ps	
40	123	10	oui	199.998Mhz	20.34Mhz	11.4 ps	13.59 ps	Spurs
32	123	15	oui	249.998Mhz	10.25Mhz	15.34 ps	17.9 ps	Spurs
37	38	1	oui	66.798Mhz	66.798Mhz	5.43 ps	7.69 ps	Spurs

Tableau 2.6 Synthèse des caractéristiques de jitter en entrée et en sortie d'une PLL dans Actel M7AFS600FG484

6.4 PLL dans Altera Cyclone III

Les FPGA de la famille Altera Cyclone III présentent également l'avantage d'être dotés de PLL. En effet ce FPGA comporte 4 PLL embarquées. Une description des PLL de cette famille est fournie à la Figure 2.31. On trouvera la *datasheet* de ces PLL dans [UA]. Le signal de sortie est donné par l'équation :

$$F_{out_1} = \frac{M}{N \times K} F_{in} \tag{2.12}$$

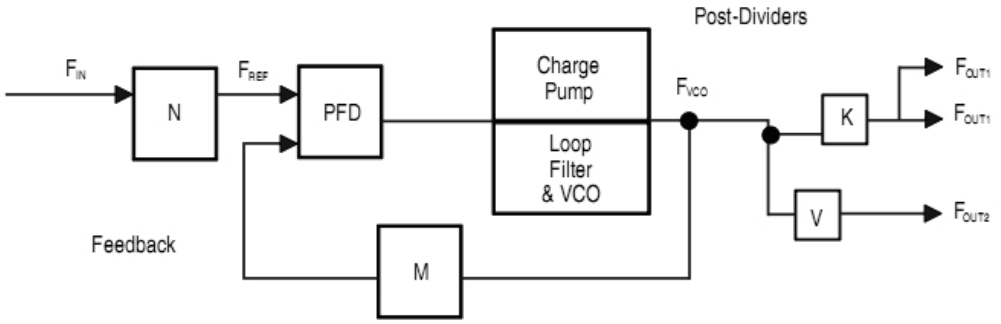

Figure 2.31 Schéma simplifié de la PLL dans Altera Cyclone III

Ces PLL sont différentes de celles embarqués dans les familles Actel et Xilinx. En effet en plus du fait que les coefficients de multiplication et division sont paramétrables dans une plage plus importante (de 1 à 512), ces PLL offrent la possibilité de configurer la bande passante du filtre interne. Or cette caractéristique est essentielle pour les performances du point de vue du jitter. De plus, le VCO interne fonctionne à une vitesse importante de 640Mhz. Dans le cadre de cette campagne de caractérisation, nous nous bornerons aux coefficients M et N uniquement.

Cependant ces PLL présentent l'inconvénient de n'accepter en entrée que des signaux d'horloge issus de broches physiquement dédiées à cette opération. Ainsi nous n'avons pas été capables d'utiliser un oscillateur en anneau comme signal de référence avec ces PLL. Ceci limite considérablement la liberté du concepteur de TRNG car il se voit obligé d'utiliser des signaux externes comme signaux de référence et la sécurité du TRNG est aussi diminuée.

Les résultats de caractérisation du jitter dans les PLL de la famille Altera Cyclone III sont présentés aux Figures 2.32 - 2.37 et synthétisés dans le Tableau 2.7. On peu constater que comme dans la famille Actel Fusion le jitter à la sortie de la PLL est fortement influencé par les paramètres de configuration (coefficients, bande passante du filtre, *etc...*). Cependant contrairement aux PLL de la famille Actel, nous constatons que globalement le jitter est diminué à la sortie par rapport au jitter du signal de référence.

6.5 Conclusion sur le jitter dans les PLL

Nous avons montré que les PLL offrent une source d'aléa intéressante dans les FPGA. Elles permettent notamment d'obtenir des signaux paramétrables qui contiennent du jitter. Bien que des composantes périodiques sont parfois présentes, elles offrent une flexibilité plus importante et une immunité aux perturbations externes plus importantes que les oscillateurs en anneau.

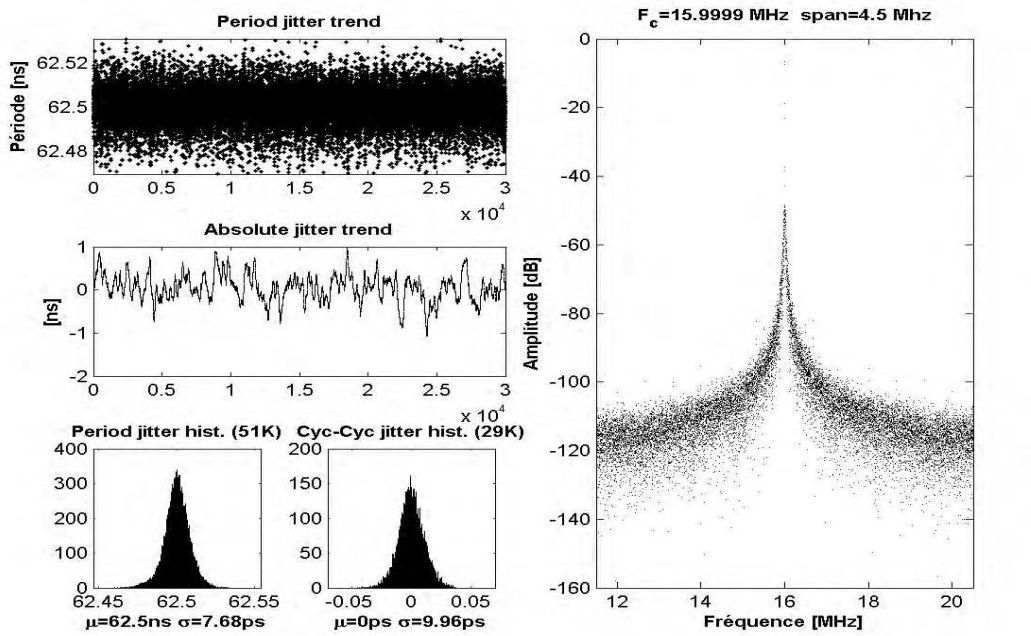

Figure 2.32 Jitter à la sortie d'une PLL dans Altera Cyclone III(module Micronic), avec un oscillateur à quartz –98NG de 16Mhz comme signal de référence. Paramètres : M=1, D=1, Bandwidth=Low

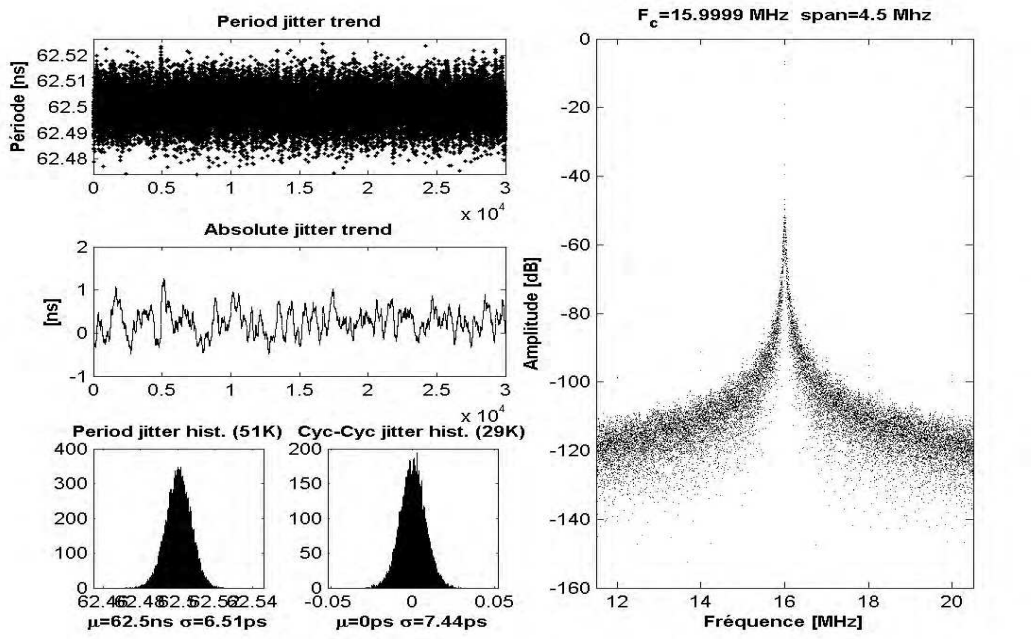

Figure 2.33 Jitter à la sortie d'une PLL dans Altera Cyclone III(module Micronic), avec un oscillateur à quartz –98NG de 16Mhz comme signal de référence. Paramètres : M=1, D=1, Bandwidth=Medium

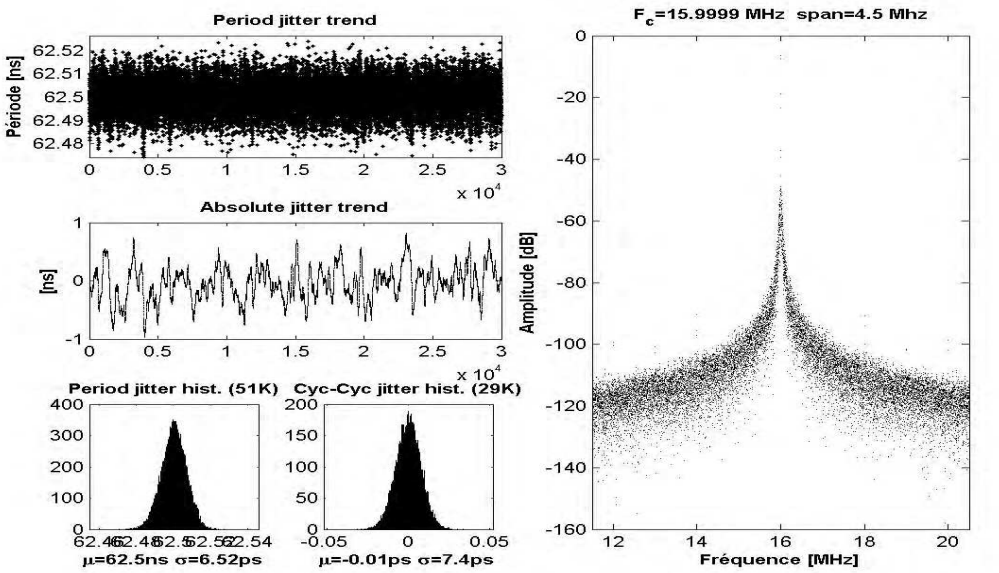

Figure 2.34 Jitter à la sortie d'une PLL dans Altera Cyclone III(module Micronic), avec un oscillateur à quartz –98NG de 16Mhz comme signal de référence. Paramètres : M=1, D=1, Bandwidth=High

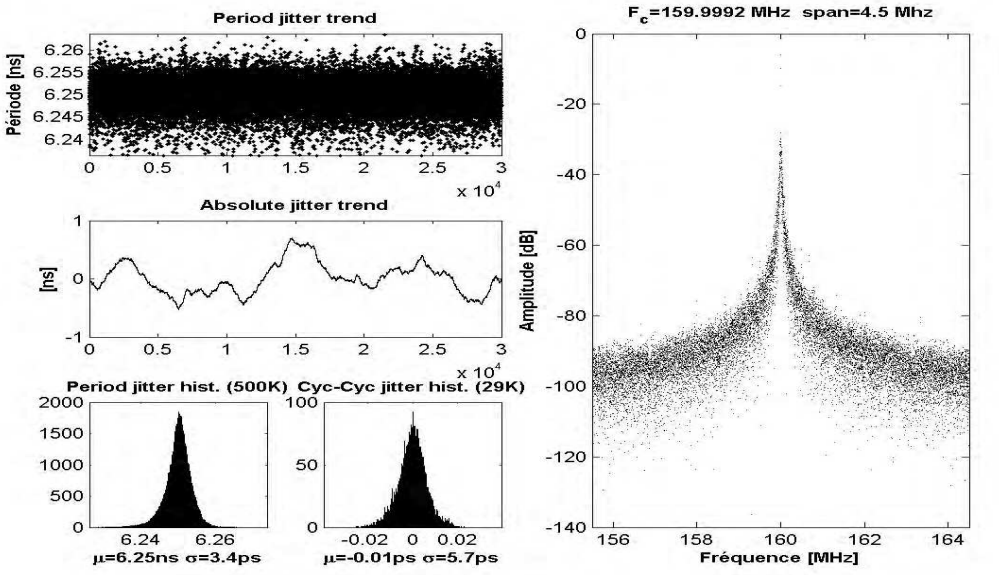

Figure 2.35 Jitter à la sortie d'une PLL dans Altera Cyclone III(module Micronic), avec un oscillateur à quartz –98NG de 16Mhz comme signal de référence. Paramètres : M=10, D=1, Bandwidth=Low

Les différentes familles de FPGA offrent des PLL. L'amplitude de la partie aléatoire du jitter dépend en grande partie du signal de référence à l'entrée de la PLL ainsi que des paramètres de réglage de la PLL. Nous avons montré que les PLL acceptent

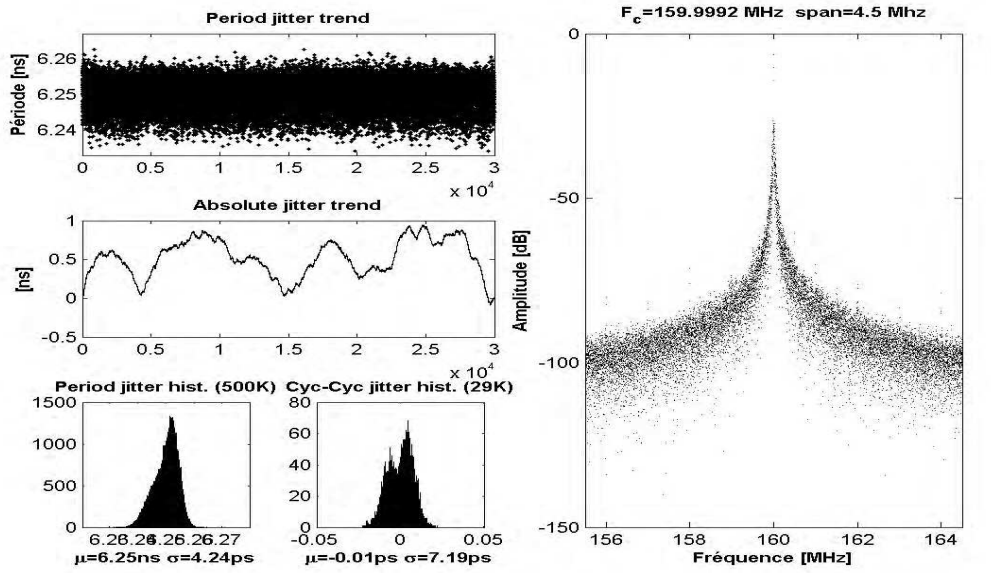

Figure 2.36 Jitter à la sortie d'une PLL dans Altera Cyclone III(module Micronic), avec un oscillateur à quartz –98NG de 16Mhz comme signal de référence. Paramètres : M=10, D=1, Bandwidth=Medium

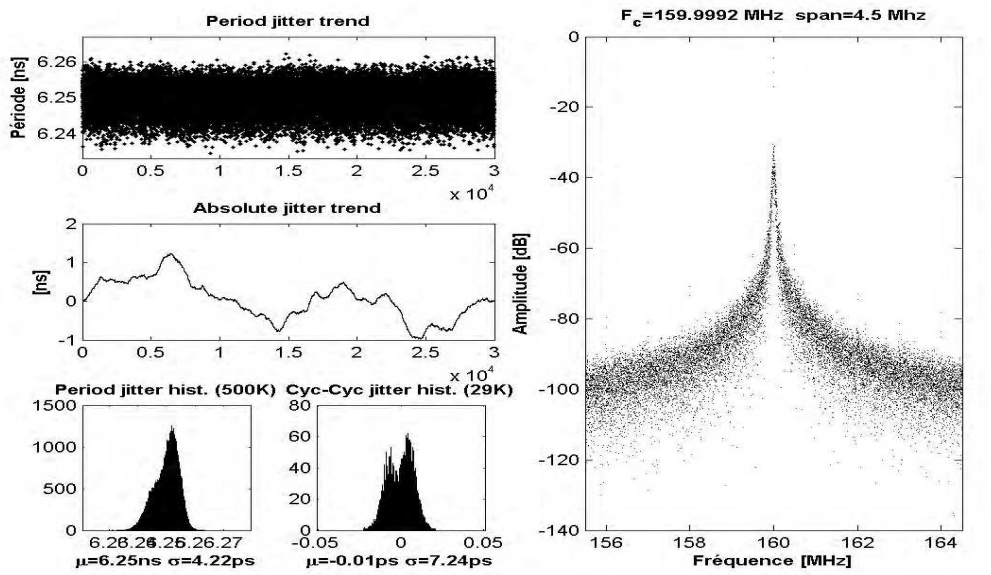

Figure 2.37 Jitter à la sortie d'une PLL dans Altera Cyclone III(module Micronic), avec un oscillateur à quartz –98NG de 16Mhz comme signal de référence. Paramètres : M=10, D=1, Bandwidth=High

une grande variété de signaux. Ainsi le jitter à la sortie de la PLL est plus complexe que celui d'un oscillateur en anneau.

M	D	BW	Lock	VCO	F_{out}	$\sigma_{Per.}$	σ_{C-C}	notes
						12.97 ps	20.75 ps	
1	1	Low	oui	640Mhz	16	7.68 ps	9.96 ps	jitter globalement diminué
1	1	Medium	oui	640Mhz	16	6.51 ps	7.44 ps	jitter globalement diminué
1	1	High	oui	640Mhz	16	6.52 ps	7.4 ps	jitter globalement diminué
10	1	Low	oui	640Mhz	160	3.4 ps	5.7 ps	jitter globalement diminué
10	1	Medium	oui	640Mhz	160	4.24 ps	7.19 ps	jitter globalement diminué
10	1	High	oui	640Mhz	160	4.22 ps	7.24 ps	jitter globalement diminué

Tableau 2.7 Synthèse des caractéristiques de jitter en entrée et en sortie d'une PLL dans Altera Cyclone III

7 Jitter dans les DFS

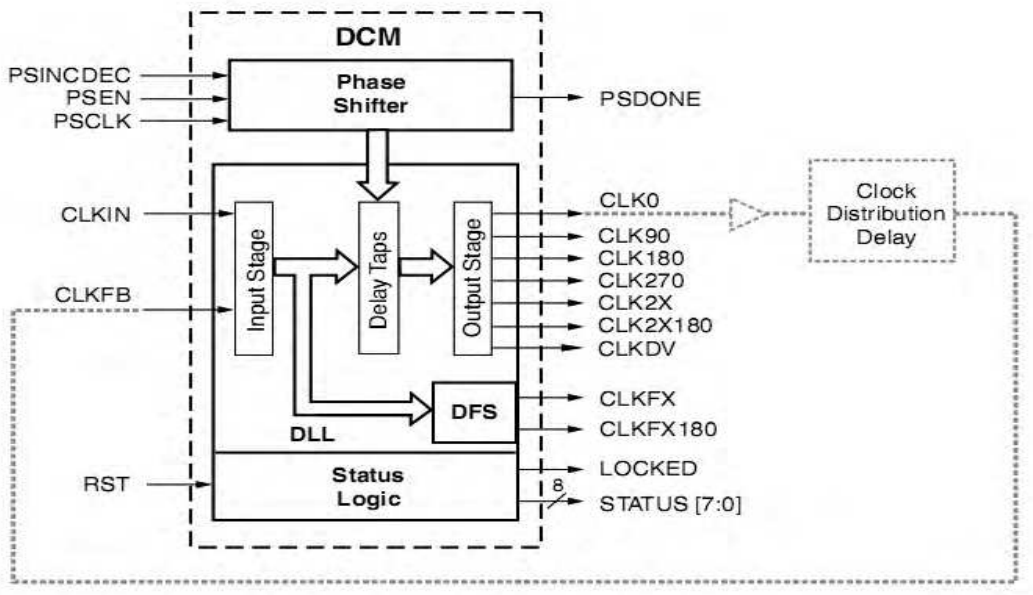

Figure 2.38 DFS principe

La famille Xilinx Spartan 3 ne propose pas de PLL embarquées. Il faut utiliser la famille Xilinx Virtex 5 pour avoir accès à ces composants sous Xilinx. Cependant, il existe la possibilité de synthétiser des signaux de fréquences réglables. Ces blocs ont été plusieurs fois utilisés pour la génération d'aléa [KL06] [TLL03]. Les composants qui accomplissent cette fonction sont les DFS *Digital Frequency Synthesis*. Les DFS font partie d'un bloc dédié à la gestion de l'horloge dans ces FPGA qui est le DCM *Digital Clock Manager*. Une description schématique avec les différents signaux est donné à la Figure 2.38. Les DCM intègrent des DLL *Delay Locked Loop* pour compenser le *clock skew* et fournir des sorties avec un déphasage programmable. Les DFS peuvent fonctionner en deux modes : avec ou sans DLL. Dans le cadre de cette campagne de caractérisation,

nous avons utilisé les DFS en mode synthèse de fréquence sans l'utilisation de DLL. Les DFS sont ainsi utilisés pour synthétiser un signal CLKFX à partir d'un signal de référence CLKIN grâce à deux coefficients M et D selon la relation :

$$F_{CLKFX} = F_{CLKIN} \times \frac{M}{D} \tag{2.13}$$

Le choix des coefficients est toutefois limité de 2 à 32 pour M et de 1 à 32 pour D offrant ainsi un gamme plus restreinte de synthèse de fréquences comparativement aux PLL. Il est possible d'utiliser comme signal de référence aussi bien les lignes d'horloge système comme des signaux de logique interne. Il est donc possible d'utiliser le signal issu d'un oscillateur en anneau comme signal de référence.

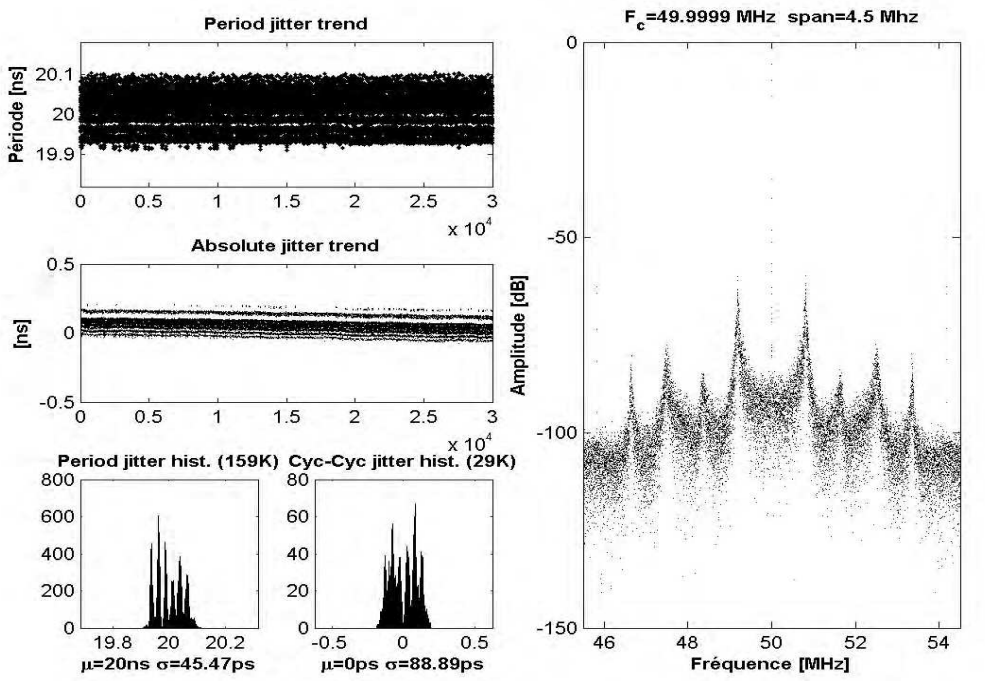

Figure 2.39 Jitter à la sortie d'un bloc DFS dans Xilinx Spartan 3 avec un oscillateur à quartz SG-8002JF de 50 Mhz comme signal de référence. Paramètres : M=2 D=2

Dans le cadre de cette campagne, nous avons utilisé l'oscillateur externe ESG-8002JF présent sur la carte Digilent Spartan 3 comme signal de référence pour illustrer le profil de jitter présent à la sortie des DFS. Ce jitter est très différent de ceux observés dans les PLL ou dans les oscillateurs en anneau. Les Figures 2.39 - 2.41 montrent le profil de jitter pour différents paramètres de configuration des DFS. On constate que le bloc DFS remplit bien la fonction de synthèse demandée par les coefficients M et D. Cependant, on note une modulation discrète importante du jitter importante qui introduit une composante déterministe dans le jitter. De plus, cette modulation discrète est différente pour chaque jeu de paramètres. Ainsi la modélisation des DFS pour la conception de générateurs d'aléa est relativement difficile.

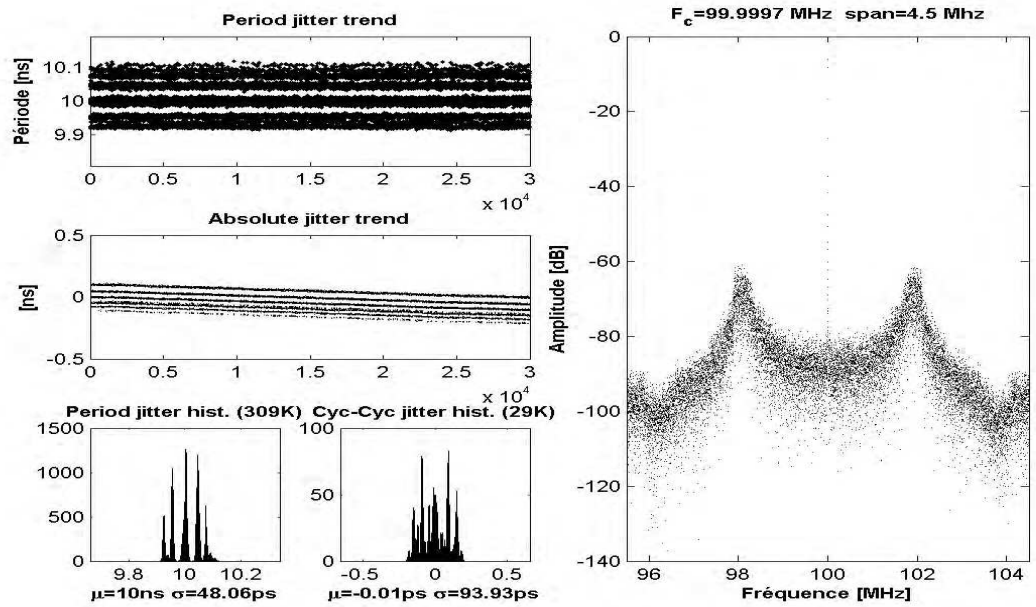

Figure 2.40 Jitter à la sortie d'un bloc DFS dans Xilinx Spartan 3 avec un oscillateur à quartz SG-8002JF de 50 Mhz comme signal de référence. Paramètres : M=2 D=1

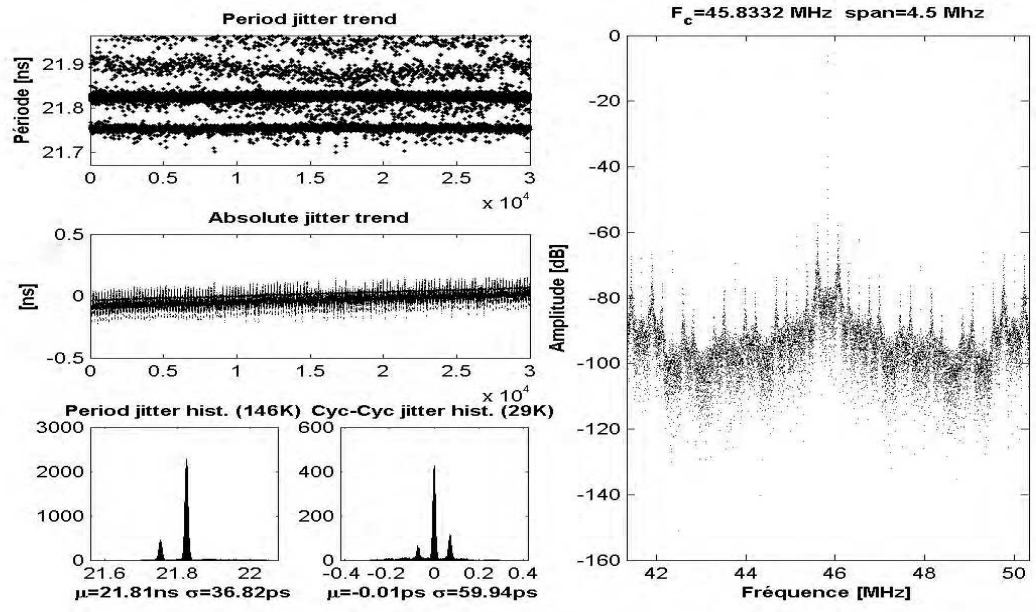

Figure 2.41 Jitter à la sortie d'un bloc DFS dans Xilinx Spartan 3 avec un oscillateur à quartz SG-8002JF de 50 Mhz comme signal de référence. Paramètres : M=11 D=12

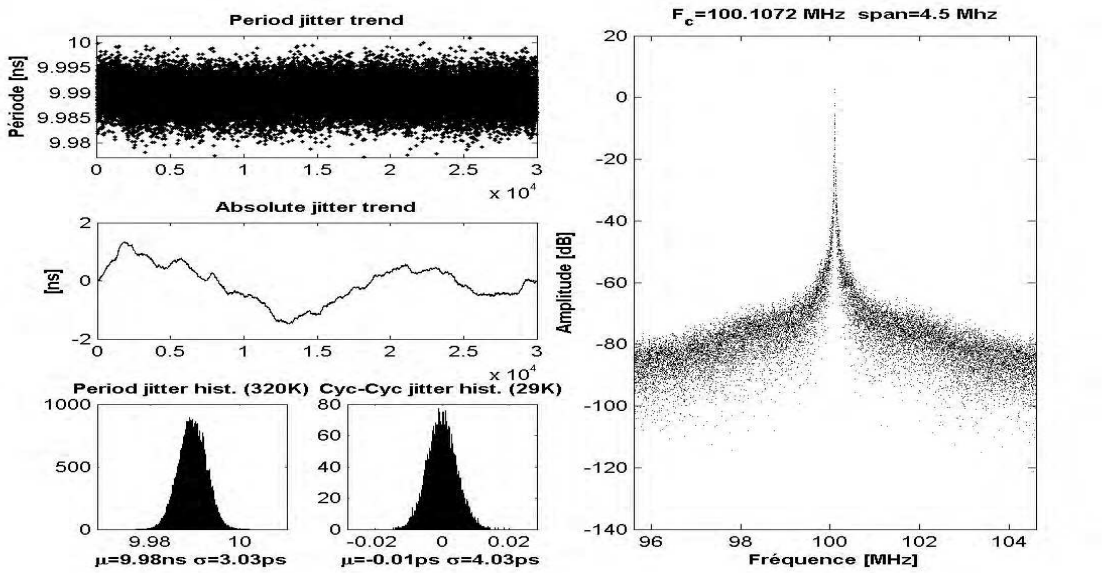

Figure 2.42 Jitter dans le plan temporel et fréquentiel d'un signal issu d'un oscillateur RC embarqué dans Actel Fusion M7AFS600

8 Jitter dans les oscillateurs embarqués dans les FPGA

Les FPGA de la famille Actel Fusion ont la particularité d'intégrer un oscillateur RC. Cette fonctionnalité est particulièrement intéressante pour la génération d'aléa car le signal issu de cet oscillateur contient inévitablement du jitter et contrairement aux signaux obtenus par des oscillateurs à quartz il est interne au FPGA. Le jitter présent dans cet oscillateur est présenté à la Figure 2.42. Comme on peut le constater son allure est très propre. On ne constate aucune composante périodique dans son spectre contrairement aux signaux obtenus avec des oscillateurs en anneau ou bien des PLL. Cependant il n'est pas paramétrable et sa fréquence est fixée à 100Mhz. De plus nous avons observé qu'il est sujet à des perturbations par la logique environnante tout comme les oscillateurs en anneau.

9 Synthèse comparative

10 Conclusion

Nous avons montré dans ce chapitre l'importance de la caractérisation d'une source d'aléa particulière - le jitter. En effet comme nous l'avons montré dans le Chapitre 1, le jitter étant largement utilisé comme source d'aléa dans les TRNG, cette caractérisation est souvent absente dans les articles traitant des TRNG.

Sources de jitter	Avantages	Inconvénients
Oscillateurs en anneau	– Facilement implantable dans toutes les cibles – Forte composante aléatoire du jitter	– Sensibilité importante aux conditions externes (température, tension d'alimentation) – Sensibilité importante au fonctionnement de la logique environnante
Oscillateurs à quartz	– Composante déterministe du jitter très faible – Peu sensible aux conditions externes à la logique environnante	– Généralement des composants externes à la cible FPGA – Faible composante aléatoire (fréquence très stable)
PLL Altera Cyclone III	– Coefficients M et D compris dans une plage très étendue – Bande passante du filtre réglable	– Ne peut pas fonctionner avec un signal de référence issu d'un oscillateur en anneau
PLL Actel Fusion	– Peut fonctionner avec une large gamme de signaux de référence (oscillateurs en anneau, RC, ...) – Fréquence ajustable dans une plage étendue	– Composantes déterministes du jitter fortement dépendantes des paramètres de la PLL
DFS Spartan 3	– Nombreux blocs DFS disponibles dans beaucoup de cibles Xilinx – Possibilité de cascader facilement plusieurs blocs DFS	– Composante déterministe du signal très importante
Oscillateur RC Actel Fusion	– Composant interne à la cible FPGA – Composante déterministe faible	– Disponible uniquement sur les plate-formes Actel – Fréquence non paramétrable

Tableau 2.8 Synthèse comparative générale des sources de jitter disponibles dans les différentes familles de FPGA étudiées dans le cadre de cette thèse

Dans un premier temps, nous avons soulevé la problématique de la mesure en in-

terne par rapport à la mesure en externe du jitter présent dans les signaux périodiques. La mesure en interne présente l'avantage de ne pas influencer le signal à mesurer, par contre elle est très difficile à obtenir à cause des fréquences élevés des signaux à mesurer. La mesure en externe à l'aide d'un oscilloscope fournit des résultats beaucoup plus précis grâce à la grande vitesse d'échantillonnage de l'oscilloscope. Cependant elle présente l'inconvénient d'influencer le signal à mesurer et elle est donc susceptible de fausser les valeurs mesurés du jitter. Nous avons montré que la mesure en externe fournit des résultats très différents selon le type de sonde utilisé mais également selon le standard de transmission utilisé par le buffer de sortie du FPGA (LVDS, LVTTL). Nous avons publié cette observation dans [VAFB10]. Cette observation est particulièrement importante dans l'estimation du jitter dans les cibles FPGA car ce dernier est très souvent la source d'aléa du TRNG et une erreur sur sa mesure peut avoir des répercussions lors de la modélisation et la qualité de la suite aléatoire obtenue.

Dans un second temps nous avons choisi d'utiliser une méthode externe de mesure du jitter. Cette même méthode a été utilisée pour confronter les comportements du jitter présent dans des signaux périodiques issus de différentes sources largement utilisées dans la conception de TRNG. Ainsi nous avons montré que les différentes sources de signaux périodiques à l'intérieur du FPGA présentent des comportements du jitter fortement différents les uns des autres. Ainsi un même principe de TRNG utilisant deux sources de signaux périodiques différentes aura des comportements très différents. Nous avons proposé une synthèse comparative des sources de jitter sous la forme de tableau décrivant les avantages et les inconvénients. Nous avons montré également que dans le cas des signaux issus d'oscillateurs en anneau, le jitter dépend également de la topologie de l'oscillateur, notamment des interconnexions et du type d'élément à retard utilisé. Ces caractérisations externes ont fait l'objet de plusieurs communications scientifiques [VAFB10] [VFA09b] [VFAF09].

Enfin un phénomène nouveau a été présenté dont les conséquences sont importantes pour le domaine de la génération d'aléa. En effet le verrouillage ou *locking* obtenu en modifiant la tension d'alimentation du FPGA peut se révéler potentiellement très dangereux pour la sécurité des générateurs d'aléa basés sur les oscillateurs en anneau particulièrement pour ceux qui mélangent du vrai aléa et du pseudo-aléa. Ce phénomène nouveau a fait l'objet d'une publication scientifique dans un journal [BBFV10] et d'une communication dans une conférence internationale [FBVB10].

Chapitre 3

Implantation de TRNG dans les FPGA

Dans ce chapitre seront présentés deux principes d'extraction physique d'aléa à partir de signaux d'horloges jittés. Un premier extracteur basé sur des compteurs sera étudié comme alternative à l'extraction d'aléa historique présenté dans [FMC85] utilisant une bascule DFF. En effet il sera montré que l'utilisation de l'accumulation du jitter permet d'améliorer les qualités de la classe de générateurs à base d'échantillonnage. Avec cette nouvelle approche ce générateur fournit uniquement du vrai aléa à la sortie et pas un mélange de pseudo aléa et de vrai aléa. De plus, la modélisation mathématique se retrouve simplifiée. Cependant ces améliorations se feront au détriment d'une diminution importante du débit. Un second principe d'extraction physique d'aléa basé sur l'échantillonnage cohérent sera présenté. Deux signaux de fréquences très proches seront mélangés pour fournir un signal de battement dont la période est directement proportionnelle au jitter dans les deux signaux. Enfin diverses configurations de ces extracteurs formant des TRNG complets, avec différentes sources de signaux jittés, seront analysées et évaluées dans différentes familles à l'aide de la suite de tests statistiques NIST SP800-22.

1 Introduction

Nous allons décrire dans ce chapitre le principe et l'implantation de deux *extracteurs physiques* d'aléa. La source de cet aléa sera le jitter contenu dans des signaux périodiques produits par l'une ou l'autre des sources décrites dans le Chapitre 2. Un des objectifs de cette thèse est de trouver un moyen d'obtenir des suites aléatoires de bonne qualité facilement implantables dans un grand nombre de familles différentes de FPGA. C'est pour cette raison que les oscillateurs en anneau seront largement utilisés dans ce chapitre. Cependant nous rechercherons à réduire au maximum l'intervention manuelle pour l'implantation de ces générateurs. En effet dans une optique réelle de production d'un produit sécurisé embarquant un TRNG, même à moyenne échelle, une

telle intervention est inacceptable car elle serait trop coûteuse en terme de ressources humaines qualifiées pour sa réalisation. C'est pour cette raison que nous validerons également les extracteurs physiques d'aléa avec des PLL.

Un autre objectif tout aussi important sera l'obtention de suites aléatoires de très bonne qualité statistique sans l'utilisation de *post-traitement* complexe. Comme nous l'avons montré dans le Chapitre 1, la plupart des générateurs produisent des suites aléatoires de bits indépendants mais souvent biaisés. Ces suites sont transformées en une suite de bits indépendants non biaisés par des correcteurs ou post-traitement. Cependant dans tous les cas décrits dans le Chapitre 1, les correcteurs opèrent sur des suites de bits déjà extraits.

Nous étudierons également dans ce chapitre l'effet d'une fonction déterministe simple comme post-traitement. Cependant ce post-traitement diffère de la majorité de ceux exposés dans le Chapitre 1. En effet, nous proposons d'utiliser la fonction déterministe *avant* d'extraire logiquement les bits aléatoires, c'est à dire sur un signal aléatoire de n bits qui suit une certaine loi de probabilité. Une fois ce signal passé dans la fonction de post-traitement, un extracteur logique aura pour fonction d'extraire autant de bits aléatoires et uniformément distribués que possible.

Ce chapitre aura pour but d'introduire deux principes d'extraction physique d'aléa contenu dans le jitter. L'accent sera mis ici sur la présentation des résultats expérimentaux. Le premier extracteur physique d'aléa sera basé sur l'accumulation du jitter à l'aide de compteurs, le second reprendra un principe existant, présenté dans [KG04], en l'améliorant et en le généralisant, tout en proposant une plus grande souplesse d'utilisation. Les deux principes seront implantés dans des cibles matérielles et des résultats expérimentaux seront donnés comme preuve du concept. L'explication théorique complète sera quant à elle fournie dans le Chapitre 4 où une modélisation de ces extracteurs et d'une source physique d'aléa, l'oscillateur en anneau, seront présentés. Dans un second temps, nous évaluerons la qualité des suites aléatoires obtenues grâce aux TRNG complets basés sur ces deux extracteurs et différentes sources d'aléa. La qualité des suites aléatoires sera évaluée avec la suite de tests statistiques NIST SP800-22.

2 Extracteur physique d'aléa basé sur l'accumulation du jitter à l'aide de compteurs

2.1 Introduction et description

Comme nous l'avons montré dans le Chapitre 1, le principe proposé par Fairfield *et al.* [FMC85] est largement repris et réutilisé. Cependant comme l'ont remarqué les auteurs, il existe un certain nombre de problèmes dans ce générateur. En effet tous les principes qui sont inspirés de ce moyen de collecter l'aléa fournissent des suites

aléatoires biaisées, même avec un grand nombre d'oscillateurs en parallèle [SMS07]. De plus le fait qu'à la sortie il existe un mélange de bits pseudo-aléatoires et de bits vraiment aléatoires est un inconvénient pour estimer effectivement la qualité de cet extracteur comme cela a été montré dans [BBF09].

Nous allons proposer un principe d'extraction d'aléa très similaire basé sur l'accumulation du jitter qui s'affranchit en très grande partie de ces inconvénients. En effet au lieu d'utiliser une bascule d'échantillonnage en mélangeant ainsi des bits aléatoires et des bits pseudo aléatoires, nous proposons d'utiliser un compteur dont on peut facilement séparer les bits aléatoires des bits non aléatoires ou constants. Ainsi nous pouvons facilement évaluer seulement la qualité des bits vraiment aléatoires.

Le principe de fonctionnement est présenté à la Figure 3.1. Ce principe est très

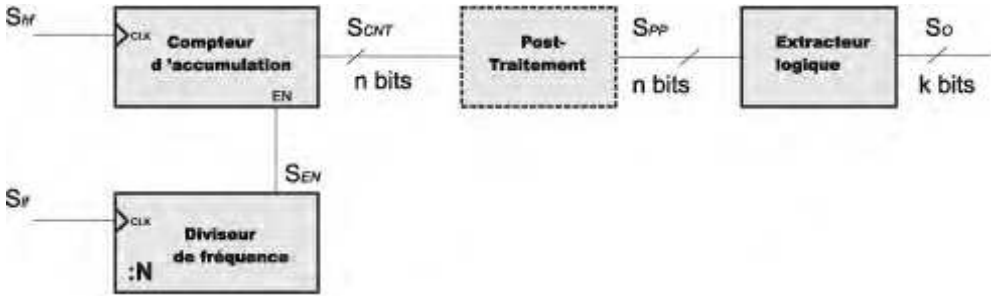

Figure 3.1 Principe de fonctionnement de l'extracteur physique d'aléa à base de compteurs.

similaire à celui présenté par Fairfield *et al.* avec deux signaux qui contiennent du jitter. Toutefois il présente l'avantage de présenter à sa sortie uniquement du vrai aléa et non pas un mélange de pseudo-aléa et de vrai aléa. On retrouve un premier signal de fréquence élevée S_{hf} et un signal de fréquence basse S_{lf}. Le signal S_{hf} cadence un compteur de n bits. Ce compteur est contrôlé par un signal S_{EN} qui autorise ou réinitialise le compteur selon sa valeur. Le signal S_{EN} est lui même généré par un diviseur de fréquence. Ainsi la valeur du compteur est évaluée à intervalles réguliers fixés par ce diviseur de fréquence. Si la fréquence de S_{hf} est suffisamment élevée et si ce signal présente un jitter qui s'accumule, comme cela a été montré dans le Chapitre 2, alors la valeur du compteur présentera des variations aléatoires de la valeur du compteur à la fréquence de S_{EN}. Ces variations seront d'autant plus grandes que le temps d'accumulation est plus important.

Ainsi le signal S_{CNT} est un signal de n bits dont les valeurs changent à intervalles réguliers de manière aléatoire. Ce signal n'est pas encore une suite de bits aléatoires uniformément distribués. C'est le rôle de l'extracteur logique de transformer le signal S_{CNT} en une suite de bits uniformément distribués. Ainsi l'extracteur prendra en entrée n bits et fournira en sortie k bits aléatoires uniformément distribués. Cette opération d'extraction peut être réalisée de différentes manières et avec différents niveaux d'efficacité. Pour cette évaluation, nous allons utiliser comme fonction d'ex-

traction le cas le plus simple où seul le bit de poids faible (LSB) du signal S_{CNT} est conservé. Ainsi dans le reste de ce chapitre k sera fixé à 1 bit. Il est toutefois possible, si l'entropie de la suite le permet, d'extraire plus que un bit aléatoire. Toutefois si les k bits de poids faible (LSB) sont conservés il est important de s'assurer que ils ne présentent pas de corrélation à long terme. Nous avons observé que si l'oscillateur à

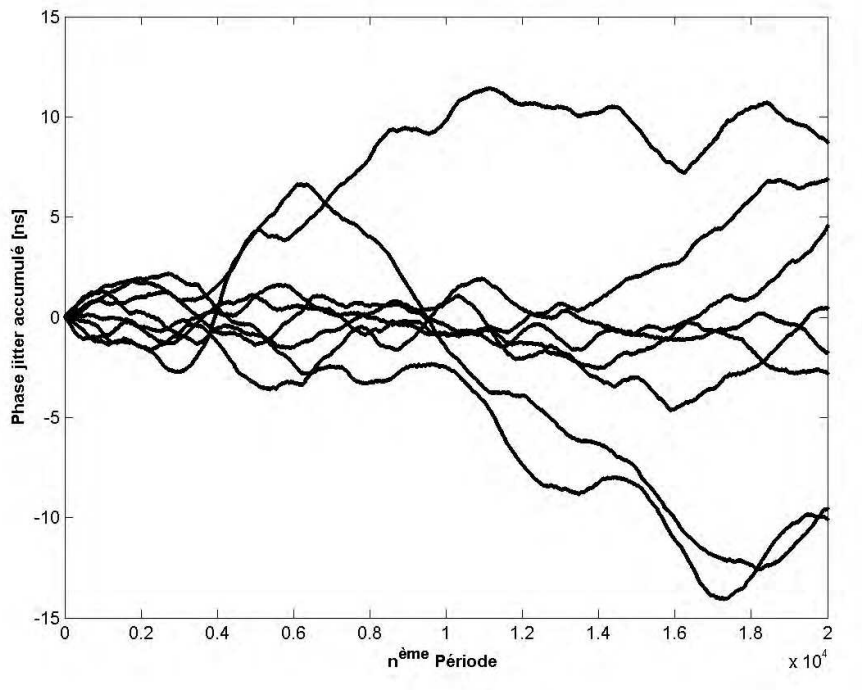

Figure 3.2 Mise en évidence du phénomène de l'accumulation du jitter. Données expérimentales, oscillateur en anneau de 3 éléments fonctionnant à 5,266 ns.

l'origine du signal S_{hf} est en oscillation libre, l'accumulation se fait proportionnellement à la racine carrée du temps à court terme puis linéairement à long terme. Pour l'instant nous allons retenir que l'accumulation du jitter augmente avec le temps à condition que l'oscillateur soit en oscillation non contrôlée et donc que l'amplitude du *phase jitter* ne soit pas bornée.

Pour illustrer ce phénomène d'accumulation, nous avons observé de façon expérimentale, l'accumulation du jitter d'un oscillateur en anneau de 3 éléments implanté dans une cible Xilinx Spartan 3. La période moyenne de ce signal étant de 5,266 ns. Nous avons représenté à la Figure 3.2 neuf réalisations du signal jitté accumulé pendant 20000 périodes. Comme on peut le constater, l'accumulation est différente pour chaque réalisation et si elle est grande devant la période du signal alors elle pourrait s'exprimer comme des variations de la valeur d'un compteur que nous aurions laissé tourner pendant un temps fixe. Plus ce temps est grand plus l'accumulation sera grande et les variations des valeurs de compteur seront importantes.

Les conditions exactes de fonctionnement ainsi qu'un modèle précis de ce phénomène seront décrits dans le Chapitre 4. Pour le moment, nous nous intéresserons au principe de fonctionnement et aux résultats expérimentaux de cet extracteur physique.

L'impact d'un post-traitement simple sera évalué dans ce chapitre. Nous verrons que son utilisation n'est pas nécessaire pour avoir des signaux aléatoires de bonne qualité cependant il contribue fortement à rendre les signaux aléatoires stationnaires et donc plus facilement modélisables. Dans ce chapitre, l'utilisation de ce post-traitement sera toutefois optionnelle.

Le signal S_{CNT} est un signal numérique de n bits à fréquence fixe, dont la nature est aléatoire. Il est donc aisé d'utiliser des fonctions déterministes connues pour en améliorer les qualités statistiques. Des opérations tels que le filtrage numérique peuvent en effet corriger une grande partie des défauts dus aux perturbations déterministes notamment.

Ainsi le bloc de post-traitement remplit la fonction de filtre passe haut donnée par l'équation :

$$S_{PP}(n) = S_{CNT}(n) - S_{CNT}(n-1) \tag{3.1}$$

2.2 Résultats expérimentaux

Il est très important de noter ici que les résultats présentés dans ce chapitre sont obtenus dans des conditions idéales de fonctionnement du FPGA : c'est à dire avec seulement le TRNG et la logique de contrôle implantés dans le circuit. Dans de telles conditions, le jitter s'il est issu d'un oscillateur en anneau, présentera un comportement idéal avec des éventuelles composantes déterministes très faibles (comme nous l'avons montré dans la Section 4 du Chapitre 2). Cependant le jitter d'un oscillateur en anneau sera très sensible au fonctionnement du reste du circuit.

Il est toutefois très important de décrire le comportement dans un tel cas idéal pour plusieurs raisons. On peut imaginer un scénario où le circuit est volontairement mis dans un état non perturbatif pendant la phase de production de suites binaires aléatoires. De plus les deux extracteurs présentés ici peuvent fonctionner avec différentes sources de signaux jittés et leur utilisation n'est pas limitée aux signaux produits par un oscillateur en anneau. Ainsi si la source est un oscillateur embarqué comme l'oscillateur RC de la famille Actel ou une PLL avec un signal de référence stable, la sensibilité à la logique perturbatrice s'en retrouvera fortement diminuée.

Nous notons également que tous les articles qui présentent des TRNG basés sur le jitter dans les oscillateurs en anneau comme dans les autres sources d'aléa généralement utilisés (métastabilité) présentent les résultats dans de telles conditions idéales. A notre connaissance le problème de la sensibilité du jitter, dans un oscillateur en anneau,

à une logique perturbatrice interne au circuit FPGA n'est soulevée que dans [San09].

Pour illustrer cet extracteur physique d'aléa et cette fonction de post-traitement, nous avons implanté un oscillateur en anneau de 10 éléments dans une cible Altera Cyclone III. L'oscillateur en anneau a été implanté sans tenir compte des interconnexions et sans contraintes de placement routage, en laissant l'outil de synthèse placer les cellules. Le signal S_{lf} est obtenu à l'aide de l'oscillateur –98NG de 16Mhz disponible sur la carte Micronic et dont le profil de jitter a été présenté dans la Section 5 du Chapitre 2. Nous avons représenté sur la Figure 3.3 trois cas de fonctionnement de cet extracteur pour différentes valeurs de N du diviseur de fréquence. Sur cette Figure sont représentés 50 000 valeurs du signal S_{CNT} ainsi que les histogrammes de ces valeurs respectives. Comme on peut le voir, plus le temps d'accumulation fixé par N est grand plus des composantes basses fréquences apparaissent dans le signal S_{CNT} rendant la moyenne de ce signal variable dans le temps et donc le signal non stationnaire. Ces fréquences basses entraînent une asymétrie dans l'histogramme des valeurs et donc un biais. En effet, l'extraction étant faite sur le LSB pour avoir un biais aussi proche que possible de 0, l'histogramme doit présenter autant de nombres pairs que de nombres impairs. Or cette condition est remplie lorsque l'histogramme est très proche d'une Gaussienne et parfaitement symétrique.

Sur la Figure 3.4 est représenté le signal S_{PP} après le post-traitement décrit dans la Section 2.1 de ce chapitre. On remarque que les composantes basses fréquences sont filtrées par la fonction de post-traitement. Les histogrammes sont parfaitement symétriques. Pour mieux illustrer l'effet de cette fonction de post-traitement, nous avons représenté à la Figure 3.5 les signaux S_{CNT} et S_{PP} pour le cas N=10000. Cette Figure montre les données expérimentales vis-à-vis d'une loi Normale en pointillée sur la Figure. Il s'agit de la représentation d'une droite de Henry réalisée par la fonction *normplot* de Matlab. Nous voyons clairement que la distribution du signal S_{CNT} s'écarte de la loi normale. Le signal S_{PP} suit une loi normale dans une large plage de probabilités. Seul quelques échantillons extrêmes de très faible probabilité (<0.0001) s'écartent de la loi Normale. Ce résultat est important pour deux raisons. Il améliore la symétrie de la distribution mais également le fait que le signal S_{CNT} suive un comportement Gaussien sera précieux pour la modélisation du TRNG en rendant le signal S_{CNT} stationnaire.

Une synthèse de cet extracteur d'aléa avec et sans fonction de post-traitement est présentée dans les Tableaux 3.1 et 3.2. Dans ce tableau, nous avons évalué les paramètres statistiques des signaux S_{CNT} et S_{PP} tels que les valeurs moyennes et les écarts-types des valeurs de compteurs. De plus, le biais est également calculé avec une extraction simple par le bit de poids faible. Comme on peut le constater celui ci est amélioré par la fonction de post-traitement bien qu'il soit déjà suffisamment faible sans l'utilisation de celle-ci.

Nous nous sommes également intéressés à l'entropie véhiculée par la suite de nombres de n bits du signal S_{CNT} et S_{PP} ainsi que de la suite binaire S_O. Une définition

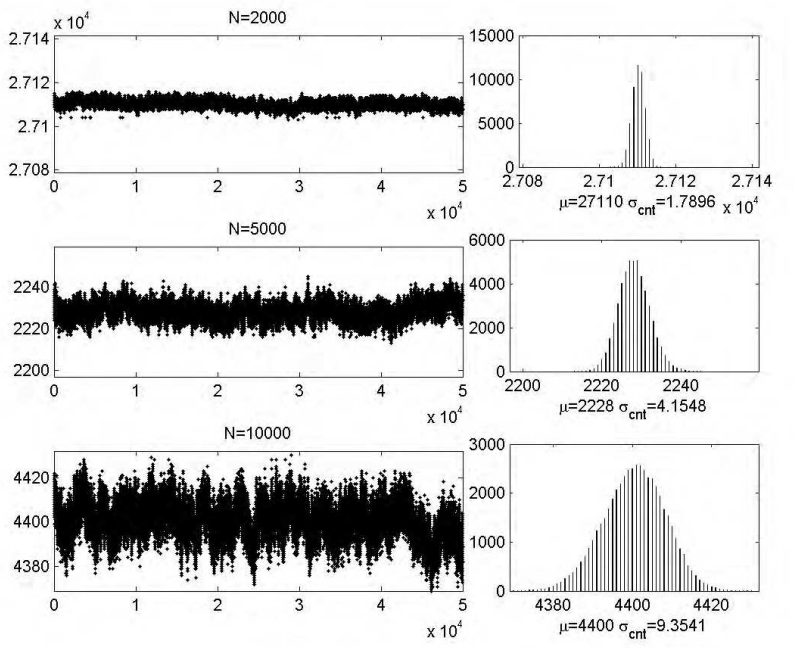

Figure 3.3 Signaux numériques aléatoires issus de l'extracteur physique après différents temps d'accumulation.

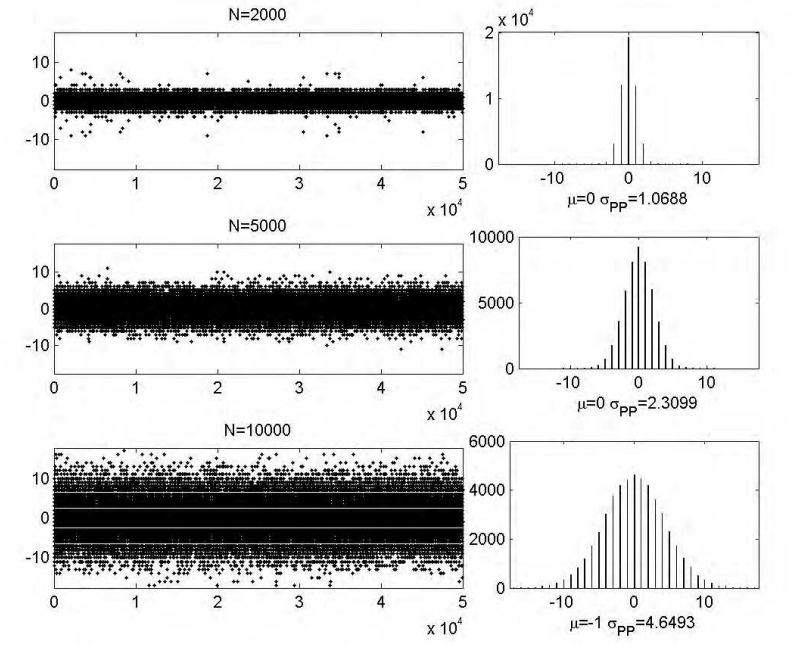

Figure 3.4 Signaux numériques aléatoires filtrés (post-traitement) issus de l'extracteur physique après différents temps d'accumulation.

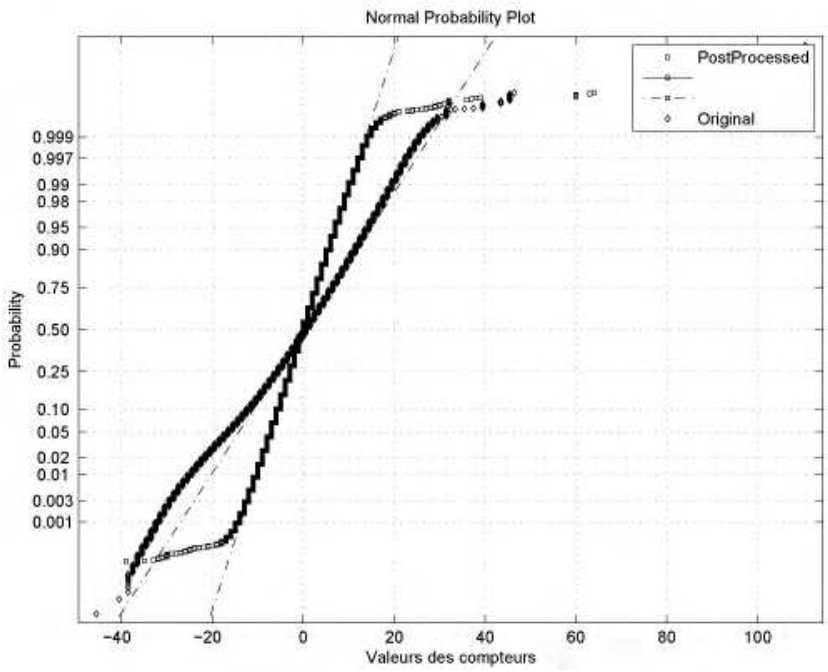

Figure 3.5 Mise en évidence de l'intérêt du post-traitement. Droite de Henry sur les signaux aléatoires avant et après post-traitement.

N	$\mu_{S_{CNT}}$	$\sigma_{S_{CNT}}$	$B(S_O)$	$H(S_{CNT})$	$H(S_O)$
2000	27110	1.789	0.00421	2.886	0.999948
5000	2228	4.154	0.00472	4.091	0.999935
10000	4496	9.354	0.00533	5.246	0.999917

Tableau 3.1 Synthèse des caractéristiques statistiques de suites aléatoires sans post-traitement, obtenues par l'extracteur physique d'aléa à base de compteurs après différents temps d'accumulation. Population de 204K échantillons.

N	$\mu_{S_{PP}}$	$\sigma_{S_{PP}}$	$B(S_O)$	$H(S_{PP})$	$H(S_O)$
2000	0	1.068	0.00706	2.125	0.999855
5000	0	2.309	0.00211	3.204	0.999987
10000	0	4.649	0.00126	4.177	0.999995

Tableau 3.2 Synthèse des caractéristiques statistiques de suites aléatoires avec post-traitement, obtenues par l'extracteur physique d'aléa à base de compteurs après différents temps d'accumulation. Population de 204K échantillons.

et une discussion sur l'utilisation de l'entropie seront présentées dans la Section 2.3 du Chapitre 4. Elle est ici calculée sur un ensemble $\Omega = \{0...2^n - 1\}$ pour les signaux de n bits S_{CNT} et S_{PP} (ici $n = 8$) et sur un ensemble $\Omega = \{0, 1\}$ pour le signal S_O après extraction logique. Il est important de noter ici que l'entropie est calculée sur une population expérimentale donnée. Comme on peut le constater dans le Tableau 3.1 cette entropie se retrouve abaissée par la fonction de post-traitement à cause du

phénomène de filtrage. Cependant puisque l'extraction finale de bits aléatoire est ici uniquement de $k = 1$ bit, nous disposons de suffisamment d'entropie même après la fonction de post-traitement.

2.3 Propriété de *scalabilité* de l'extracteur à base de compteurs

Il est évident que ce type d'extraction basé sur l'accumulation du jitter fournit un débit faible. En effet, l'accumulation devant se faire sur un nombre important de périodes basses fréquences du signal S_{lf}, le débit est de l'ordre du Kbs. Cependant pour le cas particulier où le signal S_{hf} est obtenu par un oscillateur en anneau, nous avons évalué la possibilité d'accumuler du jitter de plusieurs oscillateurs en anneau en même temps. Cela reviendrait à implanter plusieurs fois le TRNG avec différents signaux $S_{hf_{1...n}}$ et un seul et unique signal S_{lf}. Ainsi le temps d'accumulation est identique mais les sources de jitter accumulé sont différentes.

N	$H(S_{CNT_1})$	$H(S_{CNT_2})$	$Mi(S_{CNT_1}, S_{CNT_2})$	$\rho(S_{CNT_1}, S_{CNT_2})$	$Mi(S_{OUT_1}, S_{OUT_2})$
5000	4.31	4.13	0.019	0.12	$5.16\ 10^{-7}$
5000PP	3.22	3.106	0.041	0.22	$7.89\ 10^{-7}$
10000	4.84	4.82	0.074	0.28	$5.531\ 10^{-6}$
10000PP	3.88	3.74	0.042	0.21	$4.453\ 10^{-6}$

Tableau 3.3 Synthèse des caractéristiques statistiques de suites aléatoires obtenues à l'aide de deux extracteurs à base de compteurs en parallèle avec et sans post-traitement. Population de 204K échantillons.

Nous présentons ici de manière empirique l'accumulation de deux oscillateurs en anneau de 10 éléments implantés dans une cible Cyclone III (Micronic) et fonctionnant en même temps. Comme pour les données de la Section 2.2 de ce chapitre, le placement-routage n'est pas spécifié et est laissé au synthétiseur. Les données des signaux S_{CNT_1} et S_{CNT_2} avant post-traitement sont présentés à la Figure 3.6 et avec post-traitement à la Figure 3.7. Nous constatons que les deux signaux issus des deux compteurs suivent bien des tendances différentes. Nous avons de plus représenté à la Figure 3.6 et 3.7 un tracé en XY des deux signaux S_{CNT_1} et S_{CNT_2}. Nous pouvons constater que les deux valeurs ne semblent pas présenter une corrélation. Pour nous en convaincre, nous avons également calculé *le coefficient de corrélation* ρ de ces deux signaux ainsi que *l'information mutuelle*. Ces deux outils caractérisent la dépendance entre les deux suites aléatoires fournies par les deux compteurs et seront introduits dans la Section 2 du Chapitre 4. Le facteur de corrélation ρ est représentatif de la dépendance linéaire (ou corrélation) : sa valeur est comprise entre 0 et 1. L'*information mutuelle* quant à elle traduit la dépendance statistique dans un sens plus large et est exprimée en bits.

Les résultats sont présentés dans le Tableau 3.3. Comme on peut le constater, les coefficients de corrélation ainsi que l'information mutuelle sont faibles aussi bien dans le cas avec post-traitement que dans le cas sans post-traitement traduisant ainsi un

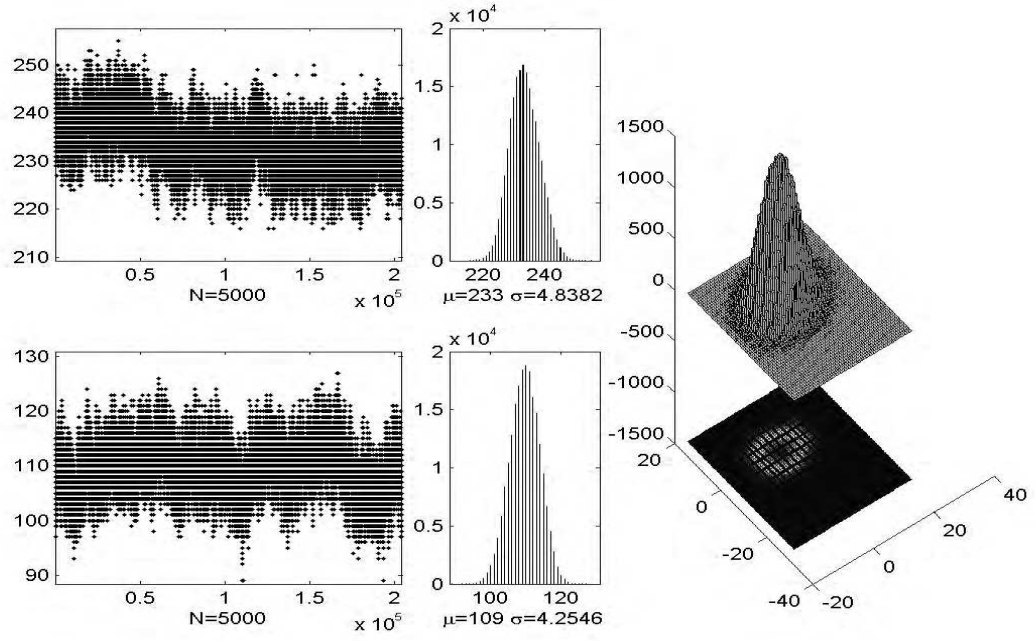

Figure 3.6 Signaux aléatoires non post-traités issus de l'accumulation de deux oscillateurs en anneaux fonctionnant en même temps.

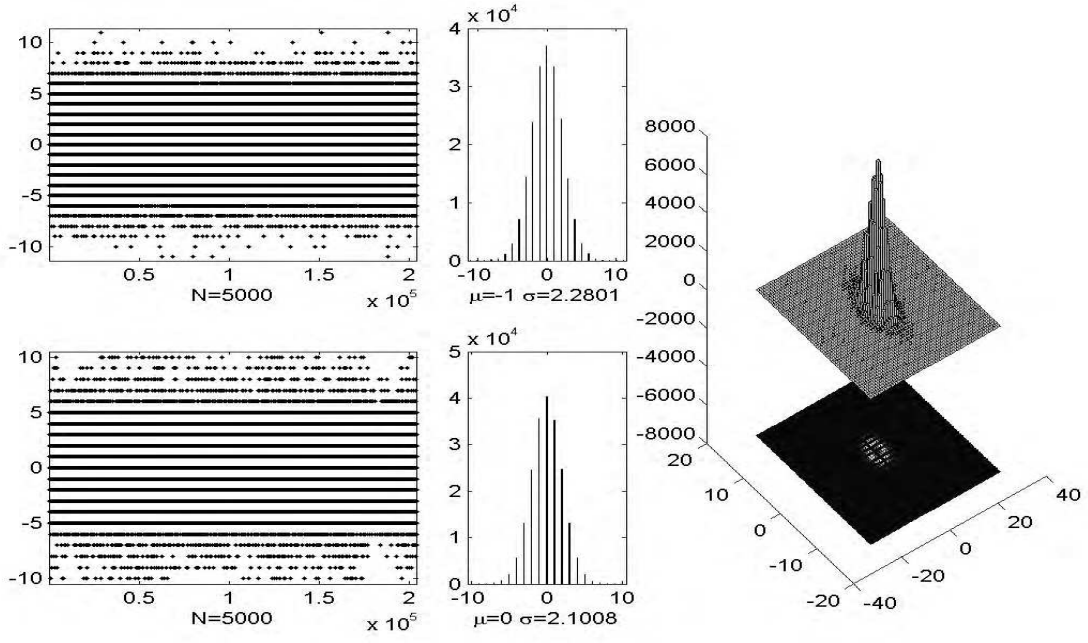

Figure 3.7 Signaux aléatoires post-traités issus de l'accumulation de deux oscillateurs en anneaux fonctionnant en même temps.

comportement qui tend à être indépendant.

Pour en être certains et pour rechercher d'éventuelles corrélations à long terme, nous avons étudié également la fonction de corrélation. Cette fonction consiste à calculer le coefficient de corrélation entre les deux signaux pour différents temps de retard entre les deux signaux (*time lags*). Les fonctions de corrélation sont présentées à la Figure 3.8 pour les signaux avant et après post-traitement. Comme on peut le voir sur cette Figure, la valeur de ρ est maximale pour un retard de 0, c'est à dire au même moment. Ainsi nous pouvons conclure qu'il n'existe pas (ou peu) de corrélation à long terme entre les deux signaux issus de deux compteurs fonctionnant en même temps. Pour faire les calculs de la fonction de corrélation nous avons utilisé la fonction *crosscorr* de Matlab.

Il est intéressant de noter que le coefficient de corrélation est plus important pour les signaux après post-traitement. Nous n'avons pas approfondi cette question cependant nous pensons que cela est du au fait que pour les signaux non post-traités les composantes basses fréquences aléatoires contribuent à la non corrélation des signaux. Nous remarquons également que les coefficients de corrélation semblent moins importants sans l'emploi de post-traitement dans le cas N=5000. Cependant ces coefficients de corrélation ainsi que l'information mutuelle restent faibles.

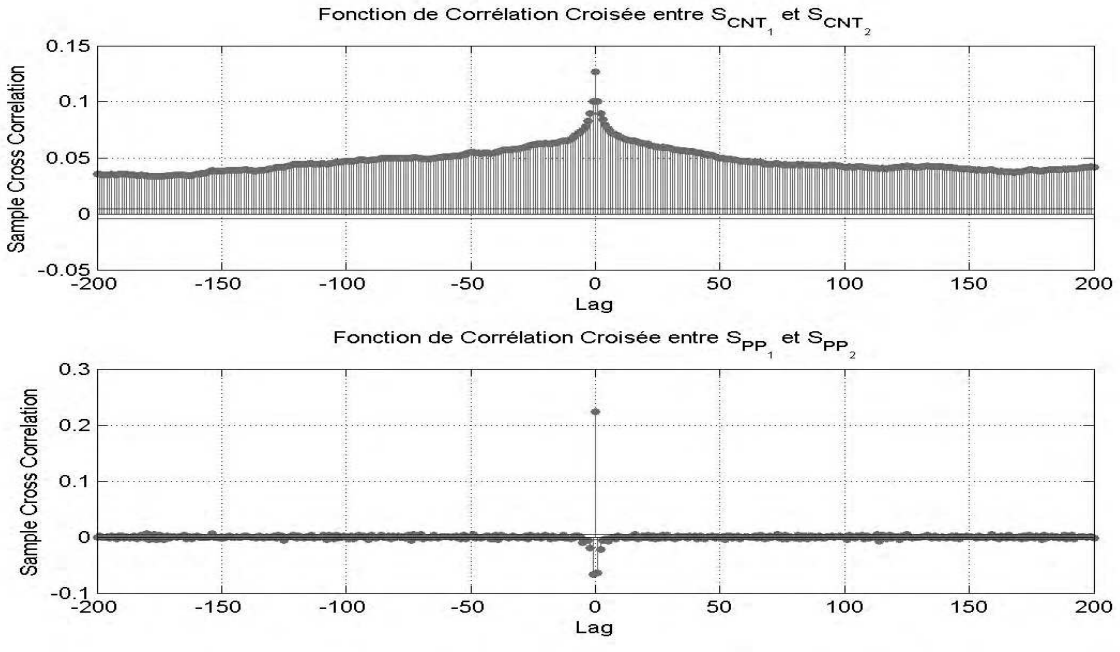

Figure 3.8 Fonction de corrélation des signaux issus de deux compteurs fonctionnant en même temps avant et après post-traitement.

2.4 Conclusion

Cet extracteur physique d'aléa est très intéressant car il fournit un signal numérique de n bits de nature aléatoire à une fréquence fixe. Ce signal peut être facilement caractérisé et éventuellement traité pour obtenir un signal aléatoire Gaussien. Ce signal présente l'avantage par rapport à d'autres TRNG de ne pas tendre vers une suite de bits uniformément distribués mais plutôt vers une loi Gaussienne. En jouant sur le temps d'accumulation on peut ainsi obtenir plus ou moins de bits vraiment aléatoires (entropie) autour d'une moyenne fixe. Ainsi il est aisé de prévoir suffisamment d'entropie avant l'extraction de la suite aléatoire uniformément distribuée afin d'éviter l'utilisation d'un correcteur ou d'un brouilleur.

Un autre avantage de cet extracteur physique d'aléa est sa simplicité, tant au niveau architecture qu'au niveau fonctionnement. De ce fait, il est intrinsèquement implantable dans n'importe quelle cible FPGA.

Nous notons également qu'une architecture très semblable à celle proposée dans cette thèse a été exposée dans [JRH10] lors du dernier *Workshop CryptArchi*. Le principe est identique, cependant les auteurs de [JRH10] ont implanté leur TRNG dans un micro-contrôleur et non pas dans un FPGA. Cette implantation avec des signaux diffère de celle présentée ici par l'extraction logique de l'aléa et la nature de la cible. En effet sur le micro-contrôleur le temps d'accumulation n'est pas fixe mais variable en fonction de l'activité du micro-contrôleur induisant ainsi un comportement différent de celui présenté ici. Nous avons basé nos travaux présentés dans cette section à partir d'une méthode de mesure du jitter en interne publiée dans [VABF08].

3 Extracteur physique d'aléa à base d'échantillonnage cohérent

Les résultats obtenus grâce à l'extracteur à base de compteurs sont très satisfaisants sur le plan de la génération d'aléa, cependant le débit que l'on peut obtenir reste faible. Nous avons donc cherché un autre moyen d'obtenir un signal aléatoire sur n bits pour extraire une suite binaire aléatoire uniforme. Le principe retenu est celui de l'échantillonnage cohérent. Avec ce principe le débit est augmenté et atteint le Mbs. De plus le biais obtenu reste très faible sans besoin de fonction de correction ou de post-traitement complexe.

3.1 Description du principe

L'échantillonnage cohérent est une technique d'échantillonnage que l'on peut utiliser pour échantillonner des signaux périodiques à une très grande fréquence virtuelle. Cette technique est notamment utilisée dans les oscilloscopes à échantillonnage équivalent (*équivalent time sampling oscilloscopes*). Le principe est simple, l'élément d'échan-

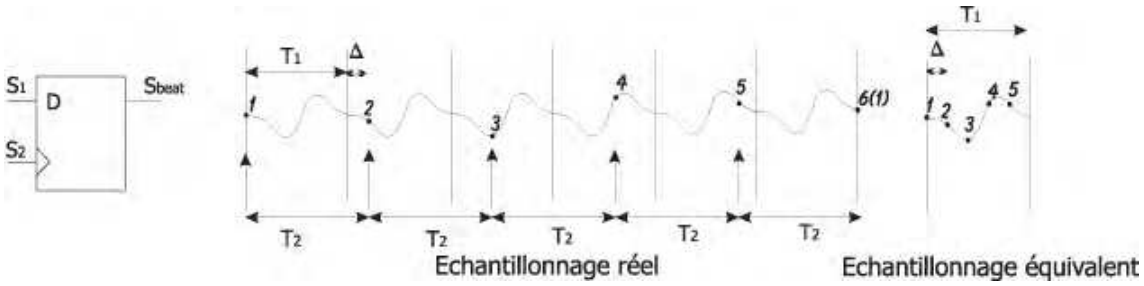

Figure 3.9 Principe de l'échantillonnage cohérent pour des signaux périodiques.

tillonnage est toujours une bascule DFF cependant les deux signaux ont des fréquences et donc des périodes très proches. Le principe est décrit à la Figure 3.9. Soit S_1 et S_2 deux signaux de périodes T_1 et T_2 très proches :

$$T_1 = T_2 + \Delta \tag{3.2}$$

Ainsi comme on peut le voir sur la Figure 3.9, si Δ est très petit devant T_1 et T_2 nous aurons un échantillonnage virtuel à une fréquence très élevée de Δ^{-1} Hz. Comme nous le voyons sur cette Figure, l'échantillonnage se produit à des intervalles réguliers de Δ secondes et ce dans un ordre consécutif. Puisque les signaux sont périodiques il sont donc identiques d'une période à l'autre. Il s'en suit qu'à la sortie de la bascule DFF sera présent un signal qui sera une copie basse fréquence du signal T_1. La fréquence sera d'autant plus basse que Δ est petit, tandis que la fréquence d'échantillonnage virtuelle sera d'autant plus grande. Nous appellerons ce signal à la sortie de la bascule DFF, signal de battement et nous le noterons S_{Beat}. La période de ce signal peut être exprimée par :

$$T_{Beat} \approxeq \lfloor \frac{T_1}{\Delta} \rfloor \times T_2 \tag{3.3}$$

Si les signaux S_1 et S_2 présentent des variations aléatoires de leurs périodes respectives, exprimées sous la forme de jitter, alors la période du signal T_{Beat} présentera aussi des variations de sa période basse fréquence. Cependant ces variations seront des multiples de la période T_2. Le signal à la sortie de la bascule sera une succession d'état hauts et d'états bas ininterrompus à condition que Δ ne soit pas trop petit devant l'amplitude du jitter dans le signal S_1. Nous supposerons et nous validerons expérimentalement dans ce chapitre le fait que $\Delta > \Delta_{critique}$ et que le signal S_{Beat} soit bien une suite ininterrompue d'états bas et hauts.

Nous pouvons remarquer ici qu'il est possible d'échantillonner le signal de période la plus petite par celui de période plus grande et vice-versa. En effet la seule différence sera l'ordre des échantillons constituants du signal T_{Beat}. Nous proposons donc d'utiliser le principe d'échantillonnage cohérent pour réaliser un extracteur physique d'aléa. Le schéma de principe de cet extracteur d'aléa est donné à la Figure 3.10. Il s'agit d'une version d'étude de cet extracteur. Le signal de battement S_{Beat} est introduit dans une *machine d'état* qui incrémente la valeur d'un compteur de n bits à chaque

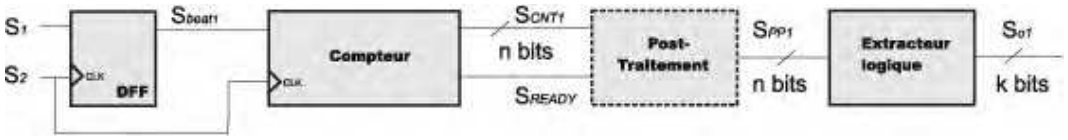

Figure 3.10 Extracteur physique d'aléa basé sur l'échantillonnage cohérent.

front montant du signal S_2. La valeur du compteur est bloquée dans un registre puis le compteur est remis à zéro à chaque nouvelle période du signal T_{Beat}. A chaque nouvelle valeur de compteur correspondant à chaque période T_{Beat}, le signal S_{READY} présente une impulsion synchronisée avec le signal d'horloge du système pour faciliter la gestion des valeurs du signal S_{CNT} en aval. Dans cet exemple concret et dans l'ensemble de la thèse, ces valeurs sont transmises par USB sur un PC pour être analysées.

Ainsi nous disposons d'une population de valeurs sur n bits à une fréquence moyenne de la fréquence du signal S_{Beat} définie par l'équation 3.3. Il est important de noter que c'est une fréquence moyenne et que dans le temps absolu les valeurs viennent à des temps variables multiples de T_2. Ce qui explique l'intérêt d'avoir un signal S_{READY}. Ce fonctionnement est illustré par un chronogramme sur la Figure 3.11 où sont représentés dans l'ordre descendant les signaux S_1, S_2, S_{Beat}, S_{READY}, S_{CNT1} et l'horloge système. Pour illustrer ce phénomène, nous avons implanté dans une cible Altera Cyclone III

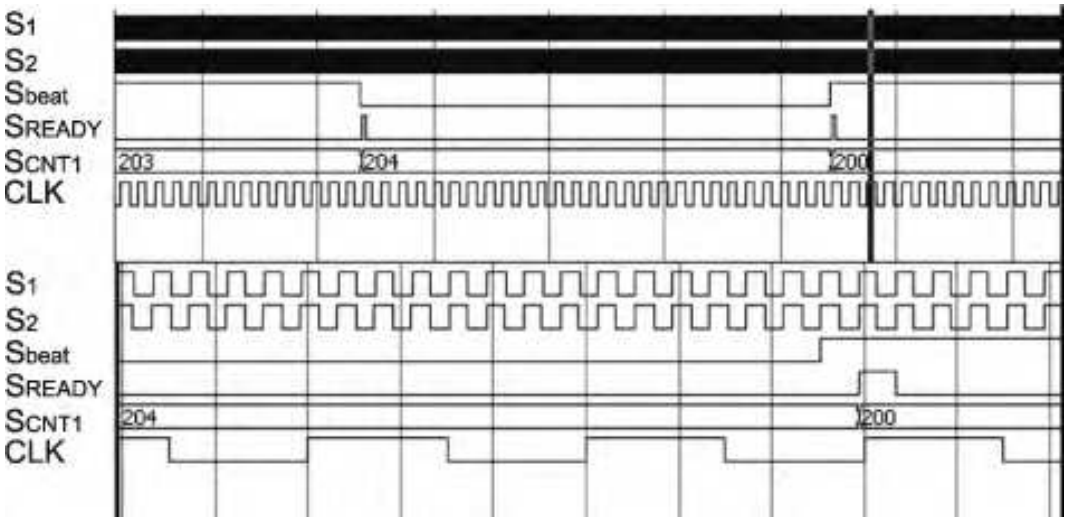

Figure 3.11 Chronogramme de fonctionnement de l'extracteur physique d'aléa basé sur l'échantillonnage cohérent.

(Micronic) deux oscillateurs en anneau de 10 éléments avec un routage semblable pour obtenir deux signaux de fréquence proches. Pendant cette phase d'implantation, nous avons veillé à ce que les oscillateurs ne soient pas proches l'un de l'autre pour éviter tous risques d'interférences. Il est aisé d'obtenir des différences de quelques centaines de pico-secondes dans les périodes moyennes des deux signaux de cette manière. Cependant, cette manière présente l'inconvénient d'être spécifique à une puce donnée et

non-reproductible sans intervention manuelle de placement-routage sur une autre puce. Nous étudierons d'autres manières d'obtenir des signaux de fréquences proches dans la Section 3.4 de ce chapitre. Les signaux ainsi obtenus présentent des périodes moyennes de $T_1 = 5,040ns$ et $T_2 = 5,020ns$ obtenant ainsi un Δ de 20ps environ. Ces mesures ont été effectuées en externe sur un oscilloscope. Nous avons représenté sur la Figure 3.12 le signal S_{CNT} représentant les variations de la période du signal T_{Beat}. Nous pouvons constater que la déviation standard est importante et symétrique. Le Tableau 3.4 donne les valeurs de biais ainsi que d'entropie obtenues dans cet exemple. Comme on peut le constater l'écart type est très important pour un débit moyen de 0.8 Mbs. Ainsi à cette fréquence, il est possible d'extraire au moins un bit aléatoire (en prenant le bit de poids faible par exemple). Le résultat du calcul de l'entropie contenue dans la suite expérimentalement obtenue est de plus de 4 bits aléatoires.

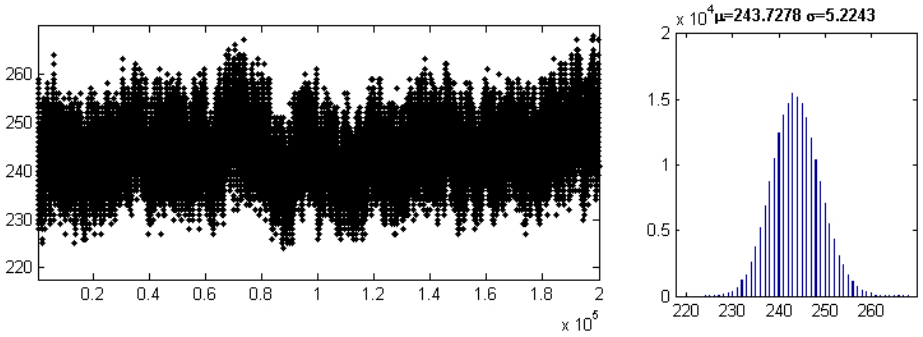

Figure 3.12 Signaux aléatoires obtenus à l'aide de l'extracteur physique d'aléa basé sur l'échantillonnage cohérent. S_1 et S_2 obtenus à l'aide de deux oscillateurs en anneau de 10 éléments dans Cyclone III. $T_1 \approx 5,040ns$, $T_{Beat} \approx 1.25us$.

S	μ_S	σ_S	$H(S)$	$B(mod(S,2))$
S_{CNT}	242.57	5.92	4.61	0.00049

Tableau 3.4 Caractéristiques statistiques des suites aléatoires obtenues par l'extracteur physique d'aléa basé sur l'échantillonnage cohérent. Population 2M échantillons.

3.2 Double extraction

Nous avons étudié la possibilité d'extraire au moins deux bits par période du signal T_{Beat}. Pour ce faire le bloc compteur de la Figure 3.10 a été configuré de manière à remettre le compteur à zéro en sauvegardant sa valeur dans un registre à chaque changement d'état du signal T_{Beat} et non pas à chacune de ses périodes. Comme dans la section précédente, les deux signaux sont obtenus avec des oscillateurs en anneau de 10 éléments, présentent des périodes moyennes de $T_1 = 5,200ns$ et $T_2 = 5,220ns$ avec un Δ de 20ps environs. Les valeurs successives des compteurs correspondant aux demies périodes successives du signal T_{Beat} sont représentées à la Figure 3.13. Comme nous pouvons le voir, la déviation standard est suffisante et la distribution suffisamment large

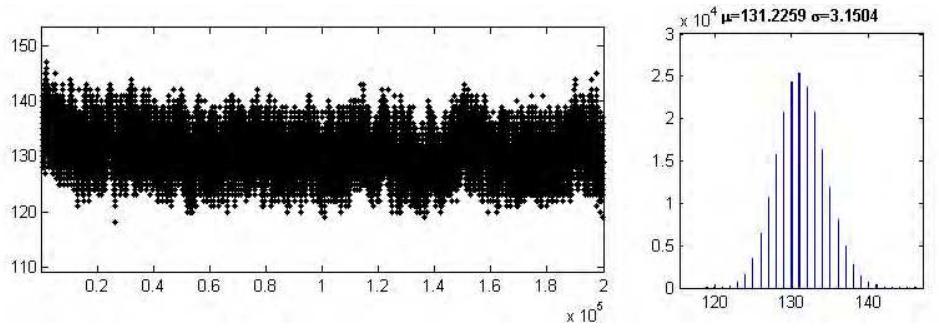

Figure 3.13 Mise en évidence du phénomène de double extraction dans Cyclone III, $S_1 \approx 5,200ns$, $\Delta \approx 20ps$, $T_{Beat}/2 \approx 680ns$.

S	μ_S	σ_S	$H(S)$	$B(mod(S,2))$
S_{CNT}	131.22	3.15	3.92	1.95×10^{-4}

Tableau 3.5 Synthèse des caractéristiques statistiques de suites aléatoires obtenues à l'aide de l'extracteur physique d'aléa à double échantillonnage cohérent. Population de 2M échantillons.

pour pouvoir extraire au moins un bit aléatoire toutes les demies périodes doublant ainsi le débit que l'on peut obtenir avec cet extracteur. Les résultats sont présentés dans le Tableau 3.5. Comme on peut le constater le biais reste très faible.

3.3 Échantillonnage mutuel

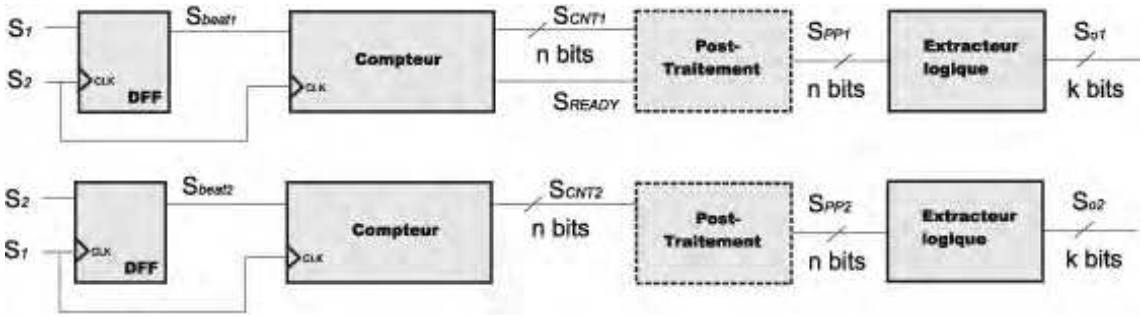

Figure 3.14 Principe de l'échantillonnage mutuel.

Dans un second temps, nous avons étudié la possibilité d'augmenter par un facteur 2 le débit obtenu dans la section précédente pour arriver ainsi à extraire jusqu'à 4 bits aléatoires par période du signal T_{Beat}. Nous proposons ainsi de réaliser l'échantillonnage simultané des signaux S_2 par S_1 ainsi que S_1 par S_2. Nous avons appelé cette technique *échantillonnage mutuel* et son principe est décrit à la Figure 3.14. Dans ce cas, un seul signal S_{READY} est utilisé. Les deux signaux S_{CNT_1} et S_{CNT_2} présenterons inévitablement une forte corrélation. Cependant les signaux S_{CNT_1} et S_{CNT_2} sont suffisamment différents (seul le LSB des deux compteurs sera extrait) et cette corrélation n'a que peu d'effet

sur la suite binaire ainsi obtenue.

Pour illustrer la possibilité d'augmenter le débit par l'échantillonnage mutuel, nous avons procédé à une évaluation expérimentale. Les signaux S_1 et S_2 ont été obtenus de la même manière que dans les sections précédentes et les deux blocs compteur ont été configurés pour mesurer les périodes entières des signaux S_{Beat_1} et S_{Beat_2}. Nous avons présenté les résultats sur la Figure 3.15. Sur cette Figure, nous avons représenté seulement les 8 derniers bits des signaux S_{CNT_1} et S_{CNT_2}. Comme on peut l'observer les deux valeurs de compteurs suivent une même tendance. Cependant nous pouvons noter sur la Figure 3.15 que le graphique en XY est suffisamment dispersé et symétrique.

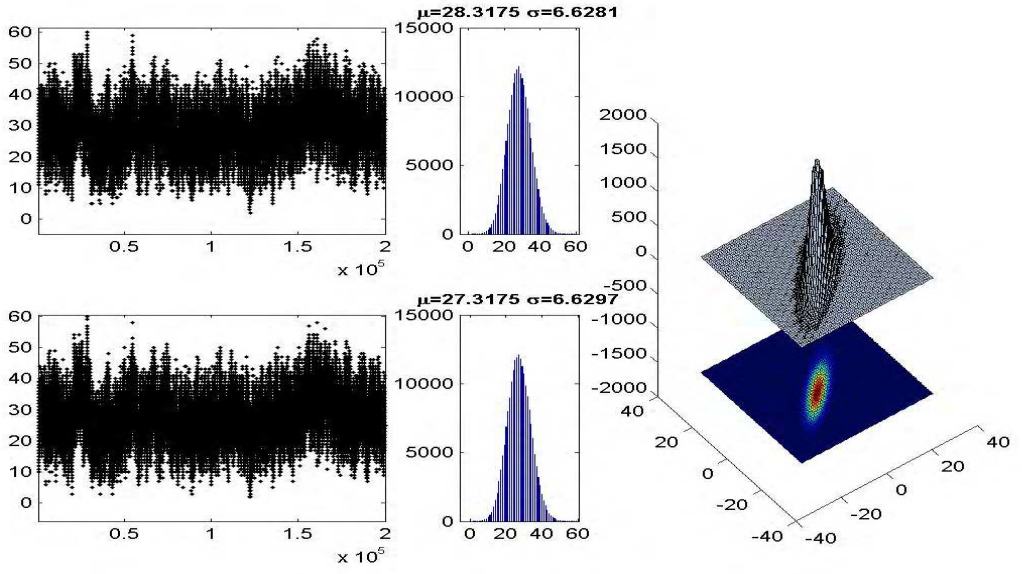

Figure 3.15 Signaux aléatoires obtenus grâce à l'échantillonnage mutuel

Nous avons représenté sur la Figure 3.16 la fonction de corrélation croisée entre les signaux issus des deux compteurs obtenus à l'aide de l'échantillonnage mutuel. Comme on peut le remarquer cette corrélation est très importante. En effet nous avons un facteur de corrélation $\rho = 0,96$. Cependant, nous remarquons sur la Figure 3.16 que la valeur de ρ est strictement décroissante en fonction du retard entre les deux signaux indiquant qu'il n'existe pas de corrélation à long terme. En fait, d'après cette Figure, la corrélation est maximale pour un retard de -1 échantillon. Il s'agit d'un artefact dû à l'architecture VHDL de l'extracteur. En effet, les registres servant à mémoriser les valeurs des deux compteurs, délivrent leurs valeurs avec un coup d'horloge de décalage ce qui explique que la corrélation est maximale pour un décalage de -1 échantillon.

Bien que la valeur du coefficient de corrélation semble très importante, elle reste

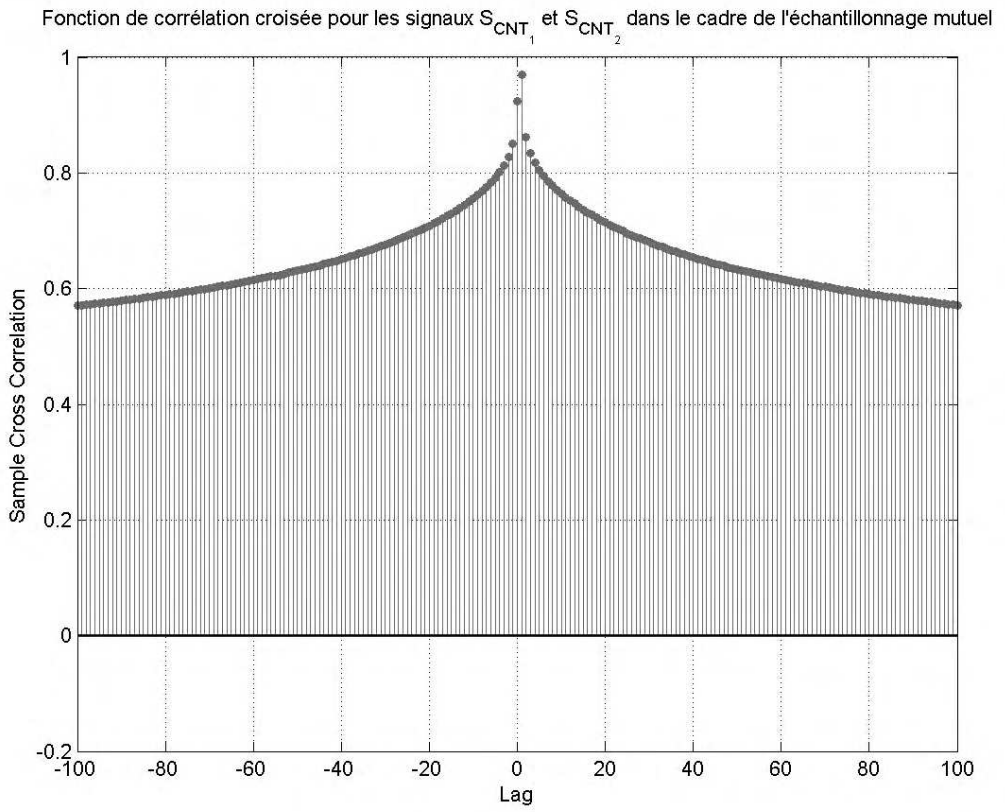

Figure 3.16 Fonction de corrélation croisée des suites aléatoires obtenues grâce à l'échantillonnage mutuel.

S	μ_S	σ_S	$H(S)$	$B(mod(S,2))$	$Mi(mod(S_{CNT_1},2),mod(S_{CNT_2},2))$
S_{CNT_1}	23.3	7.5	4.95	4.16×10^{-4}	2.95×10^{-4}
S_{CNT_2}	22.3	7.5	4.95	2.61×10^{-5}	2.95×10^{-4}

Tableau 3.6 Synthèse des caractéristiques statistiques de suites aléatoires obtenues à l'aide du l'extracteur à échantillonnage cohérent avec l'échantillonnage mutuel. Population de 2M échantillons.

inférieur à 1 et ainsi les deux suites obtenues grâce à l'échantillonnage mutuel restent suffisamment différentes l'une de l'autre pour qu'après extraction du bit de poids faible cette corrélation soit virtuellement annulée. Ce qui est validé par les valeurs de l'information mutuelle présentées dans le Tableau 3.6. De plus sur le graphique XY de la Figure 3.15 qui est réalisé pour compenser l'artefact du aux registres des deux compteurs, (décalage de -1) on peut constater que les valeurs bien que corrélées sont très différentes ce qui se traduit par une dispersion large. En effet pour une valeur en X sur ce graphique (qui correspond à une valeur du signal S_{CNT_1}) correspondent plusieurs valeurs possibles du signal S_{CNT_2}.

Nous avons calculé dans le Tableau 3.6 l'information mutuelle et le biais après

extraction logique ainsi que l'entropie, des suites obtenues grâce à l'échantillonnage mutuel. On peut donc conclure que les deux suites générées en gardant le bit LSB sont quasi indépendantes et très peu corrélées avec un biais très faible.

3.4 Obtention des signaux S_1 et S_2

Il est très important de discuter de l'obtention des signaux S_1 et S_2. En effet, comme nous l'avons montré dans le Chapitre 2, le type de jitter est différent pour chacune des sources disponibles. De plus pour cet extracteur physique d'aléa particulier la contrainte d'avoir Δ petit devant la période du signal S_1 est réalisable de différentes manières avec chacune des avantages et des inconvénients.

3.4.1 Oscillateurs en anneau

La solution la plus précise et la moins coûteuse est d'obtenir les signaux S_1 et S_2 à l'aide de deux oscillateurs en anneau de structure identique et de contrôler leur placement physique sur le circuit afin d'obtenir la différence voulue de leur période. Dans ce cas, la différence des périodes est produite par la dispersion des paramètres physiques du circuit. Il est ainsi facile, à la main, d'essayer plusieurs endroits physiques pour les oscillateurs et de choisir ceux qui donnent le Δ recherché. Cependant cette méthode présente l'inconvénient de ne pas être automatisable, en effet une intervention manuelle d'un opérateur est nécessaire.

Dans la pratique, nous avons pu obtenir facilement des écarts entre les périodes des deux signaux de quelques dizaines de pico-secondes sans difficultés dans les trois familles de FPGA. Le jitter présent dans les oscillateurs en anneau est le plus aléatoire, c'est à dire, présente le moins de composantes déterministes de toutes les sources étudiées. Par contre, comme nous l'avons montré dans le Chapitre 2, les oscillateurs en anneau sont très sensibles à la logique environnante ce qui implique qu'il faut en tenir compte lors de la conception du système cryptographique complet.

3.4.2 PLL

Une autre solution consiste à utiliser des PLL, si elles sont disponibles dans les FPGA, afin de synthétiser les deux signaux. Cette solution présente l'avantage d'être complètement automatisable et ne nécessite aucune intervention manuelle. Ainsi on peut envisager la programmation en chaîne de toute la production du système cryptographique. Un autre avantage de l'utilisation des PLL est le fait que selon la source de référence on peut obtenir une immunité à la logique perturbatrice assez importante en utilisant un oscillateur embarqué par exemple.

Cependant, avec l'utilisation des PLL, le choix du paramètre Δ se retrouve limité par les coefficients admissibles de la PLL. En effet, selon la fréquence du signal de référence et la fréquence du VCO toutes les combinaisons ne sont pas permises. De plus

les différentes PLL des différentes familles proposent des gammes de coefficients variables.

Néanmoins, comme nous allons le montrer dans la Section 4 de ce chapitre, nous avons validé la qualité des suites aléatoires générées grâce à l'extracteur à échantillonnage cohérent avec des signaux produits par des PLL dans deux familles différentes. Ainsi ce moyen d'obtention nous semble le meilleur compromis.

3.4.3 DFS

Une troisième solution consiste à utiliser des synthétiseurs de fréquence disponibles dans la famille Xilinx. Cependant cette solution présente des avantages limités. En effet les coefficients ne permettent pas de régler précisément le paramètre Δ. Pour un meilleur réglage de ce paramètre on pourrait relier en série deux DFS. Cette solution serait toutefois coûteuse en terme de ressources. De plus, comme nous l'avons montré dans la Section 7 du Chapitre 2, le jitter contenu dans les signaux générés par des DFS comporte excessivement de composantes déterministes.

3.5 Conclusion sur l'échantillonnage mutuel

Comme les résultats présentés dans le Tableau 3.6 le montrent, les deux signaux S_{CNT_1} et S_{CNT_2} bien que fortement corrélés sont capables de fournir 1 bit aléatoire indépendant chacun au moins pour former une suite aléatoire uniformément distribuée avec un biais statistique très faible sans l'aide de correcteurs ou de post traitements.

Nous allons à présent valider une implantation complète de ce TRNG dans différentes familles de FPGA avec différentes sources de signaux S_1 et S_2. Pour chaque cas nous évaluerons la possibilité d'effectuer un double échantillonnage et un échantillonnage mutuel. La validation consistera à faire passer la suite de tests statistiques NIST SP800-22 sur la suite aléatoire finale obtenue.

4 Évaluation du TRNG basé sur l'échantillonnage cohérent

Afin de valider les concepts sur l'extraction physique d'aléa introduits dans les sections précédentes, nous avons évalué les TRNG complets avec différentes sources de signaux contenants du jitter. Nous avons évalué la qualité de ces différentes versions grâce à la suite de tests statistiques NIST SP800-22. Le code utilisé était le STS2.1b accessible sur le site officiel du NIST. Pour valider l'extracteur physique d'aléa basé sur l'échantillonnage cohérent présenté dans la Section 3 de ce chapitre, nous avons utilisé l'implantation présentée à la Figure 3.17. Le signal S_{Beat} est injecté dans un compteur modulo 2 réalisé à l'aide d'une bascule T qui change d'état à chaque front montant du signal T_2. Finalement la valeur lors du front montant du signal T_{Beat} est gardée.

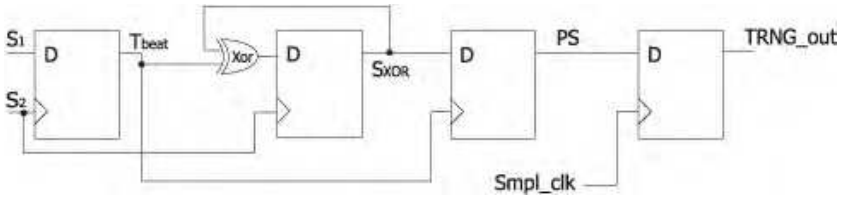

Figure 3.17 Bloc élémentaire d'un TRNG complet basé sur l'échantillonnage cohérent

Ceci est équivalent à l'extraction simple proposée dans la Section 3 de ce chapitre qui consiste à garder que le LSB du signal S_{CNT_1}.

Pour réaliser la double extraction proposée dans la Section 3.2, nous avons dupliqué le bloc de la Figure 3.17 en inversant le signal S_{Beat}. Ainsi, un second bit est extrait sur le front descendant du signal S_{Beat}. Finalement pour réaliser l'opération d'échantillonnage mutuel, ce double bloc est dupliqué de nouveau pour aboutir à la structure présentée sur la Figure 3.18, dans laquelle les signaux Sampler_clk et TRNG_clk doivent respecter les conditions suivantes :

$$F_{TRNG\_clk} = 2 \times F_{Sampler\_clk} \tag{3.4}$$

$$F_{TRNG\_clk} \leq 4 \times F_{T_{Beat}} \tag{3.5}$$

Ainsi, 4 bits consécutifs sont extraits toutes les périodes du signal S_{Beat}.

Nous avons fixé le paramètre *confidence level* α à 0.01 dans le code de la suite de tests aléatoires NIST SP800-22. Ainsi le nombre minimal de suites à tester N est de $\alpha^{-1} = 100$. Cependant lorsque cela nous a été possible, nous avons préféré utiliser $N = 1000$ suites de 1M bit chacune aboutissant à une suite de 1G bits, garantissant ainsi la qualité des suites obtenues.

Les résultats présentés dans le Tableau 3.7 valident sans ambiguïté la possibilité d'utiliser la double extraction et l'échantillonnage mutuel dans plusieurs configurations avec différentes sources physiques d'aléa. Les débits ainsi obtenus sont intéressants car supérieurs au Mbs. S'agissant d'une implantation particulière, nous croyons qu'il est possible d'atteindre des débits plus importants encore tout en fournissant une suite de qualité suffisante permettant de passer les tests NIST sans l'aide de correcteurs ou de post-traitement. Le cas le plus intéressant à nos yeux est celui obtenu dans la plate-forme Actel Fusion utilisant un oscillateur en anneau et une PLL comme source d'aléa. En effet cette configuration particulière présente l'avantage d'être totalement automatisable à la production et ne nécessite aucune intervention de calibration d'un opérateur humain.

Nous pouvons noter également que l'échantillonnage mutuel avec deux oscillateurs en anneau n'a pas donné des résultats concluants dans les plat-formes Actel Fusion et Xilinx Spartan 3 alors que il a parfaitement fonctionné dans la plate-forme Altera Cyclone III. Nous expliquons ceci par le fait que le placement des deux oscillateurs joue

un rôle important notamment sur leur interférence mutuelle. En effet, pour obtenir un Δ suffisamment petit sur les deux premières cibles, les oscillateurs ont été placés très prés l'un de l'autre alors que dans la cible Altera Cyclone III ils étaient placé suffisamment loin. Nous pensons et avons observé que lorsque les oscillateurs sont physiquement proches des phénomènes de couplage se produisent. Ainsi si les signaux S_1 et S_2 sont fortement dépendant à cause d'un tel couplage, il est probable que l'échantillonnage mutuel est à éviter ce qui est validé par les résultats des tests NIST.

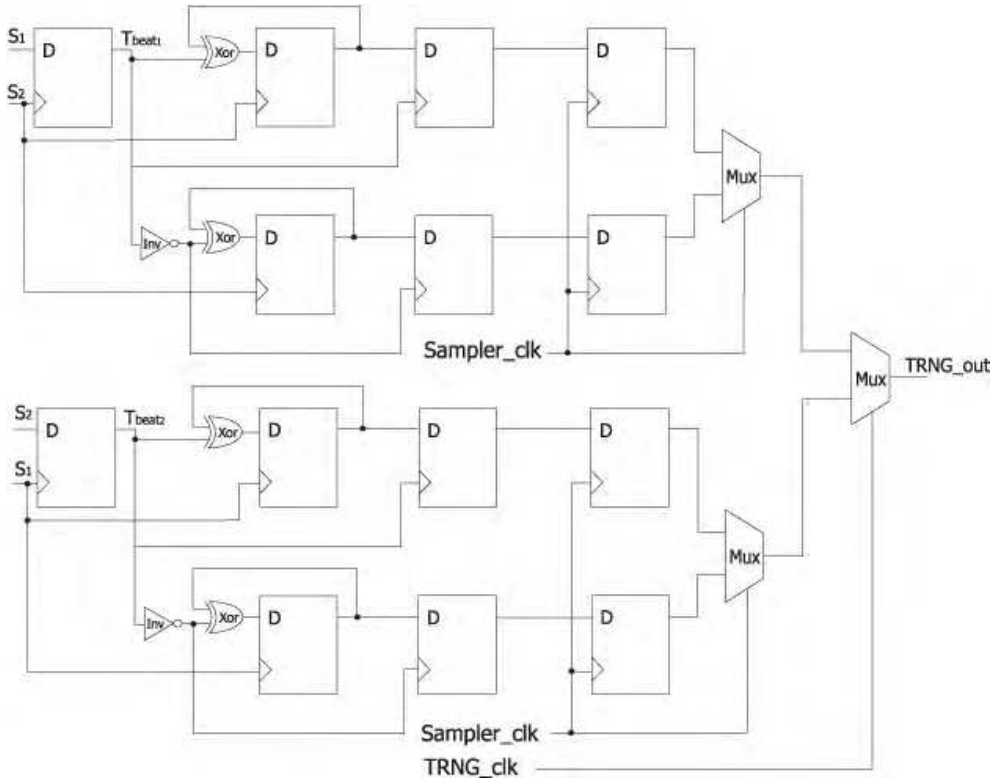

Figure 3.18 TRNG complet basé sur l'échantillonnage cohérent mettant en oeuvre la double extraction et l'échantillonnage mutuel

Plate-forme	Source	Double extraction	Échantillonnage mutuel	N	débit
Actel Fusion 600	2 RO	**oui**	non	200x1Mbits	2Mbs
Xilinx Spartan 3	2 RO	**oui**	non	200x1Mbits	1.6Mbs
Actel Fusion 600	RO + PLL	**oui**	**oui**	200x1Mbits	2 Mbs
Altera Cyclone III	2 RO	**oui**	**oui**	1000x1Mbits	3 Mbs

Tableau 3.7 Validation des principes d'extraction physique d'aléa à base d'échantillonnage cohérent avec double échantillonnage et échantillonnage mutuel dans le cadre de TRNG complets implantés dans différentes familles avec différentes sources d'aléa.

5 Conclusion

Nous avons montré dans ce chapitre deux méthodes effectives pour produire des suites aléatoires de bonne qualité sans l'aide de correcteurs ou de fonctions de post-traitements.

Une première méthode de génération d'aléa basée sur l'accumulation de jitter a l'aide de compteurs à été présentée. Cette méthode donne des suites aléatoires de très bonne qualité cependant le débit obtenu reste très faible de l'ordre de la dizaine de Kbs. C'est pour cette raison que nous n'avons pas validé cette méthode à l'aide de la suite statistique NIST SP800-22 qui exige des suites d'une longueur élevée qu'il est difficile d'obtenir avec un débit faible. Cependant au regard des suites obtenues, qui satisfont les tests FIPS 140-1 et 140-2 et particulièrement des écarts-types obtenus, nous restons persuadés que les suites ainsi obtenues passeraient les tests NIST. Ainsi, si le débit n'est pas un facteur déterminant, nous pensons que cette méthode est la plus élégante et la plus sûre pour la génération d'aléa.

Nous avons également montré qu'il était possible d'utiliser plusieurs sources en parallèle avec cette méthode. Les travaux sur cet extracteur n'ont pour le moment pas fait l'objet de publications scientifiques. Nous restons cependant persuadés de leur intérêt pour la communauté scientifique.

Un second principe de génération d'aléa a également été présenté dans ce chapitre. Bien que très semblable au principe présenté dans [KG04], nous avons apporté un nombre conséquent de modifications et d'améliorations. Ces améliorations ont fait l'objet de publications scientifiques dans les *proceedings* d'une conférence internationale IEEE [VFA09b] mais également dans d'autres conférences internationales [VFA09a], [VFAF09]. Parmi ces améliorations, nous pouvons citer l'amélioration du débit en proposant deux améliorations de l'échantillonnage : la double extraction et l'échantillonnage mutuel, la simplification du principe, ne nécessitant plus de logique de contrôle complexe et surtout l'obtention de suites de bonne qualité sans l'aide de correcteur, ce qui dans [KG04] n'est pas le cas. De plus, nous avons étudié différentes manières d'obtenir les signaux nécessaires à ce principe et donc différentes sources possibles d'aléa et nous avons validé le principe avec ces sources. Une configuration particulièrement intéressante a été présentée. Il s'agit d'un TRNG pouvant délivrer une suite aléatoire de bonne qualité sans fonction de correction avec un débit de 2Mbs et ne nécessitant pas une intervention manuelle pour le placement et le routage.

Nos travaux sur cet extracteur physique d'aléa ont pour point de départ les articles [SR04] et [HC01] dans lesquels une méthode de mesure de jitter en interne est proposée basée sur l'échantillonnage cohérent. Partant du principe que l'on pouvait facilement transformer une mesure interne du jitter en TRNG, nous avons proposé notre principe semblable à [KG04].

Chapitre 4

Modélisation d'une source d'aléa : l'oscillateur en anneau

Dans ce chapitre seront présentées les explications théoriques pour la compréhension du fonctionnement des extracteurs physiques d'aléa décrits dans le chapitre précèdent. En effet ces compléments sont essentiels pour la sécurité des générateurs car ils permettent de ne pas considérer ceux-ci comme des boîtes noires. De plus ces modèles mathématiques sont primordiaux pour les concepteurs de TRNG car ils permettent de choisir les paramètres qui permettront un fonctionnement optimal avec la connaissance des sources d'aléa utilisées. Dans ce chapitre, sera également présenté un modèle d'une source d'aléa très souvent utilisée le jitter dans les oscillateurs en anneau. Enfin il sera également introduit des outils de simulation de cette source et par conséquent de générateurs complets.

1 Introduction

Il est très important d'un point de vue de la sécurité d'un générateur de ne pas le considérer comme une boîte noire, mais au contraire de le décrire et de le caractériser autant que possible. En effet puisque l'application visée est l'utilisation des suites aléatoires produites par le générateur comme clés de chiffrement il est indispensable de comprendre d'où provient l'aléa afin de pouvoir garantir un fonctionnement dans des condition de fonctionnement strictement définies.

C'est dans cette optique de description du générateur que nous avons cherché à modéliser une source d'aléa qui est la base de nombre de générateurs : le jitter contenu dans un oscillateur en anneau.

2 Outils mathématiques

Nous présenterons dans cette section les notations et les principaux outils mathématiques utilisés dans le cadre de cette thèse.

2.1 Entropie

L'entropie est un terme commun à beaucoup de disciplines scientifiques (thermodynamique, linguistique, ...) et une certaine confusion peut apparaître lors de son utilisation. Dans le cadre de cette thèse, nous utiliserons le terme *entropie* exclusivement dans sa définition introduite par Claude Shannon en 1948 [Sha01] qui est celle de la mesure de l'incertitude sur l'information véhiculée par un message. En effet si un message est constant (si son contenu est certain) il n'est pas porteur d'information. Au contraire si un message présente des changements dans son contenu au cours du temps (s'il est incertain) alors il est porteur d'information. L'entropie est la mesure de cette information. On parle d'entropie de Shannon ou entropie au sens de la théorie de l'information (introduite par Shannon).

L'entropie est notée H et elle est définie pour une variable aléatoire X qui prend des valeurs ou réalisations ω dans un ensemble dénombrable Ω. Nous noterons $Prob(X = \omega)$ la probabilité comprise entre 0 et 1 que la variable aléatoire X prenne la valeur ω de l'ensemble des valeurs possibles Ω. L'entropie est alors définie par :

$$H(X) = -\sum_{\omega=\Omega} Prob(X = \omega) log_2 \left(Prob(X = \omega)\right) \tag{4.1}$$

Cette somme est calculée sur l'ensemble Ω et sa valeur est maximale pour une distribution uniforme sur cet ensemble. Ainsi si $\Omega = 0, 1$ l'entropie est maximale, et égale à un bit lorsque le *biais* est nul. Dans ce cas nous avons $ProbX = 1 = ProbX = 0 = 0.5$. Plus le *biais* est grand plus la valeur de l'entropie s'éloigne de la valeur maximale de 1 bit.

Lorsque ω peut prendre des valeurs dans un ensemble $\Omega = 0, 1, 2..., 2^n$, c'est à dire dans un ensemble de n bits, l'entropie représente le nombre de bits en moyenne qui sont nécessaires pour représenter les réalisations de X. L'entropie est maximale et égale à n bits lorsque toutes les valeurs possibles de Ω sont équiprobables. Si par exemple sur une population de 1000 réalisations d'un processus aléatoire qui peut prendre des valeurs sur $n = 8$ bits l'entropie $H = 3$ bits, cela est équivalent à dire qu'en moyenne seuls 3 des 8 bits sont porteurs d'information, les 5 autres étant constants. Ainsi en moyenne on peut utiliser seulement 1000 fois 3 bits pour décrire entièrement la suite initiale des 1000 x 8 bits. Puisque qu'il s'agit de bits en moyenne, la valeur de H peut être non entière.

L'utilisation du logarithme en base 2 dans le calcul de l'entropie implique d'utiliser seulement des probabilités non nulles. Généralement on fait la convention que $log_2(0) = 0$ bien que le logarithme en base 2 n'est pas défini pour une valeur en 0.

Il est très important de noter ici que l'entropie est définie pour une variable aléatoire et non pas pour des réalisations de cette variable aléatoire [Sch03]. Cela sous-entend de connaître la loi de répartition des probabilités sur l'ensemble Ω de manière analytique. Or dans le cadre des générateurs d'aléa étudiés dans cette thèse cela implique de disposer d'un modèle mathématique, ce qui est rarement le cas dans la pratique. De plus si un modèle mathématique existe, il repose la plupart du temps sur des hypothèses sur la source d'aléa qui sont difficilement vérifiables dans la pratique. C'est pourquoi on calcule l'entropie d'une suite expérimentale en estimant les probabilités d'occurrence de chaque valeur possible de l'ensemble observé Ω de manière empirique. Cette entropie empirique est utile pour caractériser une suite donnée issue du TRNG à un moment donné mais elle ne fournit aucun renseignement sur la qualité du TRNG.

Entropie et aléa Il est très important de noter que l'entropie n'est en aucune manière une mesure de l'aléa. En fait la nature même de l'aléa est difficile à saisir et encore plus difficile à quantifier. Le seul cas ou l'on peut utiliser l'entropie pour quantifier l'aléa et le cas où l'on calcule l'entropie d'une variable aléatoire. Cela revient à supposer comme vrai à priori le fait que la suite produite est réellement aléatoire, dans ce cas seulement on peut utiliser l'entropie pour quantifier l'aléa. Au contraire si l'entropie est calculée à partir d'une séquence donnée même avec une entropie très grande nous n'avons aucune garantie sur la quantité d'aléa qui peut même être nulle. C'est le cas avec des générateurs de pseudo-aléa qui produisent des suites de très bonne qualité statistiques, avec une entropie très grande mais qui sont totalement déterministes et donc ne contiennent aucune

2.2 Corrélation et coefficient de corrélation

Dans le cadre de cette thèse, nous avons été souvent amenés à étudier et quantifier la dépendance de deux phénomènes aléatoires. La corrélation est un cas particulier de dépendance entre deux variables aléatoires ou deux processus aléatoires. Il s'agit d'une dépendance affine (linéaire). On la quantifie grâce au *coefficient de corrélation* ρ.

$$\rho(x, y) = \frac{\sigma_{xy}}{\sigma_x \sigma_y} \tag{4.2}$$

où σ_{xy} est la covariance entre x et y

$$\sigma_{xy} = \frac{1}{N} \sum_{i=1}^{N} (x_i - \bar{x})\,(y_i - \bar{y}) \tag{4.3}$$

et σ_x et σ_y les écart-types. Sur la Figure 4.1 sont représentés plusieurs graphiques en XY avec différents facteurs de corrélation. Sur la ligne supérieure sont représentés différents cas de dépendance linéaire avec différents coefficients de corrélation. On peut constater que lorsque la corrélation est maximale, le graphique en XY présente uniquement une

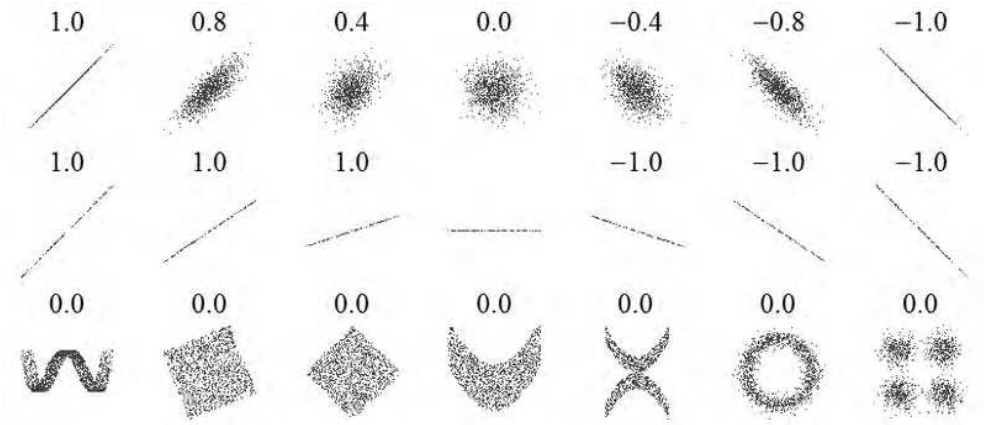

Figure 4.1 Différents cas de dépendance et le facteur de corrélation correspondant (Wikipedia)

ligne tandis que lorsque le coefficient de corrélation est faible on obtient un nuage de points d'autant plus dispersé que la corrélation est faible. Sur la ligne horizontale du milieu sur la même figure sont représentés plusieurs cas de corrélation totale. Et enfin sur la troisième ligne de la Figure 4.1 sont représentés plusieurs cas de dépendances non linéaires. Le graphique en XY révèle bien cette dépendance, cependant le coefficient de corrélation n'est pas adapté pour quantifier une telle dépendance non linéaire. Ainsi on retiendra que même avec un coefficient de corrélation ρ nul, il n'est pas exclu d'avoir une dépendance non linéaire.

2.3 Information mutuelle

Puisque le facteur de corrélation nous renseigne uniquement sur une dépendance linéaire, nous avons recherché un autre indicateur plus général de la dépendance entre deux variables aléatoires. Cet autre indicateur est l'information mutuelle qui peut être exprimée en bits. Cet indicateur tend vers 0 lorsque les deux variables aléatoires sont indépendantes.

$$Mi(X,Y) = H(X,Y) - H(X|Y) - H(Y|X) \tag{4.4}$$

où $H(X)$ et $H(Y)$ sont des entropies, $H(X,Y)$ est une entropie conditionnelle et $H(X|Y)$ est une entropie conjointe entre les variables aléatoires X et Y. L'entropie conditionnelle mesure l'entropie restante provenant de la variable aléatoire Y, si l'on connaît parfaitement la seconde variable aléatoire X. C'est l'entropie de Y conditionnée par X. L'entropie conjointe mesure combien d'information est contenue dans un système de deux variables aléatoires.

Ainsi, l'information mutuelle mesure la quantité d'information apportée en moyenne par une réalisation de la variable aléatoire X sur les probabilités de réalisation de la seconde variable aléatoire Y. C'est pour cette raison que si le nombre de bits nécessaires pour coder une réalisation du couple est égal à la somme du nombre de bits pour coder une réalisation de X et du nombre de bits pour coder une réalisation de Y, l'information

mutuelle est nulle indiquant l'indépendance des deux variables aléatoires.

Nous avons utilisé un *package* Matlab [Pen] pour estimer l'information mutuelle entre deux séquences provenant de deux sources différentes. Comme pour le cas de l'entropie toute seule, il s'agit d'un calcul sur deux séquences particulières et non sur une expression analytique des densités de probabilité des deux variables aléatoires. Cependant, c'est un indicateur intéressant pour déterminer le niveau de dépendance entre les deux suites étudiées.

3 Simulation du jitter en VHDL avec ModelSim

Nous avons développé dans le cadre de cette thèse une méthode de simulation de signaux jittés très utile lors de l'étude des extracteurs et de la conception de TRNG complets. Cette méthode nous a également servi dans la publication d'une série d'articles [BBF09], [VABF08], [FBBV08] .

La méthode traditionnelle d'analyse de circuits numériques implantés dans des cibles FPGA passe par une étape de simulation. Une fois la description matérielle synthétisée, on peut en étudier le comportement grâce à un simulateur. On distingue deux types de simulations : *simulation fonctionnelle* et *simulation temporelle*, la première comme son nom l'indique est une simulation de haut niveau (au sens informatique) ou seulement la fonction réalisée par chaque bloc du circuit est simulée sans tenir compte des temps de propagation des signaux à travers les différents circuits, les changements d'états des signaux se font de manière instantanée. La seconde méthode de simulation est plus bas niveau. Le simulateur tient compte des temps de propagation de chaque porte logique ainsi que de chaque interconnexion. En effet un modèle statique est disponible pour chaque puce avec tous les temps de propagation des cellules individuelles ainsi que des interconnexions. Ces temps de propagations sont déterminés par le constructeur du FPGA et sont liés à la technologie utilisée. Cependant les deux méthodes sont totalement inadaptées pour simuler des générateurs d'aléa ou des extracteurs physiques d'aléa à cause de leur caractère statique.Or comme nous l'avons montré dans le Chapitre 2, le jitter est essentiellement un phénomène dynamique.

Ainsi avec les outils classiques de simulation, il est impossible d'étudier et de simuler les TRNG. Or la simulation est un outil très précieux pour comprendre un système. Dans le contexte de la maîtrise des générateurs d'aléa son utilisation nous paraît très importante. C'est pour cette raison que nous avons développé une méthode qui permet de simuler un signal qui présente du jitter mais également la simulation complète d'un oscillateur en anneau.

Le principe est simple mais ouvre une grande gamme d'applications. Il est basé sur la possibilité offerte par le langage de description matérielle VHDL de travailler en lecture et écriture avec des fichiers en mode texte. Nous présenterons dans cette section

uniquement la simulation de signaux jittés. La simulation d'un oscillateur en anneau complet sera présentée dans la Section 4.3 de ce chapitre. La méthode consiste à utiliser le *package* TEXTIO et de créer un fichier texte contenant une suite de demi périodes du signal à simuler. Ce fichier sera lu par un *testbench* classique et le signal sera maintenu dans un état haut ou bas pendant toute la valeur de la durée de la demie période. Ainsi il est aisé d'intégrer n'importe quel profil de jitter aussi bien aléatoire que déterministe ou ce qui est le plus intéressant un mélange des deux. Nous avons utilisé Matlab pour générer ainsi des fichiers textes avec des signaux suivant une loi normale avec différentes tailles de jitter gaussien pour valider différents principes de génération d'aléa.

4 Modélisation d'une source d'aléa : le jitter dans un oscillateur en anneau

4.1 Présentation du modèle

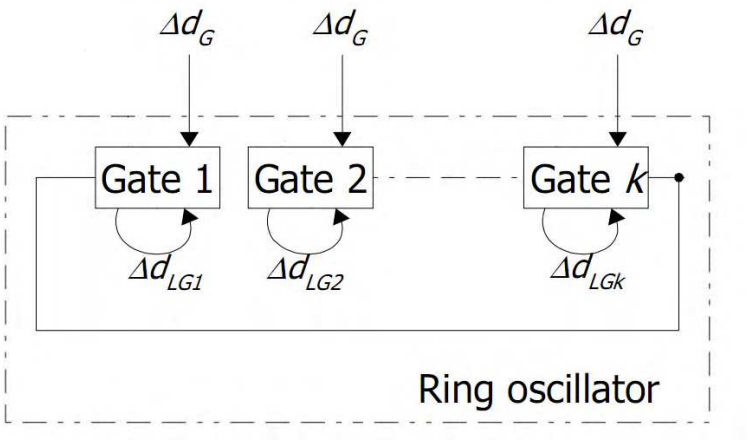

Figure 4.2 Modèle d'un oscillateur en anneau, séparation des sources de jitter.

Nous présenterons dans cette section une modélisation d'un oscillateur en anneau. Comme nous l'avons montré dans la Section 4.1 du Chapitre 2, l'oscillateur en anneau est une structure auto-oscillante basée sur la propagation d'un changement d'état à travers un ensemble d'éléments de retards constitués dans la pratique par les portes logiques et par les délais d'interconnexion. Ainsi une demie période d'oscillation P, entre deux changements d'état du signal, peut être exprimée (eq.4.5) comme la somme des k éléments à retard constituants l'oscillateur en anneau. Chaque élément i, parmi les k éléments qui constituent l'oscillateur en anneau, est constitué d'un délai provenant d'une porte logique d_i et d'un délai r_i provenant de l'interconnexion de deux portes

logiques consécutives.

$$P = \sum_{i=1}^{k} (d_i + r_i), \tag{4.5}$$

Pour tenir compte du jitter dans ce modèle, nous avons proposé dans [VABF08] une séparation des sources de jitter. Cette modélisation haut niveau du jitter est présentée à la Figure 4.2. Nous avons ainsi proposé de séparer ces sources en sources *locales* et en sources *globales*. Cette séparation est très pratique et correspond assez bien aux conditions réelles de fonctionnement de l'oscillateur. En effet chaque composant électronique est inévitablement *bruyant* et contribue ainsi de façon locale au jitter dans l'oscillateur. Tous les éléments sont cependant sujets à des perturbations *globales* tels que celles induites par de très forte dépendance à la tension d'alimentation, comme cela a été montré dans la Section 4.3 du Chapitre 2. Ainsi le délai d'un élément à retard i constitué d'une porte logique peut être exprimé grâce à l'équation 4.6 où D_i représente la valeur idéale du délai autour de laquelle viendront s'ajouter des perturbations Δd. Pour les délais engendrés par les portes logiques, ces sources se décomposent en sources *locales* Δd_{Li}, communes à chaque porte ou chaque interconnexion et en sources *globales* Δd_G.

$$d_i = D_i + \Delta d = D_i + \Delta d_{Li} + \Delta d_G \tag{4.6}$$

De même le délai de l'interconnexion i peut être exprimé de manière analogue :

$$r_i = R_i + \Delta r = R_i + \Delta r_{Li} + \Delta r_G \tag{4.7}$$

Nous supposons que les perturbations globales sont communes aux portes logiques et aux interconnexions et ainsi nous avons :

$$\Delta r_G = \Delta d_G \tag{4.8}$$

Cependant pour tenir compte de l'aspect temporel des perturbations nous devons récrire les équations 4.6 et 4.7 en fonction du temps :

$$d_i(t) = D_i + \Delta d_i(t) = D_i + \Delta d_{Li}(t) + \Delta d_G(t) \tag{4.9}$$

De même le délai de l'interconnexion i peut être exprimé de manière analogue :

$$r_i(t) = R_i + \Delta r_i(t) = R_i + \Delta r_{Li}(t) + \Delta d_G(t) \tag{4.10}$$

Cependant bien que les variations de perturbations globales et locales soient continues dans le temps, seul leur état aux instants discrets (*cf.* Figure. 4.3) est important. Ainsi nous travaillerons avec des valeurs discrètes évaluées aux temps de transition du front. Les équations finales permettant de décrire le fonctionnement d'un oscillateur en anneau sujet à des perturbation locales et globales seront donc :

$$P = \sum_{i=1}^{k} (d_i + r_i), \tag{4.11}$$

$$d_i = D_i + A\Delta d_{Li}(t_i) + C\left(K_i^d \Delta d_G(t_i^{'})\right) \tag{4.12}$$

$$r_i = R_i + B\Delta r_{Li}(t_i^{''}) + C\left(K_i^r \Delta d_G(t_i^{'''})\right) \tag{4.13}$$

Nous avons rajouté des facteurs propres à chaque porte K^d et à chaque interconnexion K^r. Ces facteurs trouvent leur justification dans le fait que chaque porte peut être exposée aux mêmes perturbations globales de manière légèrement différente. De plus des facteurs pour ajuster les amplitudes des différentes perturbations, locales (A,B) et globale (C) ont été intégrés.

Les équations 4.11, 4.12, 4.13 forment le modèle d'un oscillateur en anneau haut

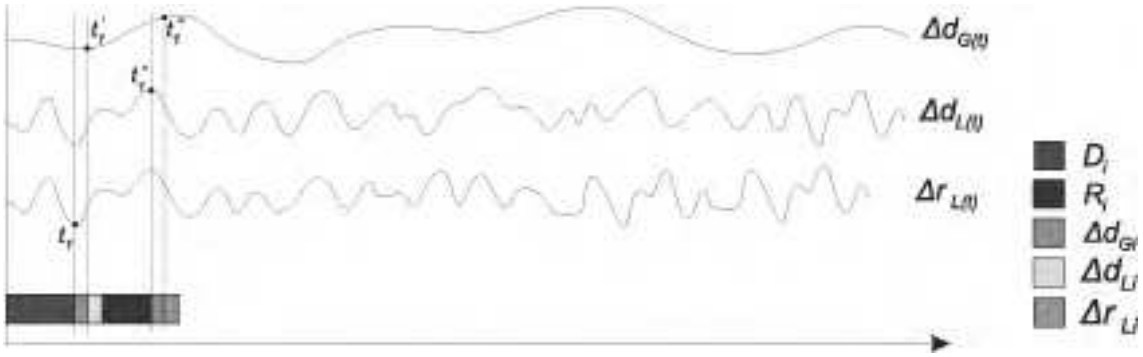

Figure 4.3 Model de l'oscillateur en anneau : influence des perturbations globales et locales sur la formation de la demie période d'oscillation.

niveau qui permet de simuler n'importe quel profil de jitter en contrôlant les signaux $\Delta d_G(t)$, $\Delta d_{Li}(t)$ et $\Delta r_i(t)$. De plus ce modèle présente l'avantage de prendre en compte deux types de sources de jitter, des sources locales et des sources globales. Les signaux $\Delta d_G(t)$ et $\Delta d_{Li}(t)$ peuvent être considérés comme des variations d'une tension dans le temps et être ramenés à des variations du temps de propagation grâce aux relations quasi linéaires existants entre la tension d'alimentation et le temps de propagation comme cela a été montré dans la Section 4.3.1 du Chapitre 2.

Pour choisir l'allure des signaux $\Delta d_G(t)$, $\Delta d_{Li}(t)$ et $\Delta r_i(t)$ à simuler, nous avons procédé à une suite d'expériences de caractérisation d'un oscillateur en anneau dans deux familles différentes et sur 3 cibles différentes (2 cibles de la famille Actel et 1a cible de la famille Spartan).

4.2 Résultats expérimentaux

Afin de tester notre modèle, nous avons procédé à une série de mesures de jitter sur les oscillateurs en anneau pour collecter suffisamment de données. Ainsi nous avons procédé à l'évaluation du jitter dans deux FPGA différents sur des oscillateurs en anneau comportant un nombre croissant d'éléments à retard. Ces résultats expérimentaux sont présentés à la Figure 4.4 pour la cible Actel Igloo et à la Figure 4.5 pour

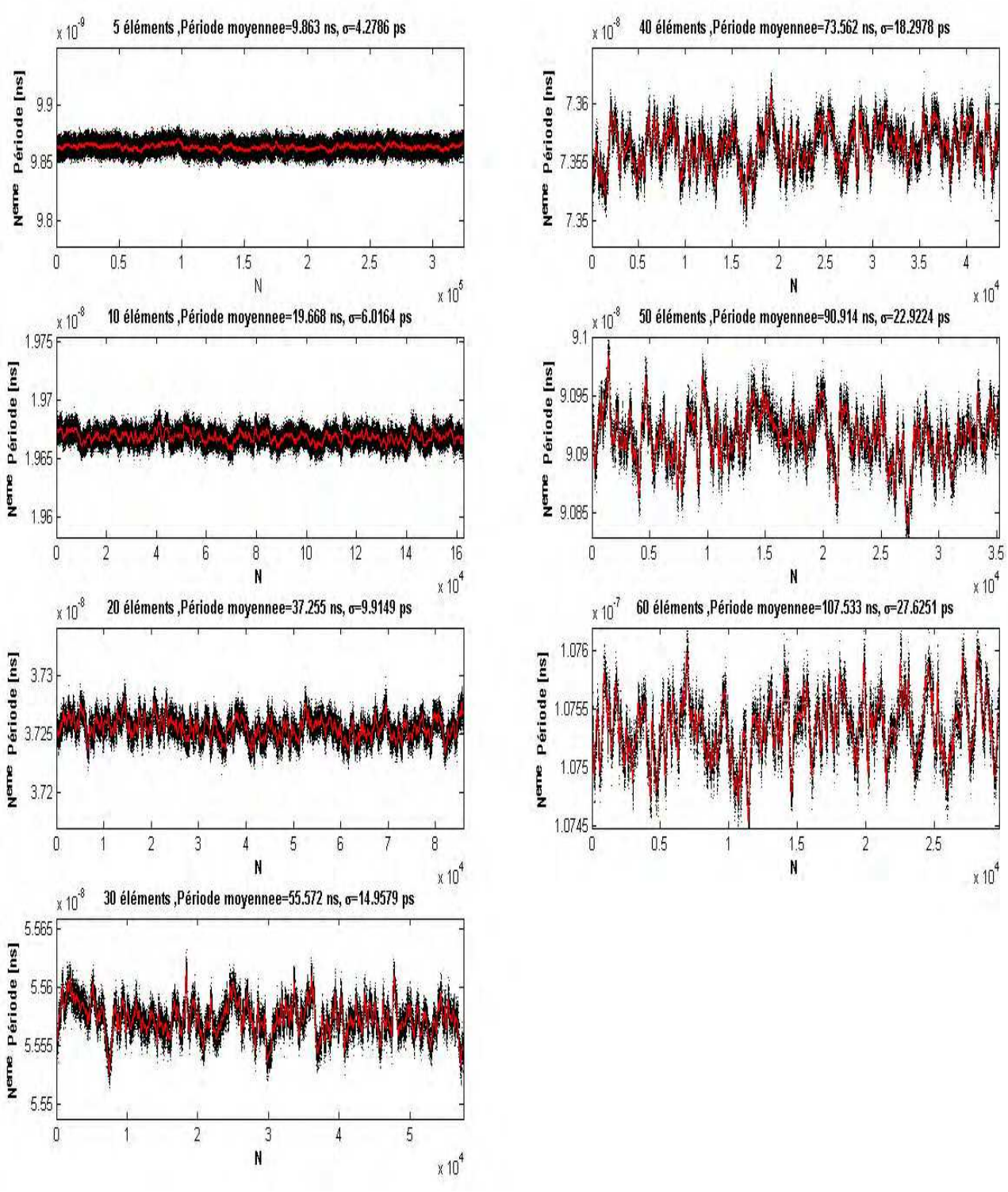

Figure 4.4 Mesures expérimentales du jitter dans un oscillateur en anneau en fonction du nombre d'éléments constituants dans la famille Actel Igloo.

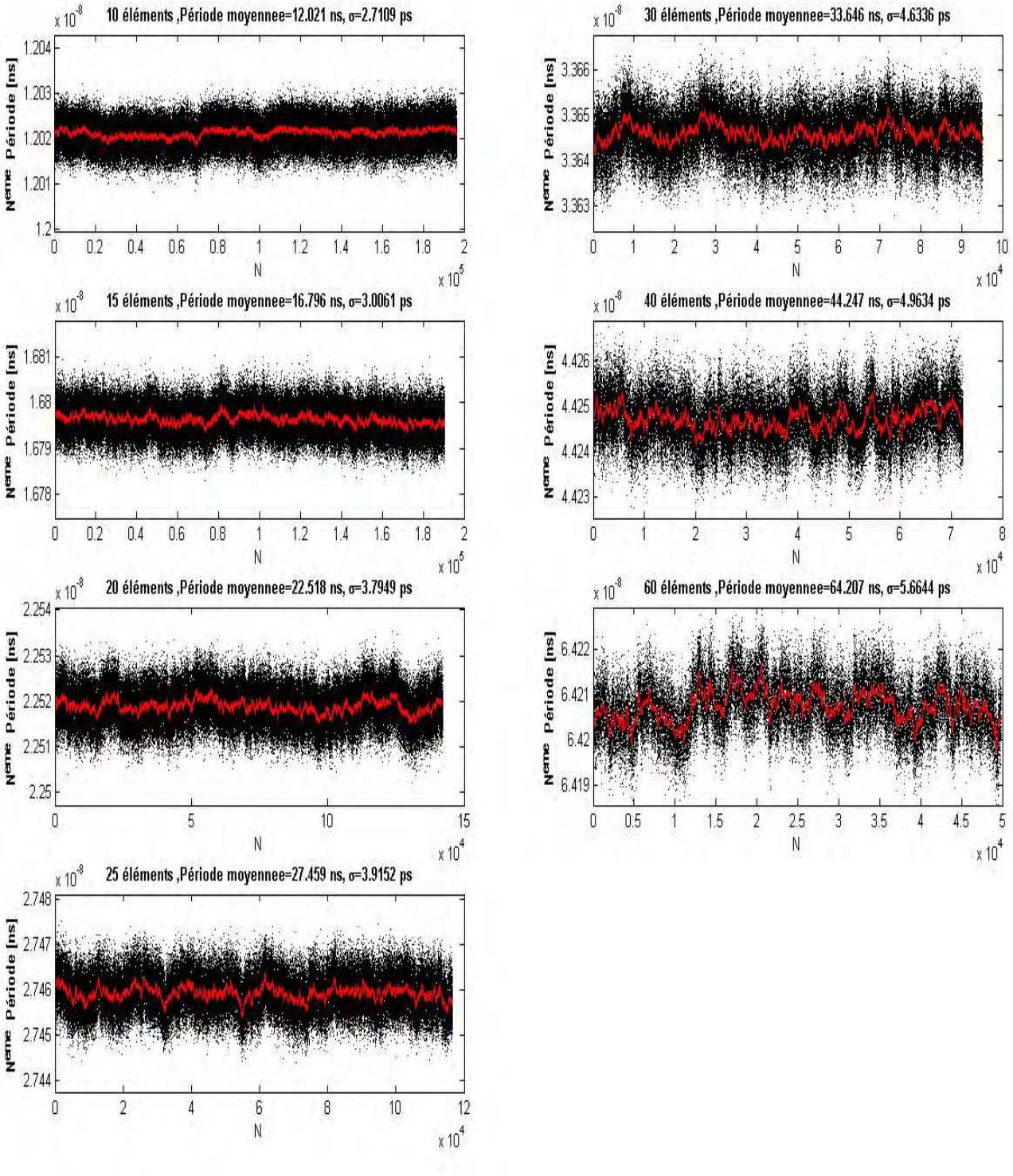

Figure 4.5 Mesures expérimentales du jitter dans un oscillateur en anneau en fonction du nombre d'éléments constituants dans la famille Actel Fusion.

la cible Actel Fusion 600. La Figure 4.4 présente le *period jitter* dans une cible Actel Igloo sur une carte d'évaluation Actel Icicle. Les périodes consécutives ont été acquises grâce aux méthodes de mesure externes présentées dans la Section 3 du Chapitre 2 en utilisant le standard de transmission différentiel LVDS. Sur cette Figure on peut observer le *period jitter* à la sortie d'oscillateurs comportant un nombre croissant d'éléments. Comme on peut le constater l'écart type du *period jitter* augmente avec le nombre d'éléments constituants l'oscillateur en anneau. Plus ce nombre est grand plus des composantes basses fréquences apparaissent dans la valeur moyenne du *period jitter*. On peut constater que la valeur moyenne du *period jitter* n'est dans aucune des configurations constante ce qui invalide l'hypothèse que le jitter dans un oscillateur en anneau suit une loi normale. Tous les graphiques présentés sur les Figures 4.4 et 4.5 sont représentés avec la même échelle verticale, et l'axe des ordonnées sur chaque graphique correspond à $\pm 20 \times (\sigma = 4.27ps)$. L'échelle horizontale est en revanche différente pour chaque graphique car la vitesse d'échantillonnage étant fixée à 40GS/s et la mémoire de taille limitée 128MS, les graphiques correspondants aux signaux basses fréquences (avec un grand nombre d'éléments à retard) présentent inévitablement moins de périodes consécutives acquises.

Pour mieux comprendre ce phénomène d'augmentation de l'écart type en fonction du

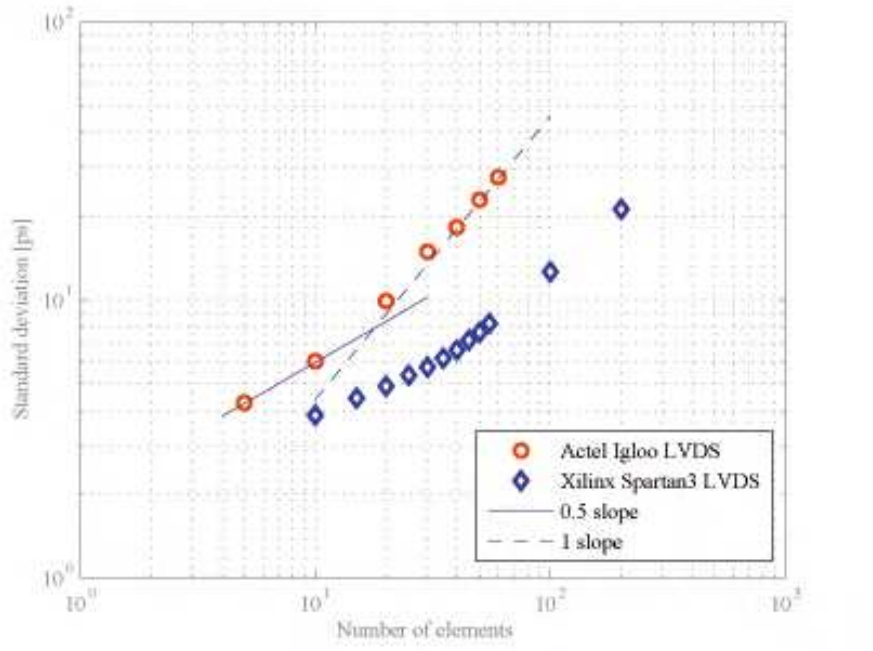

Figure 4.6 Écarts-types expérimentaux en fonction du nombre d'éléments dans l'oscillateur en échelle log-log. Visibilité de deux asymptotes correspondantes à un comportement dépendant et indépendant du jitter.

nombre d'éléments nous avons représenté cette évolution sur une échelle logarithmique. Sur la Figure 4.6 sont représentés les écarts types présentés à la Figure 4.4 mais également ceux mesurés sur une cible Xilinx Spartan 3. Comme on le voit il est possible

de distinguer nettement deux régions pour les deux familles. Une première partie des données tend vers une asymptote de pente 0.5 pour un nombre faible d'éléments à retard et une seconde partie des données tend vers une asymptote de 1 pour un nombre plus important d'éléments à retard constituants l'oscillateur.

Ce phénomène trouve sa justification dans le fait que dans la pratique deux types de sources de variations prennent place dans la formation du jitter dans un oscillateur en anneau. Comme cela a été montré dans [HLL99], les sources indépendantes et des sources dépendantes contribuent de manière différente à la formation du jitter.

En effet, d'après la formule 4.11 chaque demie période constituant le signal jitté obtenu par un oscillateur en anneau est en fait une somme des éléments à retard qui le constituent. Chaque élément intègre des variations dans ces temps de propagation. Ainsi si aucune dépendance temporelle n'existe dans ces variations, c'est à dire si le signal $\Delta d_G(t)$ évalué aux instants $t_i^{'}$ et $t_i^{'''}$ pour $i \in 1...k$ ou si les signaux $\Delta d_{Li}(t)$ et $\Delta r_{Li}(t)$ ne présentent pas de dépendance temporelle, alors l'augmentation de l'écart-type suit une tendance en $\sqrt{k}$ (k étant le nombre d'éléments constituants l'oscillateur). Si au contraire les valeurs des signaux $\Delta d_G(t)$ ou $\Delta d_{Li}(t)$ ou $\Delta r_{Li}(t)$ présentent des dépendances temporelles aux instants de transition du front, alors l'augmentation de l'écart type se fait en proportion linéaire en fonction du nombre d'éléments.

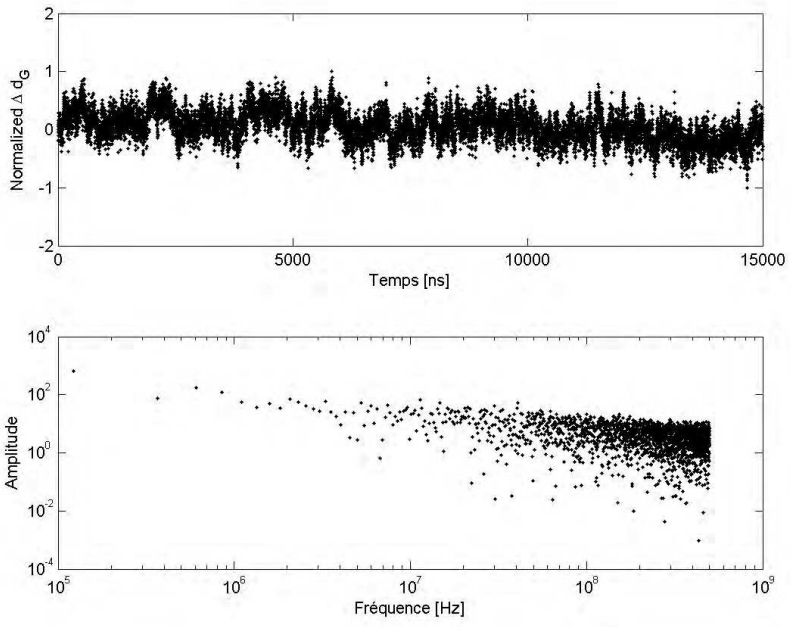

Figure 4.7 Signal de perturbation global utilisé pour reproduire le comportement de l'oscillateur en anneau et sa FFT.

4.3 Simulation grâce au modèle de l'oscillateur en anneau

Pour valider le modèle proposé dans la section précédente, nous avons voulu reproduire le comportement expérimentalement observé. L'allure des données expérimentales dans trois familles de FPGA différentes (Figure 4.4, fig 4.5 et Figure 4.6), suggèrent sans ambiguité la présence de composantes temporellement dépendantes dans la formation du jitter. Ceci peut être déduit du fait que les valeurs moyennes de ces signaux sont fortement non stationnaires. Ainsi la valeur moyenne à chaque période est dépendante de la valeur de la période précédente. Pour modéliser ces dépendances nous avons utilisé du bruit 1/F pour le signal $\Delta d_G(t)$. Ce bruit 1/F, totalement aléatoire, présente des amplitudes plus importantes aux basses fréquences qu'aux hautes fréquences. Ainsi ces composantes basses fréquences produiront des dépendances temporelles dans les perturbations successives aux instants t_i' et t_i'''. Le signal $\Delta d_G(t)$ utilisé est représenté à la Figure 4.7. C'est un signal normalisé dans une plage [-1 et 1] sans dimension. Cette amplitude est ensuite convertie en perturbation des temps de propagation des portes et des interconnexions grâce au facteur C présent dans les équations 4.12 et 4.13. Ainsi nous sommes en mesure de simuler les oscillateurs en anneau avec plusieurs niveaux différents de perturbations globales en changeant uniquement une constante et en travaillant avec un type de signal bien défini. Pour cette première campagne de simulation, les coefficients K^r et K^d, correspondants aux susceptibilités aux perturbations globales intrinsèques à chaque porte ou interconnexion, seront fixé à une valeur de 1. Pour les signaux de perturbation au niveau local de chaque porte et

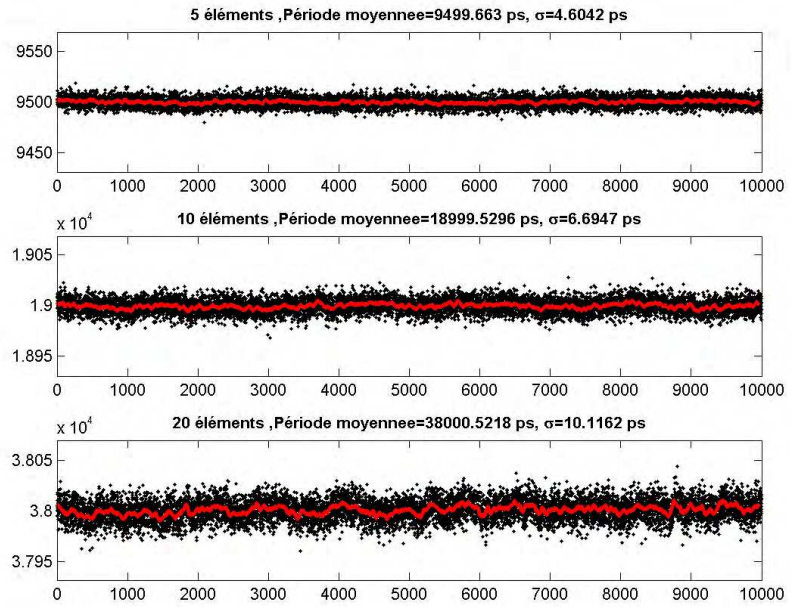

Figure 4.8 Comportement simulé temporel : oscillateur avec 5, 10, 20 éléments, Di=650ps ; Ri=300ps ; σ_L=1ps ; C=0.2 (par rapport a une normalisation entre -1 et 1)

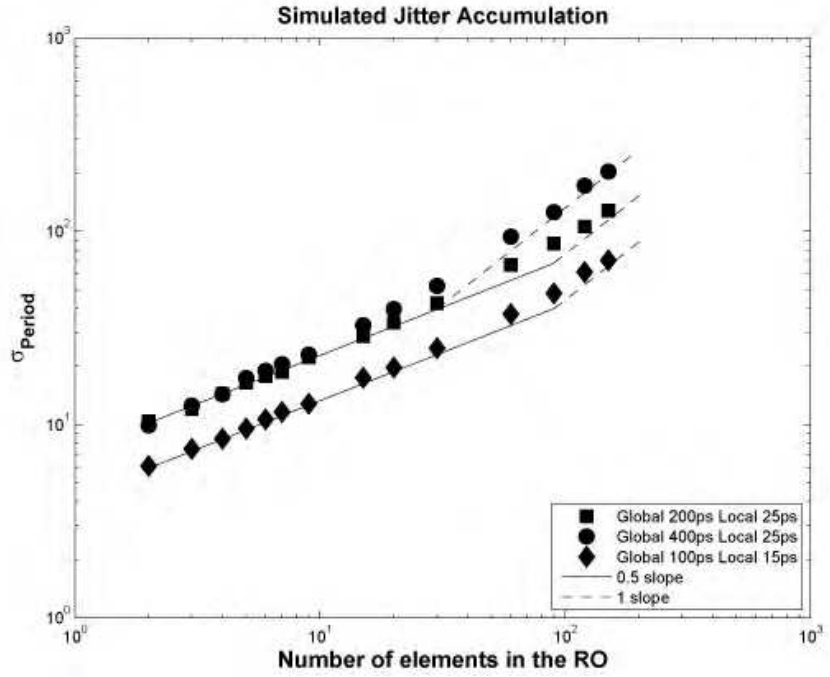

Figure 4.9 Résultats de simulation de l'oscillateur en anneau en échelle log-log avec différents niveau de perturbation

de chaque interconnexion Δd_{Li}, nous avons choisi d'utiliser un signal Gaussien qui suit une loi $N(0, \sigma_L)$ sans tenir compte de l'aspect fréquentiel de ce signal. Ainsi chaque porte et chaque interconnexion présenteront leurs propres sources de fluctuations des temps de propagation sans limite fréquentielle autre que celle introduite par les temps de propagation eux-mêmes. Chaque porte logique et chaque interconnexion sera une source de perturbations Gaussienne indépendante des autres.

Les résultats de la simulation sont reproduits à la Figure 4.8. Comme on peut le constater les allures des signaux jittés obtenus sont très semblables à ceux observés expérimentalement et présentés à la Figure 4.4. Particulièrement l'aspect non stationnaire du *period jitter*, dû à l'accumulation du jitter au sein des différents éléments constituants l'oscillateur en anneau. Pour rendre particulièrement compte de ce phénomène, nous avons procédé à plusieurs simulations et nous avons évalué l'écart type du *period jitter* obtenu en fonction du nombre croissant d'éléments à retard constituants l'oscillateur en anneau. Les résultats pour différents niveaux de perturbations locales (A et B) et globales (C) sont présentés à la Figure 4.9. On constate que les deux régions de pente 0.5 et 1 dans une échelle logarithmique sont bien respectés conformément aux observations présentées à la Figure 4.6 et que l'intersection de ces deux asymptotes est définie en fonction des amplitudes des perturbations dépendantes (C) et des perturbations indépendantes (A et B) comme suggéré dans [MS], [HLL99].

5 Conclusion

Dans ce chapitre nous avons présenté la modélisation d'une source d'aléa. Dans un premier temps une méthode haut niveau de simulation de signaux jittés a été présentée, bien que ne reposant pas sur une modélisation de la source d'aléa cette méthode est néanmoins assez puissante pour réaliser des simulations complètes de TRNG. Ainsi le signal servant de source d'aléa est définit par une fonction mathématique supposée correspondre à la source réelle d'aléa. Cette méthode est très puissante puisque elle permet de confronter les comportements de différents TRNG avec différentes quantités de jitter et d'en étudier le comportement. Cette méthode a été décrite dans [VABF08] et a servie pour étudier le comportement avec et sans jitter d'un TRNG complet [BBFV10], [BBF09].

Dans un second temps nous avons voulu disposer d'un modèle complet de la source d'aléa elle même - l'oscillateur en anneau. Pour ce faire nous avons d'abord étudié le comportement d'oscillateurs en anneau dans différentes familles et avons noté qu'une composante de bruit 1/F était présente lorsque les oscillateurs en anneau étaient composés d'un nombre important d'éléments à retard. Cette observation, valable pour toutes les familles de FPGA disponibles nous a conduit à injecter un tel bruit de perturbation *globalement* dans le modèle de l'oscillateur. Ainsi la simulation du modèle présenté dans ce chapitre correspond bien à celui observé expérimentalement. Ce modèle à fait l'objet de publications dans des *proceedings* de conférences internationales IEEE [VABF08] et [VAFB10] ainsi que dans [VFA09a].

Conclusion et perspectives

Synthèse des travaux

Les travaux présentés dans ce document concernent la génération de suites binaires aléatoires destinés à des équipements cryptographiques embarqués dans des cibles FPGA. En effet des nombres aléatoires interviennent à presque tous les niveaux dans les protocoles cryptographiques. Ces nombres ou plus précisément ces suites binaires aléatoires doivent néanmoins satisfaire des critères sévères compte tenu de leur rôle dans les systèmes de chiffrement. Ainsi les nombres aléatoires pour les systèmes cryptographiques doivent avant tout être imprévisibles et non reproductibles. Cela exclue toute génération par des voies déterministes ou algorithmiques. Ils doivent dépendre d'un phénomène physique de nature aléatoire et personne, pas même le concepteur du générateur ne doit être en mesure de les prédire ou de les reproduire. De plus, la suite binaire aléatoire doit présenter des qualités statistiques importantes pour être utilisable. Ces qualités sont testées grâce à des batteries de tests statistiques. Toutes ces exigences font que la génération d'aléa dans un système reconfigurable numérique de nature déterministe est un domaine actif de recherche et bien que de nombreuses solutions aient été proposées par la communauté scientifique, il reste encore beaucoup de questions ouvertes.

Nous avons commencé par présenter un état de l'art qui regroupe une grande partie des articles scientifiques traitants du domaine de la génération d'aléa à des fins cryptographiques. Certains principes de génération d'aléa qui nous ont paru très pertinents ont été analysés avec un point de vue critique. D'autres ont été seulement cités dans cette étude bibliographique.

La classification et la confrontation entre les différents TRNG n'est pas aisée en partant uniquement des articles scientifiques. Pour qu'un classement ou une confrontation des différents principes soit pertinent, il faudrait implanter et évaluer tous les principes concurrents dans la ou les mêmes cibles et effectuer les tests dans les mêmes conditions de fonctionnement. A notre connaissance, cette étude n'a pas encore été effectuée. Nous avons tout de même présenté une évaluation théorique des TRNG les plus importants à nos yeux sous forme d'un tableau en faisant une évaluation sur différents critères qui nous semblent les plus importants. Cependant, cette évaluation reste théorique et n'a pas l'ambition d'être un classement. Nous avons préféré classer les articles par ordre chronologique sous la forme d'un arbre. Ainsi les affiliations et les évolutions des principes de génération d'aléa sont clairement visibles.

De plus cette représentation permet de faire un classement des différents TRNG. Nous voyons ainsi que ces dernières années, les concepteurs de TRNG se focalisent sur des structures d'oscillateurs modifiés pour rentrer dans des états métastables afin d'augmenter les débits. Notre avis sur ce choix de la communauté est partagé : d'un coté il est évident que les débits sont effectivement améliorés cependant la robustesse face à des manipulations de l'environnement n'est jusqu'à présent que rarement discutée.

Nous avons abouti également à un autre constat : très peu d'articles présentent une étude de la source d'aléa utilisée. La plupart se contentent de signaler un comportement aléatoire de la source (jitter, métastabilité) mais peu fournissent une caractérisation de cette source. C'est pourquoi une très grande partie de notre travail durant cette période de recherche a porté sur la caractérisation d'une des sources d'aléa le plus souvent utilisée : le jitter.

La question de la mesure en interne et en externe a été également soulevée dans le second chapitre. Toute mesure en interne est difficile à réaliser car les signaux mis en jeux ont des fréquences élevées souvent supérieures à la centaine de Mhz. Ainsi il n'existe pas à notre connaissance de méthode de mesure en interne qui puisse fournir autant de précision qu'une mesure en externe, c'est à dire fournir des mesures de périodes successives complètes pour caractériser le jitter. Les méthodes internes fournissent uniquement des estimations du jitter mais sont incapables de nous renseigner sur la dynamique et l'aspect fréquentiel du jitter. Les mesures en externe cependant présentent un inconvénient de taille : le signal jitté qui dans un fonctionnement nominal reste interne au FPGA doit passer par des *buffers* de sortie et à travers une connexion physique entre la puce et les *pads* de sortie. De plus, la sonde de mesure externe elle même est susceptible d'apporter des modifications au signal. Ainsi le signal final mesuré en externe à l'aide de l'oscilloscope ne correspond plus au signal réel en interne utilisé pour la génération d'aléa. Nous avons ainsi montré dans le second chapitre que le standard électronique de transmission utilisé pour transmettre le signal hors du FPGA joue un rôle capital puisque la valeur du jitter peut être surestimée d'un facteur 2. Cette observation est nouvelle et soulève la question sur la quantification du jitter reporté dans la littérature jusqu'à présent. Nous avons finalement opté pour un standard de transmission différentiel (LVDS) pour effectuer nos mesures en externe de manière comparative plutôt que de manière quantitative.

Nous avons étudié le jitter dans différentes structures internes aux FPGA et donc susceptibles d'être utilisées comme source d'aléa. Nous avons montré que chaque structure présente des caractéristiques et des comportements spécifiques. Cette étude originale nous a amené également à la conclusion que chaque structure peut fournir des signaux jittés avec des avantages et des inconvénients au regard de l'intégration dans un TRNG.

De nombreuses recherches ont été consacrées aux oscillateurs en anneau puisque c'est la source d'aléa la plus souvent utilisée. Nous avons étudié la susceptibilité de ces

oscillateurs en anneau face aux manipulations de l'environnement. Nous avons conclu que la susceptibilité était élevée, notamment face à des manipulations externes de la tension d'alimentation. Nous avons ainsi montré une attaque originale consistant à verrouiller deux oscillateurs au moins sur la même fréquence pour réduire l'aléa dans un TRNG.

Le second chapitre montre ainsi les différents profils du jitter de différentes sources de signaux jittés. Une synthèse décrivant chaque source avec les avantages et les inconvénients est présentée. On peut conclure que les oscillateurs en anneau bien que susceptibles à la logique environnante, restent en bonne place dans le domaine de recherche de la génération d'aléa du fait de leur facilité d'utilisation et d'implantation. Les DFS, présents dans les familles de FPGA Xilinx, et déjà utilisés par plusieurs principes de génération d'aléa montrent des comportements périodiques qui peuvent rendre leur utilisation moins intéressante que les PLL présents dans la totalité des familles de FPGA étudiés.

Dans le Chapitre 3, nous avons proposé deux implantations de TRNG et en avons validé un des deux grâce à la suite de tests statistiques NIST SP800-22 avec différents débits. Nous avons pris le parti de faire une séparation entre la source physique d'aléa, l'extraction physique d'aléa et l'extraction logique d'aléa. Nous avons ainsi pu évaluer plusieurs TRNG complets différents avec différentes sources d'aléa. Une configuration particulièrement intéressante à nos yeux a été présentée, dans la mesure où elle remplit les objectifs fixés en début de thèse. Il s'agit de l'extracteur physique d'aléa basé sur l'échantillonnage cohérent mutuel avec un oscillateur à anneau et une PLL comme sources d'aléa. Cette configuration n'exige aucune intervention manuelle, passe les tests statistiques sans l'aide de correcteurs et fournit un débit important.

Un résultat important est également présenté dans le Chapitre 4. Il s'agit de la modélisation d'un oscillateur en anneau. En effet un modèle comportemental d'un oscillateur est fourni qui permet la simulation en VHDL d'oscillateurs en anneau avec du jitter. Cette simulation est importante pour la simulation en aval de TRNG complets. Ce modèle nous a permis d'obtenir plusieurs résultats publiés dans des communications scientifiques.

Perspectives

Le domaine de la génération d'aléa dans les circuits reconfigurables est un domaine en constante évolution et qui est encore en partie inexploré. Nous espérons avoir contribué à notre échelle au développement des connaissance dans ce domaine. Cependant, plusieurs questions restent ouvertes.

Parmi les perspectives que nous pouvons soulever, nous pouvons citer la question de l'indépendance de deux ou plusieurs oscillateurs en anneau à l'intérieur du FPGA.

Il s'agit de déterminer de manière précise le couplage qui peut exister entre deux ou plusieurs oscillateurs fonctionnant en même temps en fonction des topologies et des paramètres de ces oscillateurs. En effet un grand nombre de TRNG est basé sur l'utilisation de plusieurs oscillateurs et leur indépendance est souvent conjecturée sans preuves solides à l'appui. Dans l'autre sens aussi, plusieurs auteurs ont supposé un phénomène d'interdépendance sans vraiment en apporter la preuve ou l'étudier. Ainsi nous voyons comme perspective une étude de ce couplage en fonction de la disposition des oscillateurs et en fonction des leurs paramètres (fréquences de fonctionnement, topologie) ainsi que de l'environnement.

Une autre question dont nous n'avons pas pu développer tout le potentiel qui à nos yeux présente un réel intérêt pratique dans la conception et l'étude de TRNG est le développement d'un système sur cible FPGA qui permette d'analyser en continu et en temps réel la sortie d'un TRNG sur un écran SVGA par exemple. Il s'agirait d'un dispositif, l'équivalent d'un oscilloscope dédié à l'étude de TRNG, qui prendrait en entrée un signal d'horloge et un signal de donnée véhiculant la suite aléatoire et qui fournirait en sortie sur l'écran en temps réel des informations sur le fonctionnement du TRNG. Nous envisageons de fournir en temps réel des informations comme le biais, les histogrammes, les tests FIPS, une image *bit map* de la suite binaire aléatoire et une représentation de *l'espace des phases* tridimensionnel. Une première version de cet analyseur de TRNG nous a convaincu de l'utilité de disposer de ces informations en temps réel lors de la conception de nouveaux principes de génération mais également dans le développement d'attaques et de contre-mesures sur des TRNG. En effet pour le moment les enregistrements des suites aléatoires ne concernent que des fractions de secondes et peuvent cacher d'éventuels comportements anormaux des TRNG à long terme.

Une question intéressante qui jusque là n'a pas fait l'objet de beaucoup de discutions dans la communauté scientifique et qui est pourtant essentielle pour des applications réelles des TRNG est la question de la consommation d'énergie d'un TRNG en fonctionnement.

Enfin, une comparaison sur des circuits cibles identiques et dans des conditions de fonctionnement identique de plusieurs principes de TRNG serait aussi une voie importante de la recherche dans ce domaine.

Bibliographie de l'auteur

Revues internationales avec comité de lecture :

- [BBFV10] Nathalie Bochard, Florent Bernard, Viktor Fischer, Boyan Valtchanov
  True Randomness and Pseudo-randomness in Ring Oscillator-based True Random Number Generators
  Special Issue of International Journal of Reconfigurable Computing
  Selected Papers from ReconFig 2009 International Conference on Reconfigurable Computing and FPGA (ReconFig2009)

- [BFV10] Florent Bernard, Viktor Fischer, Boyan Valtchanov
  Mathematical Model of Physical RNGS Based on Coherent Sampling
  Tatra Mountains Mathematical Publications - Volume 45, 2010, pages : 1-14

Conférences internationales avec comité de lecture et actes :

- [VABF08] Boyan Valtchanov, Alain Aubert, Florent Bernard, Viktor Fischer
  Modeling and observing the jitter in ring oscillators implemented in FPGAs
  11-th IEEE Internation Symposium on Design and Diagnostic of Electronic Circuits and Systems (DDECS'2008), Bratislava, Slovak Republic, 2008
  pages : 158-163

- [VVDA08] Michal Varchola, Boyan Valtchanov, Milos Drutarovsky, Alain Aubert
  Embedded monitoring of robustness used for ring oscillator based TRNGs implemented in FPGA
  23-th International Conference on Design of Circuits and Inegrated Systems (DCIS'2008), Grenoble, France, 2008

- [VFA09a] Boyan Valtchanov, Viktor Fischer, Alain Aubert
  A coherent sampling based method for estimating the jitter used as entropy source for True Random Number Generators
  8-th International Conference on Sampling Theory and Applications (SAMPTA'09), Marseille, France, 2009

- [VFA09b] Boyan Valtchanov, Viktor Fischer, Alain Aubert
  Enhanced TRNG based on the coherent sampling
  3-rd IEEE International Conference on Signals, Circuits and Systems (SCS'09),

Djerba, Tunisia, 2009
Print ISBN : 978-1-4244-4397-0

- [VAFB10] Boyan Valtchanov, Viktor Fischer, Alain Aubert, Florent Bernard
  Characterization of randomness sources in ring oscillator-based true random number generators in FPGAs
  13-th IEEE International Symposium on Design Diagnostics of Electronic Circuits and Systems (DDECS'2010), Vienna, Austria, 2010
  pages : 48-53, print ISBN : 978-1-4244-6610-8

Conférences internationales avec comité de lecture :

- [VVFA08]Boyan Valtchanov, Michal Varchola, Viktor Fischer, Alain Aubert
  Poster : Simulation of an attack on the ring oscillator based TRNG
  10-th Workshop pn Cryptographic Hardware and Embedded Systems (CHES'2008), Washington, D.C., USA

- [VFA+08] Boyan Valtchanov, Viktor Fischer, Alain Aubert, Florent Bernard, Nathalie Bochard
  Modeling and securing RO-based TRNG in FPGAs - Jitter accumulation from local and global sources
  7-th International Workshop on Cryptographic Architectures Embedded in Reconfigurable Devices (CryptArchi'2008), Tregaste, France, 2008

- [VFAF09] Boyan Valtchanov, Viktor Fischer, Alain Aubert
  TRNG based on the coherent sampling
  8-th International Workshop on Cryptographic Architectures Embedded in Reconfigurable Devices (CryptArchi'2009), Praha, Czech Republic, 2009

- [FBVB10] Viktor Fischer, Florent Bernard, Boyan Valtchanov, Nathalie Bochard
  About the randomness of RO-based TRNGs in FPGAs
  9-th International Workshop on Cryptographic Architectures Embedded in Reconfigurable Devices (CryptArchi'2009), Gif-sur-Yvette, France, 2010

Colloques sans actes :

- **Journées du projet EmSoC, Villard de Lans, France, 2008**
  Boyan Valtchanov, Michal Varchola, Viktor Fischer, Alain Aubert
  Poster : Simulation of an attack on the ring oscillator based TRNG

- **Journées du projet Semba, Annecy, France, 2009**
  Boyan Valtchanov, Viktor Fischer, Alain Aubert
  Poster : Coherent Sampling Based True Random Number Generator

- **Journée de la recherche de l'école doctorale SIS 488, Saint-Etienne, France, 2008**
  Viktor Fischer, Alain Aubert, Boyan Valtchanov, Nathalie Bochard, Florent Bernard, Robert Fouquet
  Poster : Cryptographie, générateurs de clés confidentielles enfouis dans les circuits logiques programmables

- **Journée de la recherche de l'école doctorale SIS 488, Saint-Etienne, France, 2009**
  Boyan Valtchanov, Viktor Fischer, Alain Aubert
  Poster : Coherent sampling based TRNG

Bibliographie

[Abc04] B.J. Abcunas. *Evaluation of random number generators on FPGAs.* PhD thesis, WORCESTER POLYTECHNIC INSTITUTE, 2004.

[AC2] Actel Application Note AC204. Designing clean analog pll power supply in a mixed-signal environment.

[Acta] I. Actel. Actel fusion family of mixed-signal flash fpgas.

[Actb] I. Actel. Igloo dc and switching characteristics.

[BB99] V. Bagini and M. Bucci. A design of reliable true random number generator for cryptographic applications. *Lecture notes in computer science*, pages 204–218, 1999.

[BBF09] Nathalie Bochard, Florent Bernard, and Viktor Fischer. Observing the randomness in ro-based trng. *Reconfigurable Computing and FPGAs, International Conference on*, 0 :237–242, 2009.

[BBFV10] N. Bochard, F. Bernard, V. Fischer, and B. Valtchanov. True randomness and pseudo-randomness in ring oscillator-based true random number generators. *En soumission : Special Issue of the International Journal of Reconfigurable Computing (IJRC)*, Selected Papers from ReconFig 2009 Internationnal Conference on Reconfigurable Computing and FPGA(ReconFig2009)(X), 2010.

[BBL04] H. Bock, M. Bucci, and R. Luzzi. An offset-compensated oscillator-based random bit source for security applications. *Lecture notes in computer science*, pages 268–281, 2004.

[BFV10] F. Bernard, V. Fischer, and B. Valtchanov. Mathematical model of physical rngs based on coherent sampling. *Tatra Mountains Mathematical Publications*, 45(X) :1–14, 2010.

[BGL+03] M. Bucci, L. Germani, R. Luzzi, A. Trifiletti, and M. Varanonuovo. A high-speed oscillator-based truly random number source for cryptographic applications on a smart card ic. *IEEE Transactions on Computers*, 52(4) :403–409, 2003.

[BGL+06] M. Bucci, L. Giancane, R. Luzzi, M. Varanonuovo, A. Trifiletti, I.T. AG, and A. Graz. A novel concept for stateless random bit generators in cryptographic applications. In *2006 IEEE International Symposium on Circuits and Systems, 2006. ISCAS 2006. Proceedings*, page 4, 2006.

[BL05] M. Bucci and R. Luzzi. Design of testable random bit generators. *Lecture notes in computer science*, 3659 :147, 2005.

[BL07] M. Bucci and R. Luzzi. A testable random bit generator based on a high resolution phase noise detection. *IEEE Design and Diagnostics of Electronic Circuits and Systems, 2007. DDECS'07*, pages 1–5, 2007.

[BST03] B. Barak, R. Shaltiel, and E. Tromer. True random number generators secure in a changing environment. *Lecture notes in computer science*, pages 166–180, 2003.

[Cop05] W.R. Coppock. *A mathematical and physical analysis of circuit jitter with application to cryptographic random bit generation.* PhD thesis, WORCESTER POLYTECHNIC INSTITUTE, 2005.

[Cor55] Rand Corporation. *A million random digits with 100,000 normal deviates.* Free Press New York,, 1955.

[CS94] S. Crocker and J. Schiller. Rfc1750 : Randomness recommendations for security. *RFC Editor United States*, 1994.

[CSG08] Octavian Cret, Alin Suciu, and Tamas Gyorfi. Practical issues in implementing trngs in fpgas based on the ring oscillator sampling method. In *SYNASC '08 : Proceedings of the 2008 10th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing*, pages 433–438, Washington, DC, USA, 2008. IEEE Computer Society.

[Dav02] R. Davies. Exclusive or (xor) and hardware random number generators. 28 :1–11, 2002.

[DG07] M. Dichtl and J.D. Golic. High-speed true random number generation with logic gates only. *Lecture notes in computer science*, 4727 :45, 2007.

[DGH07] J.L. Danger, S. Guilley, and P. Hoogvorst. Fast true random generator in fpgas. In *Circuits and Systems, 2007. NEWCAS 2007. IEEE Northeast Workshop on*, pages 506–509, 2007.

[DGH09] J.L. Danger, S. Guilley, and P. Hoogvorst. High speed true random number generator based on open loop structures in fpgas. 2009.

[Dic03] M. Dichtl. How to predict the output of a hardware random number generator. *Lecture notes in computer science*, pages 181–188, 2003.

[Dic07] M.v Dichtl. Bad and good ways of post-processing biased physical random numbers. *Lecture notes in computer science*, 4593 :137, 2007.

[DJ00] M. Dichtl and N. Janssen. A high quality physical random number generator. In *Proc. Sophia Antipolis Forum Microelectronics (SAME 2000)*, pages 48–53, 2000.

[EHK+03] M. Epstein, L. Hars, R. Krasinski, M. Rosner, and H. Zheng. Design and implementation of a true random number generator based on digital circuit artifacts. *Lecture notes in computer science*, pages 152–165, 2003.

[Eps] Epson. Sg-8002jf programmable high-frequency crystal oscillator.

[FBBV08] Viktor Fischer, Florent Bernard, Nathalie Bochard, and Michal Varchola. Enhancing security of ring oscillator-based trng implemented in fpga. In *FPL*, pages 245–250, 2008.

[FBVB10] Viktor Fischer, Florent Bernard, Boyan Valtchanov, and Nathalie Bochard. About the randomness in ro-based trngs in fpgas. In *CryptArchi 2010*, 2010.

[FD03] V. Fischer and M. Drutarovsky. True random number generator embedded in reconfigurable hardware. *Lecture notes in computer science*, pages 415–430, 2003.

[FDS+04] V. Fischer, M. Drutarovsky, M. Simka, F. Celle, U.M. Instrumentation, and P. Komenskeho. Simple pll-based true random number generator for embedded digital systems. In *Proceedings of IEEE Design and Diagnostics of Electronic Circuits and Systems Workshop–DDECS*, pages 129–136. Citeseer, 2004.

[FMC85] RC Fairfield, RL Mortenson, and KB Coulthart. An lsi random number generator (rng). In *Proceedings of CRYPTO 84 on Advances in cryptology table of contents*, pages 203–230. Springer-Verlag New York, Inc. New York, NY, USA, 1985.

[Fon98] C. Fontaine. *Contribution à la recherche de fonctions booléennes hautement non linéaires, et au marquage d'images en vue de la protection des droits d'auteur*. PhD thesis, Université Paris VI, 1998.

[GCS09] T. Gyorfi, O. Cret, and A. Suciu. High performance true random number generator based on fpga block rams. In *Proceedings of the 2009 IEEE International Symposium on Parallel&Distributed Processing-Volume 00*, pages 1–8. IEEE Computer Society, 2009.

[Gol06] JDJ Golic. New methods for digital generation and postprocessing of random data. *IEEE Transactions on Computers*, 55(10) :1217–1229, 2006.

[GW96] I. Goldberg and D. Wagner. Randomness and the netscape browser. *Dr Dobb's Journal-Software Tools for the Professional Programmer*, 21(1) :66–71, 1996.

[HC01] J.L. Huang and K.T. Cheng. An on-chip short-time interval measurement technique for testing high-speed communication links. In *Proceedings of the 19th IEEE VLSI Test Symposium*, page 380. IEEE Computer Society, 2001.

[HKUK03] A. Hasegawa, S.J. Kim, K. Umeno, and N. Kitamachi. Ip core of statistical test suite of fips 140-2. *Communications Research Laboratory and ChaosWare, Inc., Tech. Rep*, 2003.

[HLL99] A. Hajimiri, S. Limotyrakis, and TH Lee. Jitter and phase noise in ring oscillators. *IEEE Journal of Solid-State Circuits*, 34(6) :790–804, 1999.

[HOB+06] J. Holleman, B. Otis, S. Bridges, A. Mitros, and C. Diorio. A 2.92 μw hardware random number generator. In *Proceedings of the 32nd Eurpoean Solid state circuits conference*, 2006.

[Inca] ECS Inc. Ecs-2200bx - 8 pin dip clock oscillator - ecs, inc.

[Incb] Tektronix Inc. Understanding and characterizing timing jitter.

[ISC09] H. Istvan, A. Suciu, and O. Cret. Fpga based trng using automatic calibration. In *IEEE 5th International Conference on Intelligent Computer Communication and Processing, 2009. ICCP 2009.*, pages 373 – 376, 2009. ISBN 978-1-4244-5007-7.

[JJSH00] A. Juels, M. Jakobsson, E. Shriver, and B.K. Hillyer. How to turn loaded dice into fair coins. *IEEE Transactions on Information Theory*, 46(3) :911–921, 2000.

[JK99] B. Jun and P. Kocher. The intel random number generator. *Cryptography Research Inc. white paper*, 1999.

[JRH10] Hlavac Josef, Lorencz Robert, and Martin Hadacek. Generating true random bits on general-purpose microcontrollers. In *CryptArchi 2010*, 2010.

[KC02] DJ Kinniment and EG Chester. Design of an on-chip random number generator using metastability. In *Proceedings of the 28th European Solid-State Circuits Conference, ESSCIRC*, volume 2002, pages 595–598, 2002.

[KCS08] C. Klein, O. Cret, and A. Suciu. Design and implementation of a high quality trng in fpga. In *Intelligent Computer Communication and Processing, 2008. ICCP 2008. 4th International Conference on*, pages 311–314, 2008.

[KG04] 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*, pages 71–78. ACM, 2004.

[KL06] S.H.M. Kwok and E.Y. Lam. Fpga-based high-speed true random number generator for cryptographic applications. In *2006 IEEE Region 10 Conference TENCON 2006*, pages 1–4, 2006.

[Knu73] D.E. Knuth. *The art of computer programming, Vol. 3.* Addison-Wesley, Reading, MA, 1973.

[KS38] MG Kendall and B.B. Smith. Randomness and random sampling numbers. *Journal of the royal Statistical Society*, 101(1) :147–166, 1938.

[KS01] W. Killmann and W. Schindler. Ais 31 : Functionality classes and evaluation methodology for true (physical) random number generators, version 3.1. *Bundesamt fur Sicherheit in der Informationstechnik (BSI), Bonn*, 2001.

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

[KUH04] Song-Ju Kim, Ken Umeno, and Akio Hasegawa. Corrections of the nist statistical test suite for randomness, 2004.

[Lab94] Information Technology Laboratory. Security requirements for cryptographic modules. Special Publication, January 1994.

[Lab01] Information Technology Laboratory. Security requirements for cryptographic modules. Special Publication, May 2001.

[LBRPB] S. Lopez-Buedo, P. Riviere, P. Pernas, and E. Boemo. Run-time reconfiguration to check temperature in custom computers : An application of jbits technology. *Field-Programmable Logic and Applications : Reconfigurable Computing Is Going Mainstream*, pages 117–135.

[LM05] C. Liu and J. McNeill. A digital-pll-based true random number generator. *Research in Microelectronics and Electronics*, 1 :113–116, 2005.

[Ma] J. Ma. A closer look at lvds technology, perecom corporation, application note# 41.

[Mar96a] G. Marsaglia. Diehard : a battery of tests of randomness. 1996.

[Mar96b] G. Marsaglia. The marsaglia random number cdrom. *See http ://stat. fsu. edu/geo*, 1996.

[Mau92] U.M. Maurer. A universal statistical test for random bit generators. *Journal of Cryptology*, 5(2) :89–105, 1992.

[MM09] A. Markettos and S. Moore. The frequency injection attack on ring-oscillator-based true random number generators. *Cryptographic Hardware and Embedded Systems-CHES 2009*, pages 317–331, 2009.

[MS] MK Mandal and BC Sarkar. Ring oscillators : Characteristics and applications.

[MT02] G. Marsaglia and W.W. Tsang. Some difficult-to-pass tests of randomness. *Journal of Statistical Software*, 7(3) :1–9, 2002.

[NTS99] N. Nisan and A. Ta-Shma. Extracting randomness : A survey and new constructions. *Journal of Computer and System Sciences*, 58(1) :148–173, 1999.

[PC00] CS Petrie and JA Connelly. A noise-based ic random number generator for applications in cryptography. *IEEE Transactions on Circuits and Systems I : Fundamental Theory and Applications*, 47(5) :615–621, 2000.

[Pen] Hanchuan Peng. Mutual information computation, available online : http ://mathworks.com/matlabcentral/fileexchange/14888-mutual-information-computation.

[PL04] J. Patrin and M. Li. Characterizing jitter histograms for clock and datacom applications. *Proceedings of the IEC DesignCon 2004*, 2004.

[PPL09] C. Pyo, S. Pae, and G. Lee. Dram as source of randomness. *Electronics Letters*, 45(1) :26–27, 2009.

[RSN+01] A. Rukhin, J. Soto, J. Nechvatal, M. Smid, E. Barker, BOOZ-ALLEN, and HAMILTON INC MCLEAN VA. A statistical test suite for random and pseudorandom number generators for cryptographic applications. 2001.

[San09] R. Santoro. Vers des générateurs de nombres aléatoires uniformes et gaussiens à très haut débit. *Thèse*, 2009.

[Sch01] Werner Schindler. Efficient online tests for true random number generators. In Çetin Kaya Koç, David Naccache, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2001*, volume 2162 of *LNCS*, pages 103–117. Springer, 2001.

[Sch03] W. Schindler. A stochastical model and its analysis for a physical random number generator presented at ches 2002. *Cryptography and Coding*, pages 276–289, 2003.

[Sha01] C.E. Shannon. A mathematical theory of communication. *ACM SIGMOBILE Mobile Computing and Communications Review*, 5(1) :55, 2001.

[SK03] W. Schindler and W. Killmann. Evaluation criteria for true (physical) random number generators used in cryptographic applications. *Lecture notes in computer science*, pages 431–449, 2003.

[SMS07] B. Sunar, W.J. Martin, and D.R. Stinson. A provably secure true random number generator with built-in tolerance to active attacks. *IEEE Transactions on Computers*, 56(1) :109, 2007.

[Sot99] J. Soto. Statistical testing of random number generators. In *Proceedings of the 22nd National Information Systems Security Conference, Crystal City, Virginia*. Citeseer, 1999.

[SPV06] D. Schellekens, B. Preneel, and I. Verbauwhede. Fpga vendor agnostic true random number generator. In *Proceedings of the International Conference on Field Programmable Logic and Applications (FPL'06)*, pages 1–6. Citeseer, 2006.

[SR04] S. Sunter and A. Roy. On-chip digital jitter measurement, from megahertz to gigahertz. 2004.

[SS04] N. Stefanou and S.R. Sonkusale. High speed array of oscillator-based truly binary random number generators. In *IEEE International Symposium on Circuits and Systems, ISCAS 2004*, volume 1, pages 505–8, 2004.

[SSR09] R. Santoro, O. Sentieys, and S. Roy. On-the fly evaluation of fpga-based true random number generator. In *Proceedings of the 2009 IEEE Computer Society Annual Symposium on VLSI-Volume 00*, pages 55–60. IEEE Computer Society, 2009.

[TAH06] M. Thamrin, I. Ahmad, and M.K. Hani. A true random number generator for crypto embedded systems. In *Regional Postgraduate Conference on Engineering and Science*, pages 253–256. School of Postgraduate Studies, UTM, 2006.

[Tka03] Thomas E. Tkacik. A hardware random number generator. In Burton S. Kaliski Jr., Çetin Kaya Koç, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2002*, volume 2523 of *LNCS*, pages 450–453. Springer, 2003.

[TLL03] KH Tsoi, KH Leung, and PHW Leong. Compact fpga-based true and pseudo random number generators. In *Proceedings of the IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM), California USA*, pages 51–61. Citeseer, 2003.

[TLL07] KH Tsoi, K.H. Leung, and P.H.W. Leong. High performance physical random number generator. *IEEE Proc. Computers & Digital Techniques*, 1(4) :349–352, 2007.

[UA] Altera Application Note UG-ALTPLL-8.0. Phase-locked loop (altpll) megafunctio user guide.

[VABF08] B. Valtchanov, A. Aubert, F. Bernard, and V. Fischer. Modeling and observing the jitter in ring oscillators implemented in fpgas. In *Proceedings of the 2008 11th IEEE Workshop on Design and Diagnostics of Electronic Circuits and Systems*, pages 158 – 163s. IEEE Computer Society, 2008.

[VAFB10] Boyan Valtchanov, Alain Aubert, Viktor Fischer, and Florent Bernard. Characterization of randomness sources in ring oscillator-based true random number generators in FPGAs. In *13-th IEEE Symposium on Design and Diagnostic of Electronic Circuits ans Systems (DDECS), Vienna, Austria (2010)*, pages 48–53, Vienna Austria, 04 2010. print ISBN : 978-1-4244-6610-8.

[VD09] M. Varchola and M. Drutarovsky. Embedded platform for automatic testing and optimizing of fpga based cryptographic true random number generators. *Radioengineering*, 18, 2009.

[VFA+08] Boyan Valtchanov, Viktor Fischer, Alain Aubert, Florent Bernard, and Nathalie Bochard. Modeling and securing ro-based trng in fpgas - jitter accumulation from local and global sources. In *cryptarchi 2008*, Trégastel France, 06 2008.

[VFA09a] Boyan Valtchanov, Viktor Fischer, and Alain Aubert. A coherent sampling-based method for estimating the jitter used as entropy source for true random number generators. In Laurent Fesquet and Bruno Torrésani, editors, *SAMPTA'09, International Conference on Sampling Theory and Applications SAMPTA'09*, page Special session on sampling and industrial applications, Marseille France, 2009.

[VFA09b] Boyan Valtchanov, Viktor Fischer, and Alain Aubert. Enhanced trng based on the coherent sampling. In *Proceedings of 2009 International Conference on Signals, Circuits ans Systems 2009 International Conference on Signals, Circuits ans Systems*, pages 1–6, Tunisie, 2009. Print ISBN : 978-1-4244-4397-0.

[VFAF09] Boyan Valtchanov, Viktor Fischer, Alain Aubert, and Bernard Florent. Trng based on the coherent sampling. In *CryptArchi 2009*, Prague Czech Republic, 06 2009.

[VHKK08] I. Vasyltsov, E. Hambardzumyan, Y.S. Kim, and B. Karpinskyy. Fast digital trng based on metastable ring oscillator. *Lecture notes in computer science*, 5154 :164–180, 2008.

[VN51] J. Von Neumann. Various techniques used in connection with random digits. *Applied Math Series*, 12 :36–38, 1951.

[VVDA08] Michal Varchola, Boyan Valtchanov, Milos Drutarovsky, and Alain Aubert. Embedded monitoring of robustness used for ring oscillator based trngs implemented in fpga. In *Proceedings of the 2008 Conference on Design of Circuits and Integrated Systems Conference on Design of Circuits and Integrated Systems - DCIS 2008*, page 9B.2, Grneoble France, 11 2008. 6 pages.

[VVFA08] B. Valtchanov, M Varchola, V. Fischer, and A. Aubert. Poster : Simulation of an attack on the ring oscillator based trng. 2008.

[Wed06] SW Wedge. Predicting random jitter-exploring the current simulation techniques for predicting the noise in oscillator, clock, and timing circuits. *Circuits and Devices Magazine, IEEE*, 22(6) :31–38, 2006.

[WT08] K. Wold and C.H. Tan. Analysis and enhancement of random number generator in fpga based on oscillator rings. In *Proceedings of the International Conference on Reconfigurable Computing and FPGAs (ReConFig/'08)*, pages 385–390, 2008.

[Xil] I. Xilinx. Spartan-3 fpga family : Complete data sheet. *Product Documentation,(November 2005)*.

[YSKB07] S. Yoo, B. Sunar, D. Karakoyunlu, and B. Birand. A robust and practical random number generator. 2007.

[Zal01] M. Zalewski. Strange attractors and tcp/ip sequence number analysis, 2001.

[ZZW08] S. Zhou, W. Zhang, and N.J. Wu. An ultra-low power cmos random number generator. *Solid-State Electronics*, 52(2) :233–238, 2008.