Conception d'un générateur de valeurs aléatoires en technologie CMOS AMS 0.35μm

Julio Alexander Aguilar Angulo

UNIVERSITÉ DU SUD TOULON VAR

ECOLE DOCTORALE
Equipe conception de circuit

THESE présentée par

Julio Alexander AGUILAR ANGULO

Soutenue le 15 Juin 2015

pour obtenir le grade de Docteur en sciences

Spécialité : Microélectronique

Sujet de la Thèse :

Conception d'un Générateur de Valeurs aléatoires en Technologie CMOS AMS 0.35μm

Thèse dirigée par :

Hervé Barthélemy

Edith Kussener

JURY :

- Philippe Maurine, Maître de Conférences HDR, LIRMM
- Alain Pegatoquet, Maître de Conférences HDR, Université de Nice
- Jean-Baptiste Begueret, Professeur, Université de Bordeaux
- Hervé Barthélemy, Professeur Université Sud Toulon Var
- Edith Kussener, Enseignant Chercheur HDR, ISEN Toulon
- Benjamin Duval, Ingénieur, Invia

Table des matières

Table des figures

Liste des tableaux

Liste des acronymes

Acronyme	Définition	Page
BPUF	Butterfly PUF	22
CTAT	Complementary To Absolute Temperature	68
DEMA	Differential EM radiation Analysis	12
DFA	Differential Fault Analysis	12
DPA	Differential Power Analysis	12
FIB	Focused Ion Beam	11
IM2NP	l'Institut Matériaux, Microélectronique et Nanosciences de Provence	2
ISEN	Institute Supérieur de l'Electronique et du Numérique	2
LLE	Largest Lyapunov Exponent	42
LFSR	Linear Feedback Shift Register	94
MIT	Massachusetts Institute of Technology	32
PIN	Personal Identification Number	6
PRNG	Pseudo Random Number Generator	14
PTAT	Proportional To Absolute Temperature	68
PUF	Physical Unclonable Functions	20
PVT	Process Voltage Temperature	68
RFID	Radio Frequency Identification	20
ROPUF	Ring Oscillator PUF	21
RNG	Random Number Generator	13
SDD	Système Dynamique Discret	36
SIM	Suscriber Identity Module	9
TRNG	True Random Number Generator	14

Chapitre 1

Introduction Générale

L'observation de phénomènes aléatoires, par des philosophes puis des scientifiques, au long de l'histoire, nous a aidé à créer un concept du probable, de l'aléatoire, à travers les statistiques et les mathématiques, appliquées à de différents phénomènes observables dans la nature.

La notion de l'aléatoire repose sur l'absence d'ordre ou de forme sur une suite d'événements, dans lesquels il est impossible d'identifier un patron ou règle de formation.

Par la suite, la notion d'aléatoire a continué à passionner les savants, et a progressivement marqué sa présence au sein des différentes disciplines telles que l'astronomie, la biologie, la physique et les mathématiques. Le support mathématique nécessaire s'est développé au cours des siècles, notamment grâce aux travaux de Pascal, Poisson, Moivre, Bernoulli, Bayes, Lagrange, Laplace, Markov, Kolmogorov, entre autres.

Avec l'essor des télécommunications durant le XX siècle, les phénomènes aléatoires ont pris leurs envols dans la sécurisation des communications. Cette utilisation n'est pour autant pas récente car dès l'empire romain, les systèmes de cryptographie étaient une partie importante du système de messagerie militaire (le code César), permettant le masquage de texte en changeant leur structure, suivant une règle connue par l'émetteur récepteur uniquement.

Initialement exclusive de l'utilisation militaire, la sécurisation de l'information est stratégiquement importante. La deuxième guerre mondiale représente l'étape de développement la plus forte des techniques cryptographiques. Des algorithmes de chiffrement et des méthodes de décryptage sont ainsi développés, basés sur les statistiques et calculs mathématiques.

A la même époque la puissance de calcul augmenta significativement et les algorithmes de cryptage plus sophistiqués. L'élément aléatoire intervient de façon contrôlé, sur la forme de clés de chiffrements connues que par les utilisateurs.

En effet, l'intervention humaine associé à la nature cyclique des algorithmes permet lors de l'interception des codes de chiffrement, une étude d'ingénierie inverse. Par exemple, le calculateur allemand *"Enigma"*, à été décrypté en 1932, par M. Re-

jewski et son équipe, lors d'un travail d'espionnage et d'analyse. L'aléatoire montra son utilité comme un élément qui renforce énormément un algorithme de chiffrement, garantissant la sécurité d'une communication, donc d'un échange de données.

Avec l'évolution de l'électronique intégrée et sa présence dans les applications de notre quotidien, l'aléatoire a permis que la transmission de données aborde diverses applications autres que le militaire, jusqu'au point d'incorporer des informations personnelles sur des dispositifs. Des cartes d'accès, cartes bancaires ou cartes de séjour biométriques sont des exemples importants de ces applications.

Dans de tels contextes, la nécessité de la sécurisation des données échangées est vitale, vis-à-vis de l'évolution des méthodes malveillantes d'attaque sur les puces électroniques, ainsi que sur les niveaux de la communication de données.

Les variables aléatoires interviennent dans ces types d'applications comme éléments de masquage de données. Ils participent à des algorithmes de cryptage de données effectués par des processeurs numériques (le chiffrement AES, par exemple). Un autre rôle de masquage implique l'ajout de patrons aléatoires sur les alimentations du circuit, dans le but de dissimuler les traces de tension-courant lors des changements d'états binaires. Ceci rend inexploitable une attaque sur les alimentations et l'analyse spectrale.

Un tel générateur aléatoire doit être robuste, rendant impossible toute prédiction dans le temps, et donc posséder une forte source d'entropie et une distribution binaire la plus proche de l'équiprobable.

Dans les applications sans contact, dans lesquelles le *tag*, doit récolter l'énergie d'un transmetteur de radio fréquences pour alimenter le système embarqué, la consommation du générateur aléatoire à l'intérieur ne doit pas ponctionner une quantité importante sur la consommation totale.

En prenant en considération ces contraintes de base, le générateur aléatoire intégré doit passer des tests de validation standardisés (tel que la suite de tests proposée par le NIST) pour confirmer et quantifier les caractéristiques aléatoires.

Structure du document

Le présent document présente le travail de thèse au sein de l'Institut Matériaux, Microélectronique et Nanosciences de Provence (*IM2NP*), dans l'équipe de Conception de Circuits Intégrés (*CCI*), réalisés à l'*ISEN* (Institute Supérieur de l'Electronique et du Numérique) de Toulon, en collaboration avec *INVIA*, entreprise française spécialisée dans le domaine de la conception des systèmes sécurisés embarqués sans contact, entre février 2011 et 2014.

C'est dans ce contexte que nous avons orienté nos travaux de recherche vers le développement d'un circuit générateur de valeurs aléatoires, caractérisé par sa non prédictibilité. Il doit posséder un niveau de consommation faible afin d'avoir un bas impact d'intégration dans un système basé sur un microcontrôleur effectuant des calculs cryptographiques.

Le choix d'une architecture viable, résulte d'une étude comparative des systèmes existants, avec une emphase sur les architectures construites à partir de blocs analogiques-mixtes. La source d'aléa et le système de conversion vers des séquences numériques doit être capable de passer les tests standards existants pour la caractérisation des générateurs aléatoires.

Dans le premier chapitre, une présentation des systèmes à carte à puce (*smartcard*) est réalisée. Un focus particulier sera réalisé sur la sécurisation de ces dispositifs embarqués. L'importance de l'utilisation des contre mesures aux techniques d'attaque est justifiée. L'utilisation d'un générateur de valeurs aléatoires comme moyen de sécurisation est présentée, vis à vis des attaques au canal de communication. Finalement, une description des architectures des RNG, caractérisées par un fort aléa et notamment centrées sur des implémentations analogiques, est présenté.
Un comparatif entre les solutions existantes facilitera ainsi le choix d'une solution viable pour notre contexte. Ainsi, via la définition du cahier de charges, nous pourrons orienter nos démarches sur la conception d'un système.

Dans le second chapitre, axé sur la solution retenue, les concepts des systèmes dynamiques et chaos sont présentés, avec les bases théoriques pour converger vers la topologie de l'architecture à utiliser. Une formulation mathématique du modèle du système idéal est également proposée permettant de comparer l'approche théorique des différentes solutions.

Un troisième chapitre sera consacré à la mise en oeuvre du générateur aléatoire, en dimensionnant chacun des blocs, et en réalisant des simulations pré et post Layout. L'étude du circuit nous emmène au dimensionnement des blocs, du circuit et des simulations. En modifiant l'architecture pour la rendre programmable par l'utilisateur sur un banc de test et de calibration, un test chip est préparé pour sa réalisation sur silicium en technologie AMS 0.35 μm . Les résultats de simulation pré et post Layout, avec une description des types de tests à effectuer pour valider le niveau de aléatoireté des résultats.

Le quatrième chapitre correspond à l'évaluation des résultats obtenus sur le montage global que ce soit aussi bien en simulation qu'en test sur le silicium. Ces résultats sont ensuite exploités versus le passage des tests statistiques (NIST). Des améliorations et du post-traitement sont aussi analysés et proposés. Basés sur les résultats, nous allons proposer des améliorations à effectuer ainsi que des perspectives sur la

suite du travail.

Suite à ces résultats, nous conclurons sur le travail réalisé sur les performances obtenues, la valorisation des résultats ainsi que les perspectives possibles pour ce type d'architecture.

Chapitre 2

Contexte et l'état de l'art

2.1 Introduction.

La sécurisation des communications est devenue un élément de plus en plus important dans le quotidien, pour l'identification, l'accès ou l'échange d'informations avec des dispositifs portables.

Initialement focalisé dans un contexte militaire, la problématique vécue par les systèmes de télécommunications, notamment sur la protection des communications lors de la deuxième guerre mondiale, manifestera un parallèle dans les systèmes embarqués miniaturisés quelques décennies après. La mission principale est, en tout temps, de masquer les données avant la transmission dans un milieu de propagation (filaire ou radio fréquences) en évitant toute possibilité d'interception avec les messages d'un tiers. Ils existent deux possibilités pour protéger les messages :

- l'altération de la structure des messages,
- l'altération du canal de communication ou des signaux accessibles du système.

La première technique, implique l'utilisation d'algorithmes cryptographiques embarqués sur les composants. De tels algorithmes, de nature séquentielle, permettent d'altérer une séquence de données de façon à ce qu'elle ne soit pas directement exploitable. La fiabilité dépend de la robustesse des algorithmes.

La deuxième technique est centrée sur l'altération des signaux physiques générés comme conséquence de la génération et transmission de l'information. Dans ce cas, leur robustesse dépend du niveau d'aléa injecté sur les signaux d'intérêt.

La croissance de la puissance de calcul, ainsi que l'évolution de la cryptographie et cryptanalyse ont permis d'améliorer les algorithmes de cryptage et les techniques de validation de leur robustesse, ainsi que le perfectionnement sur les techniques d'ingénierie inverse et interception des signaux exploitables.

Mais au delà de l'évolution des techniques de calcul, les supports (comme les voies de communication) permettant le transite de l'information n'ont pas subit la même évolution, tels que les radio fréquences ou les lignes d'alimentation du circuit électronique. D'où l'importance de la protection au niveau matériel, accompagnée d'algorithmes robustes, pour augmenter le niveau de fiabilité, contre attaques sur une

carte électronique.

Ce chapitre présente les cartes à puce, problématique sur laquelle se positionnera la sécurisation dans cette thèse, dans une perspective de masquage au niveau matériel. Le besoin d'altérer les données transmises, en utilisant un générateur aléatoire pour incrémenter la puissance des algorithmes de cryptage est exprimée, et les caractéristiques des générateurs aléatoires seront énoncés, et les différentes approches de conception seront présentées, pour finalement converger vers celles du système générique qui doit communiquer avec le générateur à concevoir.

2.2 Les cartes à puce.

Une carte à puce (*smart card*) est un type de carte contenant un système embarqué qui permet de stocker et transférer des données de l'utilisateur.
Ce système d'échange de données utilise une borne lecteur externe, qui récupère les identifiants de la carte et permettant de valider une transaction.

Le concept de ce type de système avait déjà été envisagé à la fin des années soixante. Entre 1974 et 1978, Honeywell développe le premier système de carte à puce, avec un microprocesseur et une architecture dédiée pour sa programmation.

FIGURE 2.1 – La télécarte.

La première application de ce système était le télé-payement. A partir de 1983, le développement massif de ce type de cartes répond à l'arrivée de la "télécarte", apparue dans le système de téléphonie publique française (fin de sa commercialisation étant prévue pour avril 2014).

En 1992, les *cartes à puce* sont intégrées dans les payements bancaires (*Carte Bleue*). De tel systèmes impliquent l'utilisation d'un code personnel pour chaque utilisateur (*code PIN, Personal Identification Number*), afin de rendre active toute transaction.

FIGURE 2.2 – Carte bancaire avec smartcard.

Ce système nécessite toujours un contact entre la carte et le lecteur. Plus tard, une variante sans contact est apparue, liée aux systèmes (*RFID*, Radio Frequency Identification). Dans ce cas, la carte à puce reçoit les informations et même l'énergie nécessaire pour l'alimentation des circuits à travers des ondes transmises par le lecteur.

2.2.1 Classification des cartes à puce.

Les systèmes de carte à puce sont classifiés en termes des fonctionnalités du système embarqué, ainsi que de la façon dont le système utilise ces dernières pour communiquer.
En fonction de leurs fonctions, nous avons deux variantes : cartes de stockage mémoire et cartes multifonctions :

Cartes de stockage mémoire.

Les cartes de stockage mémoire permettent de garder des données importantes avec une fonctionnalité robuste de validation/identification. Le transfert de données est géré par le lecteur. Possédant de la mémoire non volatile (flash-I^2C), elles peuvent avoir aussi un mécanisme de protection d'accès par mot de passe.

Elles peuvent avoir une fonction spécifique, comme dans le cas des cartes téléphoniques qui peuvent être rechargeables ou jetables à la fin de leur utilisation ; à chaque appel effectué, un espace mémoire est effacé.

Cartes multifonction.

Les cartes multifonctions intègrent un niveau plus élevé de sécurisation des transactions ainsi que la possibilité de gérer différents types d'informations d'accès sur un même système. Un exemple peut être une carte étudiante permettant d'avoir accès aux services numériques de la bibliothèque ainsi que la recharge des repas au restaurant universitaire.

FIGURE 2.3 – Carte à puce : carte bancaire

En terme de connectivités matérielles, nous pouvons distinguer (figure 2.4) :

Cartes à contact : ce type de cartes a besoin du contact physique avec le lecteur de cartes via des contacts électriques visibles généralement positionnés sur le dessus.

Cartes sans Contact : la puce utilise une antenne sur une surface souple à l'intérieur de la carte.

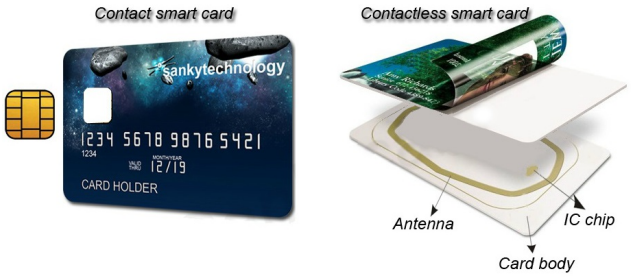

FIGURE 2.4 – Carte à puce avec et sans contact

Une antenne sur la carte permet au dispositif d'échanger des signaux avec un lecteur, via radio fréquences. Il est ainsi possible d'alimenter la carte ainsi que de générer des signaux pendant une transaction (système RFID, Radio Frequency IDentification, figure 2.5).

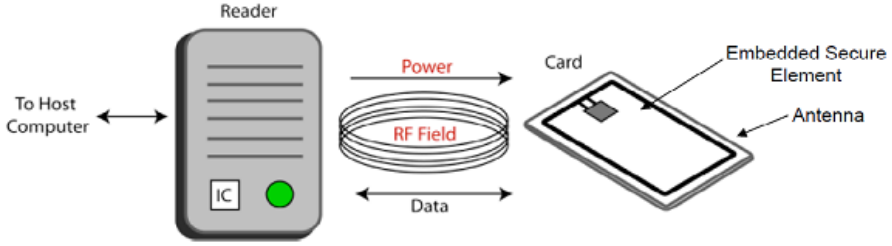

FIGURE 2.5 – Communication entre lecteur et carte RFID.

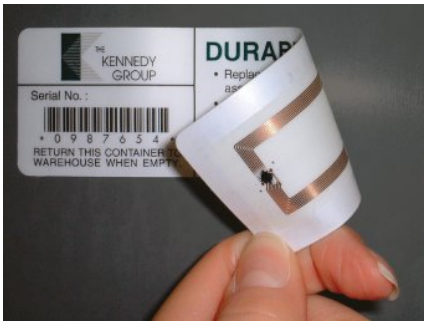

FIGURE 2.6 – Module RFID utilisé pour l'identification de produits.

2.2.2 Applications des systèmes à carte à puce.

Les applications de ce type de dispositifs (fig. 2.7), incluent les cartes bancaires, de téléphonie mobile, de télépéage, de badges personnels d'accès-identification, entre autres.

FIGURE 2.7 – Les applications des systèmes à carte à puce

Un grand consommateur de ce type de dispositifs est la téléphonie mobile, dont chaque terminal mobile est muni d'une ou deux cartes à contact, la carte *SIM*(Suscriber Identity Module) contenant les identifiants de l'abonné. Elles sont compatibles avec les modems d'accès à internet nomades ("clés 3/4G").

Les cartes bancaires représentent un groupe sensible de ce type de cartes, en terme de quantité sur le marché et l'importance des données stockées/échangées. Ils présentent plusieurs niveaux de sécurisation pour maximiser la fiabilité chez l'utilisateur. Cependant, il existe des outils qui permettent aux personnes malveillantes de réaliser des copies des cartes bancaires, à partir de données récupérées par l'utilisateur.

2.3 Sécurisation de l'information dans les cartes à puce.

Le coté pratique des systèmes à *carte à puce*, va s'accompagner de la nécessité de protéger les informations qui sont transférées d'un coté à l'autre durant une transaction quelconque.

Ce besoin se manifeste fortement pour des applications où sont transmises des informations personnelles, qui ne doivent pas être accessibles que par les entités concernées par la transaction (clé du compte bancaire, document d'identité civile, etc).

La difficulté consiste à implanter les systèmes de protection de la puce. Cette protection doit être dans les deux sens de communication (la puce, en tant que système embarqué, et le lecteur, avec ou sans contact) de façon à éviter le succès de tout type d'attaque de la part d'un utilisateur malveillant.

Un mécanisme de protection doit alors être ajouté au niveau physique sur les signaux qui sont transmis, afin d'éviter toute fuite d'information, vis à vis des techniques d'attaque existantes sur les circuits intégrés et notamment, sur les cartes à puce.

2.3.1 Techniques d'attaque.

Une attaque sur un système électronique consiste dans la récupération des signaux en utilisant différentes méthodes. Ceci est possible en ayant comme objectif l'extraction d'informations utiles ou réplication du comportement du composant d'origine, en violant la propriété intellectuelle et le caractère privé des informations gérées par le système électronique. Nous pouvons classifier les techniques d'attaque dans deux grands groupes : invasives et non invasives (figure 2.8).

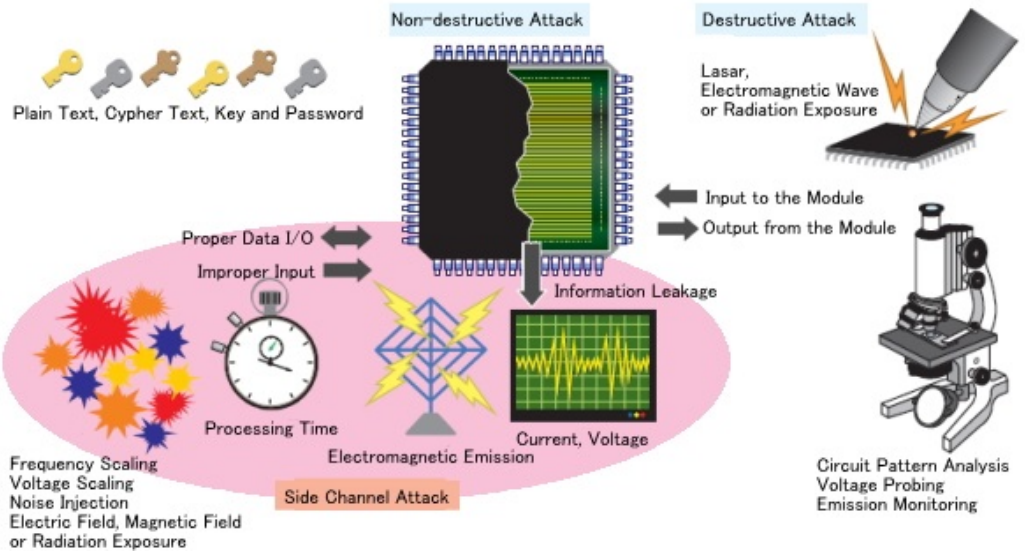

FIGURE 2.8 – Classification des techniques d'attaque sur les systèmes *smartcard*.

2.3.1.1 Techniques Invasives.

Ces techniques demandent une manipulation directe du matériel, impliquant une altération physique de sa structure. C'est le type de technique la plus coûteuse et complexe en terme de mise en place.

Un traitement corrosif (HNO_3) permet d'avoir accès à la puce électronique (fig. 2.9.a), qui permet de placer des micro sondes et capturer des signaux à l'intérieur de la puce (fig. 2.9.b).

Une autre variante de technique implique l'utilisation d'équipements spécialisés, tel que le *FIB* (Focused Ion Beam - Sonde Ionique focalisée), qui permet la modification du comportement des transistors de la puce via la variation des dépôts sur le matériel, ou simplement en changeant un état logique, à la recherche de désactiver le mécanisme de protection interne.

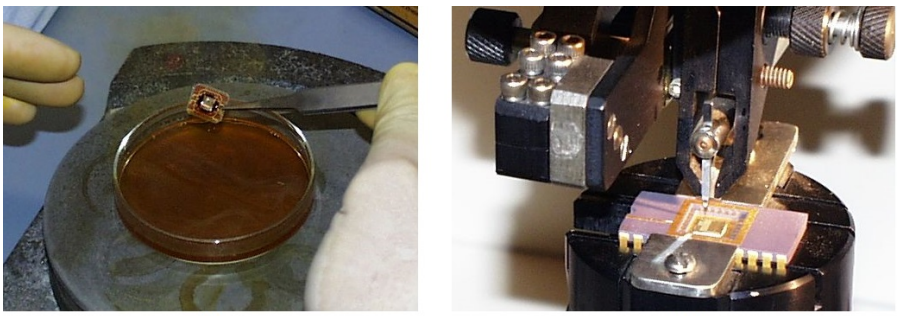

FIGURE 2.9 – Attaque invasif (a)corrosion du boitier. (b)insertion des sondes.

2.3.1.2 Techniques Non Invasives.

Ce type d'attaques essaie d'avoir accès localement au dispositif, mais sans aucun endommagement du type physique sur le processeur par contact électrique.

En utilisant l'observation-mesure de signaux générés par le système existant, puis l'analyse mathématique de l'information, il est possible de discriminer et extrapoler les caractéristiques des informations importantes transférées.

Le traitement de signaux, la statistique et cryptographie participent activement à ce type de technique.

Elles sont connues comme attaques du canal actif (*active side channel attacks*), vu qu'elles se réalisent sur le canal de communication des données. Nous pouvons citer, entre autres :

- ***DPA*** (Differential Power Analysis - analyse de puissance différentiel), qui permet de déduire les valeurs des clés cryptographiques en mesurant et analysant les variations de courant sur l'alimentation du système.
- ***DEMA*** (Differential EM radiation Analyse), est une variante de la DPA, qu'implique l'étude des variations des champs électromagnétiques rayonnés à partir de la puce, pendant les transitions logiques à l'intérieur de celui-ci.
- ***DFA*** (Differential Fault Analysis), concerne les stratégies qui forcent le système à exécuter des calculs erronés, en exploitant les résultats obtenus.
- ***Cryptanalyse Acoustique***, qui prend en compte les sons produits par le système pendant l'exécution d'une routine de calcul. Cette technique a permis [13], la réussite dans l'identification des patrons des clés générés par l'algorithme RSA de 4096 bits, en "écoutant" le processeur avec un microphone.

Les techniques non invasives sont les plus souvent utilisées pour réaliser des attaques sur des *cartes à puce*, en termes d'accessibilité aux signaux et réalisation basées sur le calcul mathématique, au contraire des méthodes invasives qui requièrent des équipements très coûteux.

2.3.2 Mécanismes de Protection.

Comme contre mesure, les concepteurs de circuits intégrés établissent des méthodologies pour masquer le plus possible toute trace contenue sur le canal de communication qui puisse faciliter aux attaquants la découverte du comportement du processeur et encore les codes de cryptage transmis. Du coté algorithmique, des techniques robustes sont implémentées pendant la transmission de données pour altérer la structure des données.

Dans le cadre du projet *PROSECURE 32*, un des moyens de protection réalisé utilise le masquage des informations sur l'alimentation du circuit, avec l'étude des régulateurs de tension programmables, leur ajout de sources d'aléa et un démarrage programmable des étapes.

L'autre fonctionnalité concernée dans la protection de données, vient de l'intérieur du processeur, dont il utilise des processus de cryptage de données. Le processeur nécessite une source de valeurs impossibles à prédire pour leur utilisation tant que les clés uniques d'initialisation des algorithmes de chiffrement (AES, RSA, etc) sont maintenues.

Pour tel motif, nous avons besoin d'un bloc générateur de valeurs binaires aléatoires fiable, capable d'assurer les deux types de techniques de masquage possibles, sans pour autant occuper une grande surface physique ou consommant une énergie plus grande que le reste du système cryptographique. Nous allons décrire le comportement et proposer une classification des générateurs afin de retenir une solution utile pour

notre système.

2.4 Les générateurs de valeurs aléatoires.

Les générateurs de valeurs aléatoires (*RNG - Random Number Generator*) produisent des valeurs qui ne suivent aucun ordre ou séquence en particulier.
La figure 2.10 illustre d'une manière simple l'utilité d'un RNG dans le processus de cryptage du message vis à vis d'un potentiel attaquant.
Le processeur utilise les valeurs du générateur aléatoire pour altérer la structure du message, de façon à ce qu'elles ne soient pas accessibles qu'au récepteur du message. Vu de l'extérieur, le message transmis est incohérent et nécessite de l'analyse pour son exploitation.

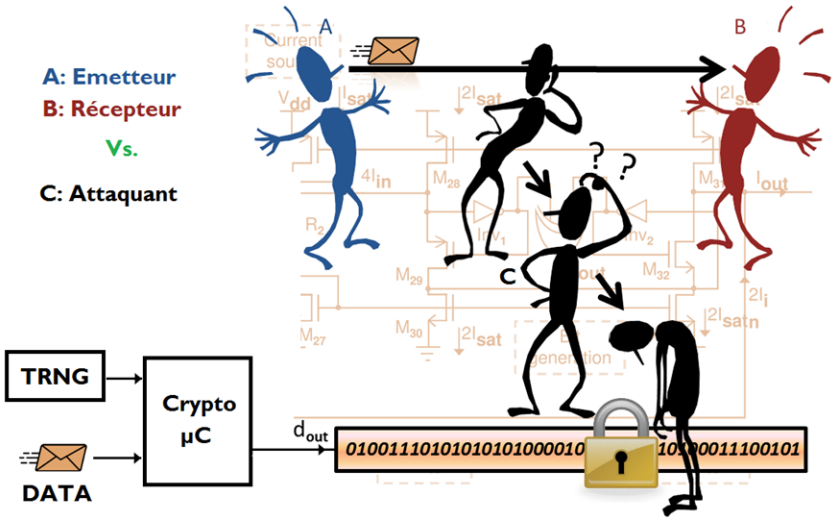

FIGURE 2.10 – Protection de données chiffrés en utilisant un TRNG.

Il est donc très important de garantir la bonne qualité des données fournies par le générateur aléatoire. Le niveau de non prédictibilité dépend de la source d'aléa utilisée sur ce générateur, ainsi que du traitement effectué sur les données avant son utilisation, ce qui définira les types de générateurs à étudier.

2.4.1 Sources d'entropie.

Nous pouvons les classer en deux catégories qui sont :

– **Les sources physiques**, dues aux caractéristiques intrinsèques des systèmes physiques. Par exemple : le bruit thermique (l'agitation moléculaire), systèmes chaotiques, la dégradation atomique, etc. Le niveau de non prédictibilité sur ces sources est très élevé.

– **Les sources non physiques** provenant des algorithmes mathématiques. Ce type de système, de nature séquentielle, essayent d'imiter le comportement des sources naturelles, mais avec moins de robustesse.

2.5 Types de générateurs aléatoires.

Le critère de classification des générateurs aléatoires coïncide avec celui des sources d'aléa. Il existe deux groupes :

Générateur de Valeurs Pseudoaléatoires, PRNG (*Pseudo Random Number Generator*), utilise des fonctions récursives arithmétiques-logiques de nature déterministe. Les données de sortie possèdent une distribution qui s'approche de l'aléatoire.

Son efficacité dépend donc de la robustesse de l'algorithme utilisé et de sa faible périodicité. Il est alors possible d'utiliser des techniques non invasives pour reconstruire le comportement des séquences et en conséquence l'algorithme générateur. Ceci justifie que ce générateur ne soit pas optimal comme bloc fondamental en cryptographie.

Générateur de Valeurs Purement Aléatoires - TRNG (*True Random Number Generator*), est un générateur plus robuste. Ils utilisent une source physique d'aléa, caractérisée par une distribution gaussienne.

2.6 Caractéristiques d'un *TRNG*.

Un TRNG doit générer des séquences aléatoires sous un format défini : entiers, binaires, caractères, etc. Dans notre contexte, nous allons parler plus précisément de séquences binaires. Ceci reproduit exactement le tirage d'une monnaie en l'air. Un tirage effectué a un instant donné, ne provoque aucune influence sur le tirage prochain. Chaque événement de la séquence obtenue, dit "aléatoire" possède donc une indépendance mutuelle. Ainsi, la probabilité d'avoir des '0' ou des '1' doit être très proche du 50 %. Malheureusement, du à diverses facteurs physiques, cette condition n'est pas obtenue directement. Le biais dans une séquence impliquera la présence d'un correcteur.

Un TRNG doit satisfaire deux propriétés fondamentales :

Posséder une distribution statistique uniforme, car une distribution non uniforme permet aux attaquants d'espérer des séquences répétitives à la sortie du générateur.
Contenir une source d'entropie, responsable de la caractéristique aléatoire, sujet à une étape de compensation du biais.

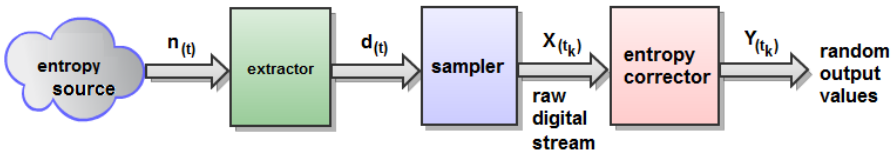

FIGURE 2.11 – Diagramme de blocs d'un RNG.

La figure 2.11 indique de façon très générique les blocs qu'intègre un TRNG :

Une source, caractérisé par des variations physiques intrinsèques aberrantes, non périodiques, nécessaires pour la génération de valeurs aléatoires.

Une extracteur, permettant de traduire les propriétés physiques vers des signaux électriques. Il ajoute aussi une limitation dans la dynamique et dans la bande passante du signal obtenu.

Un échantillonneur, permettant de récupérer les variations électriques sous la forme de données binaires.

Un correcteur d'entropie, permettant d'uniformiser la distribution binaire biaisée, afin d'avoir la même probabilité de 0's et 1's.

Après cette brève introduction, nous allons nous focaliser sur les TRNG et les techniques de conception associées. La partie suivante sera alors dédiée à la conception des TRNG, notamment d'un point d'implémentation analogique.

Les TRNG sont soumis à une quantité très variée de paramètres qui compliquent une classification uniforme de ses variables. Nous allons utiliser une taxonomie qui applique trois optiques différentes en fonction desquelles les diverses techniques et éléments qui composent les différents TRNG peuvent être analysés :

Paramètres de Qualité.

- Source d'Aléa.
- Méthode d'extraction de l'aléa.
- Méthode de post-traitement.
- Débit de sortie.

Paramètres de Sécurité.

- Existence d'un modèle mathématique.
- Capacité d'auto-test embarqué dans le générateur [1].
- Sécurisation (robustesse, résistance contre attaques).

1. Nous allons considérer la capacité d'auto-test embarqué dans le générateur comme optionnelle

Paramètres de Design.

- Utilisation de ressources.
- Consommation.
- facilité d'implantation.

2.7 Techniques utilisées pour la conception des TRNG.

Comme mentionné précédemment, le TRNG se base sur l'utilisation d'une source physique d'aléa. Dans le cas de l'implantation sur silicium, l'aléa est extrait des propriétés intrinsèques des semiconducteurs. Nous allons, dans cette section, nous focaliser sur les techniques usuellement utilisées et centrées sur un approche analogique.

2.7.1 Amplification directe du bruit.

La Fig.2.12 schématise le fonctionnement du système.

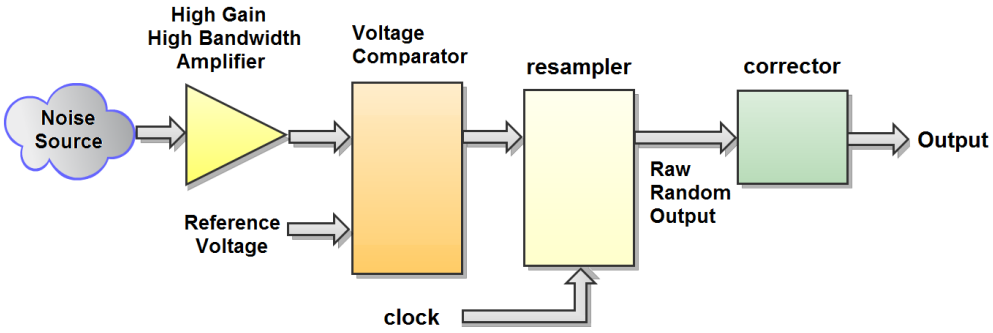

FIGURE 2.12 – TRNG basé sur l'amplification directe du bruit.

Paramètres de Qualité.

- *Source d'Aléa* : le bruit thermique, inhérent à la structure interne d'un composant (résistance, transistor, diode) [45].
- *Méthode d'extraction de l'aléa* : les variations dues au bruit thermique sont menées vers des variations de tension-courant et finalement amplifiées pour avoir une dynamique conséquente. Finalement ces signaux sont échantillonnés à 1 bit (figure 2.13).
- *Méthode de post-traitement* : due à la nature de la source ; très peu de post-traitement est requis.
- *Débit de sortie* : il dépend de la limitation de bande passante imposée par le circuit d'amplification. Toutefois, un circuit de cadencement est requis du aux variations instantanées observées.

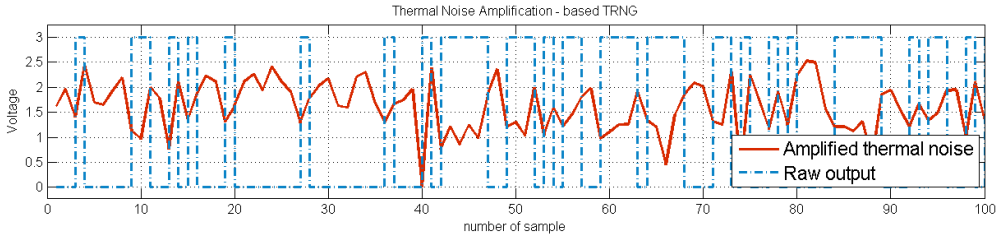

FIGURE 2.13 – Obtention de valeurs aléatoires à partir du seuillage de la sortie amplifié du bruit.

Paramètres de Sécurité.

- *Existence d'un modèle mathématique* : le bruit thermique est tout à fait modélisable ; il comporte des variations avec une distribution gaussienne. En fonction de la bande de fréquences exploitée, le bruit possédera des composantes fréquentielles dues à des effets très particuliers (bruit blanc, bruit de grenaille bruit rose , etc)
- *Sécurisation* : due à la nature de la source, il a un niveau élevé d'entropie et est assez robuste.

Paramètres de Design.

- *Utilisation de ressources matérielles* : afin d'amplifier au maximum toute la bande utile, le bloc amplificateur possède un produit gain-bande très élevé. Le signal amplifié est ensuite échantillonné à 1 bit, avec un comparateur de tension capable de gérer la gamme de fréquences souhaitées pour le débit binaire de sortie, (figure 2.13). Finalement une étape de synchronisation et correction d'entropie est utilisée pour améliorer la distribution binaire.
- *Consommation* : ce type de circuits nécessite des niveaux de tension et de courant qui dépassent les expectatives du *low power*, allant vers mA.
- *Implantation sur Silicium* : ce type de circuits est très facile en terme de réalisation via l'utilisation de composants discrets. Cependant, pour une réalisation en intégré, il est déconseillé car l'inconvénient le plus pénible vient du fait qu'il faut isoler l'élément générateur de bruit du reste du circuit pour éviter une pollution du reste des signaux du système.

2.7.2 Échantillonnage des Oscillateurs.

Cette méthode est utilisée tant dans les réalisations numériques qu'analogiques. La figure 2.14 illustre le principe d'opération utilisé.

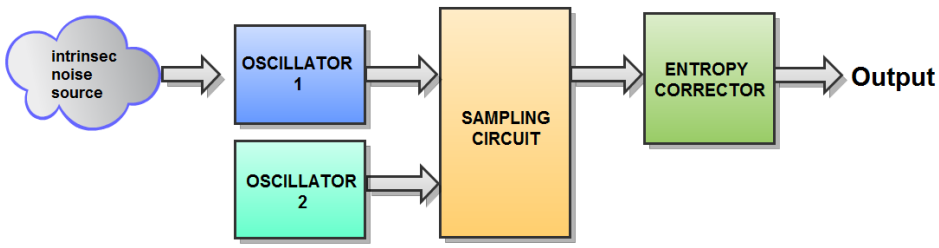

FIGURE 2.14 – TRNG basé sur l'échantillonnage d'horloges.

Paramètres de Qualité.

– *Source d'Aléa* : il repose sur un phénomène caractéristique des signaux périodiques réels, appelé *jitter*. Le *jitter* représente les variations observables de la période du signal autour de sa valeur nominale (figure 2.15.a). La superposition des créneaux, (*diagramme d'oeil*) (figure 2.15.b) montre que la période n'est pas tout à fait régulière, mais dispose de faibles variations.

Un histogramme de telles variations, (fig. 2.15.c,d) nous permet de quantifier les variations ainsi que leur caractère aléatoire.

Contrairement aux applications de communications de données, le but ici est d'augmenter le *jitter* afin d'avoir une plage aléatoire plus grande.

– *Méthode d'extraction de l'aléa* : il utilise dans son fonctionnement deux oscillateurs. L'un des oscillateurs présente un niveau de *jitter* plus élevé par rapport à l'autre, de façon à ce que l'échantillonnage d'un des signaux par rapport à l'autre provoquera l'obtention de résultats non périodiques. Finalement, un circuit correcteur nous permet de corriger tout défaut dans la distribution obtenue.

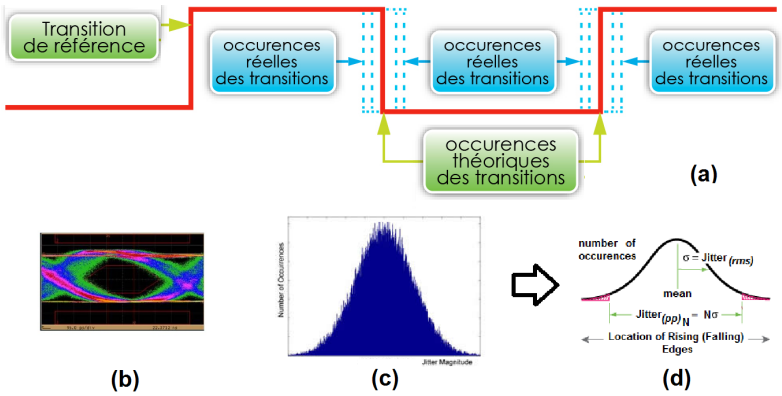

FIGURE 2.15 – (a)Le *jitter* (b) diagramme d'oeil (c) histogramme des variations de la période (d) analyse du *jitter* à partir de l'histogramme.

Possédant par nature un niveau de *jitter* bas, il est nécessaire d'incrémenter le nombre d'éléments de l'anneau ou d'augmenter la fréquence de l'oscillateur "rapide" afin d'assurer un taux d'incertitude satisfaisant d'un échantillon.
- Le niveau du *jitter* des oscillateurs en anneau, impose que le deuxième oscillateur soit à une fréquence très élevée.
- *Méthode de post-traitement* : une solution [43],[9] pour compenser cet effet sur la sortie nécessite l'utilisation d'un correcteur tel qu'un extracteur *Von Neumann* ou un PRNG [8].
- *Débit de sortie* : dépend des oscillateurs et du traitement utilisé.

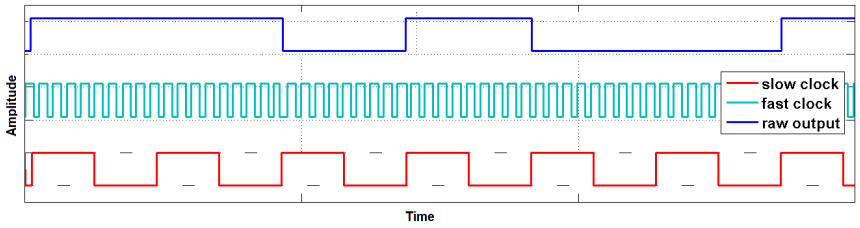

FIGURE 2.16 – Obtention de valeurs aléatoires à partir de l'échantillonnage d'horloges.

Paramètres de Sécurité.

- *Existence d'un modèle mathématique* : le *jitter* dans ce type d'architecture est une conséquence indirecte du bruit intrinsèque des composants sur le temps de propagation sur la boucle de l'oscillateur. Donc nous pouvons utiliser un modèle de distribution gaussienne pour caractériser le *jitter* et l'identifier dans le domaine fréquentiel via du bruit de phase.

Paramètres de Design.

- *Utilisation de ressources matérielles* : les oscillateurs généralement retenus [1] [33][36] sont basés sur des topologies du type anneau, basés sur des variantes d'inverseurs logiques. L'avantage de ce type de topologie repose sur sa simplicité en terme de réalisation, disposant d'une fréquence d'oscillation dépendante du nombre d'éléments de l'anneau, du courant de polarisation et de la charge de sortie. Puis le circuit de post-traitement est entièrement numérique donc synthètisable avec des outils logiciel.
- *Consommation* : l'absence de blocs de gain permettent de réduire la consommation totale. La réalisation d'un oscillateur de fréquence très élevée peut augmenter la consommation du circuit.
- *Implantation sur Silicium* : afin d'obtenir un signal "rapide", certains systèmes RFID récupèrent le signal de la porteuse [4], entraînant une réduction en surface du circuit de génération d'horloge mais également en terme de puissance consommée. Si cette solution n'est pas viable, un compromis de consommation et bande passante doit être pris en compte pour la conception de l'oscillateur

'rapide'.

Enfin, le *jitter* peut être incrémenté en augmentant la complexité du circuit. Des stratégies possibles sont la synthèse d'oscillateurs en cascade [2], ou l'addition de blocs de retard programmables sur le chemin de propagation de la boucle opérationnelle.

– Les réalisations en ASIC numérique [48] sont sensibles aux variations d'entropie dépendantes de la température et l'alimentation, donnant des possibilités aux attaques non invasives.

2.7.3 Fonctions Physiquement Non Clonables - *PUF*.

Les *PUF* (Physically Unclonable Functions) sont des blocs fonctionnels, qui produisent différents résultats (response) en fonction d'une même entrée (challenge), sur les mêmes conditions imposées sur deux éléments identiques en architecture (figure 2.17).

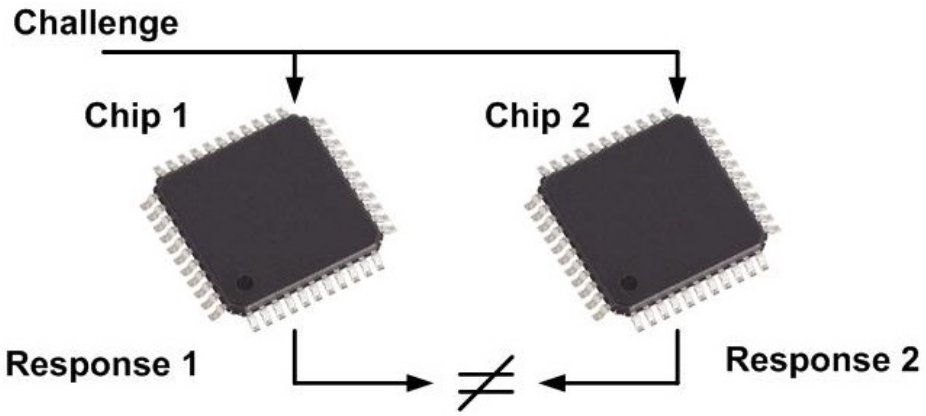

FIGURE 2.17 – Idée générale du fonctionnement des PUFs.

Paramètres de Qualité.

– *Source d'Aléa* : ce comportement répond à la sensibilité aux variations sur le chemin de propagation du signal, occasionnée par les différences aléatoires des substrats dans les étapes de fabrication entre deux composants. Cette différence individuelle représente une "empreinte personnelle", une signature, aléatoire caractéristique de chaque circuit.

– *Méthode d'extraction de l'aléa* : les *PUF* sont classées sous deux approches :

1. ***PUFs* basées sur le retard de propagation :**
   APUF (*Arbiter APUF*) : un arbitre (figure 2.18(a,c))[41] qui décide l'origine du premier signal entrant. En effet, le signal provenant d'un

même générateur, traverse deux chemins de propagation physique constitués par les interrupteurs programmables par b[0 :n], (Fig. 2.18(b)). Le chemin et la durée de propagation du signal est définie par l'état initial du mot de commande de ces interrupteurs.

ROPUF-Ring Oscillator PUF : le *jitter* associé aux oscillateurs en anneau [18] est considéré comme une variation aléatoire sur le temps de propagation du et unique de chaque oscillateur.

La synthèse synchrone additive (opérateur *XOR*) de plusieurs oscillateurs en anneau (figure 2.19), implique que le niveau de *jitter* de sortie est incrémenté de manière proportionnelle au nombre d'oscillateurs placés en entrée.

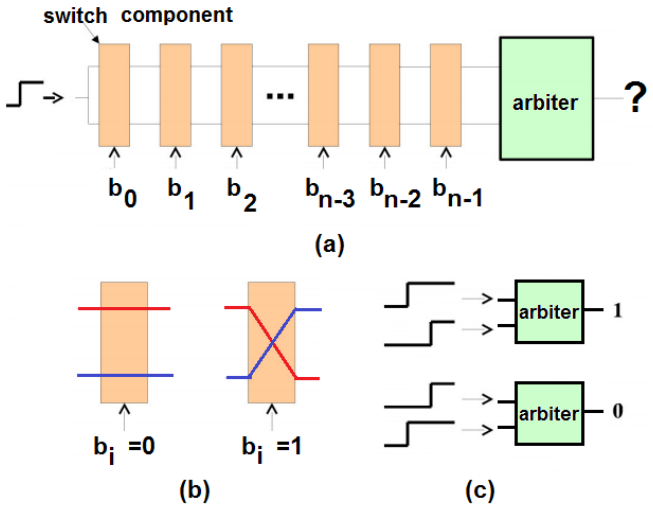

FIGURE 2.18 – *APUF* (a) schéma général ,(b) chemins de propagation,(c) opération de l'arbitre.

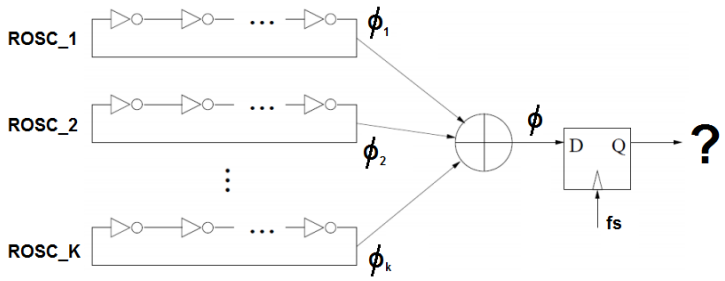

FIGURE 2.19 – schéma de fonctionnement d'un *ROSCPUF*.

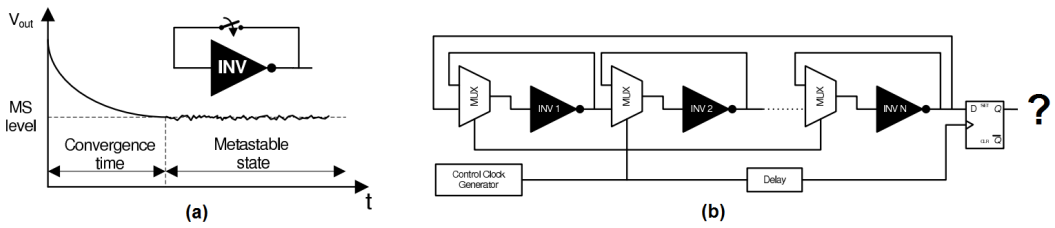

FIGURE 2.20 – *PUF* à métaoscillateur : (a) méta-inverseur (b) schéma équivalent du système .

Métaoscillateur : une autre variante [47], utilise l'incertitude sur l'état du signal, durant la stabilisation d'une porte logique à partir du court-circuit entre l'entrée et la sortie (figure 2.20). Le temps de stabilisation de chaque porte est donc une caractéristique intrinsèque. Le contrôle individuel de chaque bloc intégrant un oscillateur en anneau permet de gérer son comportement, notamment en termes du *jitter*.

2. ***PUF* basées sur l'effet mémoire :** possèdent un comportement basé sur l'état antérieur des blocs de mémorisation. Les blocs de mémorisation peuvent se présenter sous la forme de :

Flip-Flops et Latches : ils utilisent les propriétés des systèmes bistables. Un exemple est représenté par le PUF papillon, BPUF (figure 2.21) [22]. Le BPUF est composé de deux *latches* reliés de manière croisée, avec le signal d'excitation arrivant sur un des *latches* via le signal de *clear* et sur l'autre via le signal de *preset*. L'état final du circuit est imposé par le retard entre les chemins de réalimentation du circuit (les "ailes du papillon"), qui introduit des états métastables sur le système.

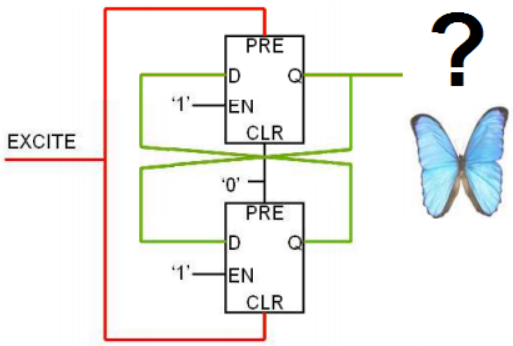

FIGURE 2.21 – PUF *"papillon"* .

SRAM : se base sur des structures métastables similaires aux *flip-flops*, avec un effet d'initialisation au démarrage ayant des états différents pour chaque SRAM.

Ces valeurs aléatoires sont souvent utilisées lors de la protection de données importantes stockées en mémoire plutôt que lors de la génération de valeurs aléatoires de façon continue.

– *Méthode de post-traitement* : les variations structurelles internes, incontrôlables par le *concepteur* garantissent un grand niveau d'aléa. Il est nécessaire d'ajouter un bloc d'extraction de la clé de taille fixe à partir des valeurs obtenues au démarrage du PUF.

Paramètres de Sécurité.

– *Existence d'un modèle mathématique* : la modélisation de ce type de circuits est très complexe et dépendante de plusieurs multiples facteurs aléatoires.
– *Sécurisation* : l'effort d'un attaquant par l'analyse mathématique augmente exponentiellement avec la complexité des structures [39].
Grâce à la manipulation des substrats par *FIB*, il est possible de faire une copie opérationnelle d'un PUF [14].

Paramètres de Design.

– *Utilisation de ressources matérielles* : les circuits sont moins complexes au niveau d'architecture, dépendant surtout des déviations de fabrication. La partie représentant la plus faible complexité est le circuit d'extraction statistique de la clé à partir des éléments aléatoires apparus au démarrage.
– *Consommation* : basse, optimisée par le processus et les blocs numériques simples.
– *Implantation sur Silicium* : ils sont faciles à synthétiser avec des cellules standards en ASIC numériques. Ils ont des structures sensibles aux variations de température, permettant de possibles attaques.

2.7.4 Échantillonnage de signaux chaotiques.

Ce type de circuits se base sur les systèmes chaotiques et leur non prédictibilité.

Paramètres de Qualité.

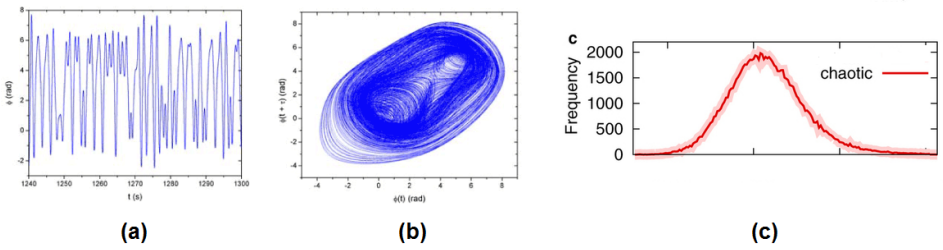

FIGURE 2.22 – Exemple d'un système chaotique : (a) sortie transitoire,(b)trajectoire chaotique,(c)histogramme du signal de sortie.

– *Source d'Aléa* : les variations non périodiques obtenues à la sortie d'un système chaotique possèdent une dynamique qui ressemble à celle du bruit. Par effet d'un élément non linéaire sur un oscillateur, cela permet de dégénérer le comportement périodique de base du système en fonction des paramètres de celui-ci. En altérant un ou plusieurs de ces paramètres, le système sort de son état d'oscillation harmonique (périodique), en faisant des trajectoires chaotiques autour d'une valeur mais sans jamais converger vers celle-ci.

  La réponse du système est très sensible aux variations des paramètres d'entrée, ce qui garantit une réponse différente à chaque démarrage.
– *Méthode d'extraction de l'aléa* : d'une façon similaire à l'échantillonnage du bruit, il requiert une étape de discrétisation et d'un correcteur dont les caractéristiques sont fonction de la dynamique de l'oscillateur chaotique.
– *Méthode de post-traitement* : il dépend de la dynamique de l'extracteur mais il présente un circuit simple, tel qu'un LFSR.
– *Débit de sortie* : il dépend du type de système chaotique utilisé, dans le cas d'un système à temps continu, dépend seulement de l'échantillonnage en sortie. Pour le cas d'un système en temps discret ou numérique, dépend des signaux de synchronisation du système.

Paramètres de Sécurité.

– *Existence d'un modèle mathématique* : il existent deux types d'implémentation de ce type de systèmes :

  En temps Continu : en obéissant un système d'équations différentielles.
  En temps Discret : en obéissant à des équations aux différences.

  Ces deux types d'implémentations sont associés à deux types de configurations avec des propriétés différentes en terme de consommation et contrôlabilité. La modélisation de ce type de systèmes est complexe et dépend du nombre de paramètres pris en compte dans la fonction caractéristique.

Paramètres de Design.

– *Utilisation de ressources matérielles* : peut se réaliser avec des blocs relativement simples, la cellule non linéaire étant l'élément le plus élaboré. Ce type de systèmes peut être réalisé tant en discret comme en intégré et en utilisant des systèmes analogiques-mixtes ou du numérique pur.
– *Consommation* : dépend de la topologie utilisée. Les implantations en temps discret peuvent être pilotables et représenter une basse consommation par rapport à leur contre-partie en temps continu [46].

L'une des limitations plus conséquente provient des variations fonctionnelles apparues pendant la réalisation matérielle. Ceci justifie l'ajout d'un bloc d'étalonnage et d'une analyse de calibration pour gérer les zones de passage vers le comportement chaotique afin de maîtriser leur fonctionnement.

Après avoir vu les différents systèmes dans une approche de réalisation analogique-mixte standard, nous allons indiquer les caractéristiques souhaitées pour le circuit à concevoir, de façon à converger en utilisant les mêmes critères indiqués antérieurement pour décider de l'architecture à étudier.

2.8 Caractéristiques du Système à Concevoir.

Maintenant, nous allons définir les caractéristiques à satisfaire pendant la conception-réalisation du système. La mise en oeuvre du circuit doit donner la priorité à une réalisation en analogique. L'IP à concevoir, d'un point de vue générique possède les caractéristiques suivantes :

- Tension d'alimentation : 3.3V.
- Adaptable à un processeur RISC 32 bits.
- Débit du RNG requis : supérieur à 50Kbps.
- Taille d'une trame à générer : au 32 - 64 bits par trame générée.
- Le générateur doit être construit de façon à rester le plus indépendant possible du reste du circuit (consommation minimale souhaitée et-ou désactivation externe du système).
- Réalisation en technologie standard.
- Le générateur doit passer des tests de vérification d'aléa standard.

Le tableau 2.1 résume les différentes caractéristiques des implémentations des TRNG dans la littérature, vis-à-vis de l'état de l'art des solutions analogiques sur technologie standard de fabrication.

L'étude des architectures implique l'analyse des techniques mais aussi des caractéristiques et résultats obtenus (pas cités en détail dans toutes les publications). Cela représente un compromis entre la technologie de fabrication, le contexte d'application et la consommation.

Author	Principle	Power	Bit Rate	Tech.	V_{DD}
Balachandran [4]	Clock Sampling	550nW	320 kbps	0.13μ	1.25V
Holleman 1 [16]	metastability + FIR filter	180 μW	50 kbps	0.35μ	5V
Holleman 2 [16]	metastability + DC filter	2.92 μW	0.5 kbps	0.35μ	5V
Holleman 3 [16]	metastability + DC filter	9.39 μW	5 kbps	0.35μ	5V
Nakura [34]	metastability + clock sampling	— μW	500 kbps	0.18μ	1.8V
Amaki [1]	clock sampling	— μW	—	65n	1.2V
Petrie [37]	noise sampling	3.9 mW	1.4Mbps	2μ	5V
Chen [8]	clock sampling	1.04uW	40 kbps	0.18μ	0.8V
Zhou [33]	noise amplif+ clock sampling	0.91uW	100 kbps	0.35μ	1.3V
Tavas [46]	cont. time chaos	35mW	2 Mbps	0.35μ	$\pm$1.65V
Beirami [6]	disc. time chaos	—	—	0.13μ	—

TABLE 2.1 – Liste des différents TRNG dans la littérature.

2.8.1 Discussion et quantification des différentes solutions.

Comme nous l'avions introduit en section 2.6, les différentes techniques présentées dans le tableau 2.1 ont été étudiées et comparées. Ainsi les résultats sont présentés ci-dessous dans l'optique d'introduire la solution retenue.

A- Paramètres de Qualité.

Source d'Aléa.

Dans l'amplification du bruit thermique, les variations sont prises directement de l'élément bruité. Elles sont de faible amplitude, mais avec une caractéristique gaussienne. Dans l'échantillonnage d'horloges, le *jitter* des oscillateurs en anneau possède des variations très faibles par rapport à la période du signal. Les PUF, dues à la nature aléatoire des variations matérielles, contiennent une forte source d'aléa.

Pour un système chaotique échantillonné, la source d'aléa dépend de la topologie du système non linéaire qui fait office de source de bruit, de dynamique restreinte mais moins limité en bande passante-consommation.

Méthode d'extraction de l'aléa - post-traitement.

Dans l'amplification directe du bruit, l'obtention d'un signal exploitable dépend de l'étage d'amplification - bande passante requise avant échantillonnage. Nous nous permettons de rejeter cette solution, due à la restriction sur la consommation imposée par notre cahier des charges, ainsi qu'à l'inconvénient lié à l'implantation matérielle sur silicium, risquant de polluer le reste du circuit avec la source de bruit. L'analyse donc continuera sur d'autres solutions.

Dans l'échantillonnage d'horloges, l'aléa, qui se manifeste comme du *jitter*, permet de capturer un échantillon de valeurs inconnues sur un oscillateur très rapide.

Étant donné (dans notre contexte) l'impossibilité de récupération d'un signal externe (une porteuse HF par exemple), la présence d'un oscillateur plus rapide impose un incrément sur la consommation dynamique du circuit. L'utilisation d'un système à deux oscillateurs aux fréquences proches sous un arbitre requiert de l'incrément du *jitter*, ceci étant associé à une augmentation de la complexité du circuit.
Le traitement du signal obtenu s'effectue de façon numérique pour compenser le faible jitter des oscillateurs.

Concernant les PUF, le système doit effectuer une extraction statistique des variations des éléments lors du démarrage du circuit. Ce type de structures est utilisé pour générer un code initial unique. Le cas basé sur la propagation dans les oscillateurs en anneau est très ressemblante au cas d'échantillonnage d'horloges. Étant garanti d'un fort niveau d'aléa, le correcteur n'est pas nécessaire et il est implicite

dans l'extracteur statistique. Il fournit une sortie numérique.

Dans le cas d'un circuit chaotique, le comportement aléatoire est contenu dans la dynamique de sortie de l'oscillateur, de nature analogique. Il n'a pas besoin d'une étape d'amplification de signal, pouvant relier un échantillonneur directement pour convertir les variations vers des suites binaires. Le besoin du post-traitement dépend de la caractéristique de la quantification et de la dynamique de l'orbite.

B- Paramètres de Sécurité.

Existence d'un modèle mathématique.

Dans le cas des solutions à base d'oscillateurs périodiques, le jitter, provenant du bruit thermique des composants, suit une distribution gaussienne. Il est donc possible, de réaliser un modèle comportemental des horloges incluant les variations dans la période et faire dépendre ces variations en fonction du nombre des caractéristiques des oscillateurs en anneau (temps de propagation plus bruit thermique).

Les PUF, possèdent une modélisation mathématique très complexe, dépendante d'un nombre très élevé de facteurs, spécifiques à chaque technologie de fabrication utilisée plus les effets de métastabilité du démarrage.

Les systèmes chaotiques possèdent aussi une grande complexité, dépendant de nombreux paramètres. Cependant, il existe la possibilité de simplifier l'analyse sous un nombre réduit de variables observées, applicable aux systèmes de temps continu et discret. De telles approximations correspondent à des projections à 1,2,3 dimensions effectuées sur des espaces encore plus complexes, du type multidimensionnel.

Dans les systèmes à temps continu, la modélisation s'effectue en terme d'équations différentielles. La méthodologie formelle de solution de ces systèmes ajoute un degré de complexité évident. Les systèmes à temps discret remplacent cette représentation par des équations aux différences, ce qu'implique une récursivité dans l'évolution du système. Dans les deux cas, existe une forte dépendance par rapport à l'élément non linéaire inclus dans le système. Cette présence garantit le passage vers le chaos.

Capacité d'auto-test embarquée dans le générateur.

Ce bloc du système est moins cité et il caractérise normalement des solutions numériques. Nous n'allons pas le traiter dans notre contexte tant que nous effectuons encore une validation d'une architecture. A défaut, cet élément peut s'implanter algorithmiquement dans le processeur externe pour gagner de la place.

Sécurisation.

Les deux conditions à valider dans ce contexte sont la protection des signaux contre des attaques non invasives et le passage des tests d'aléa standard.

Dans le cas des attaques malveillantes, les variables communes entre toutes les solutions sont la température et la tension d'alimentation, en jouant sur ces paramètres il est possible d'obtenir une information importante du comportement du circuit. Dans le cas des PUF, ce fait est moins aggravant car ces variations continuent à générer, dans l'ensemble, des patrons aléatoires.

Dans les PUF, il existe aussi un risque , moins proche, de le réplication matérielle.

Les systèmes chaotiques possèdent un avantage, notamment dans le cas des systèmes à temps discret, où nous pouvons utiliser un réglage programmable pour, soit désactiver l'oscillateur, soit le rendre périodique. En faisant cette reconfiguration , le système peut être masqué sur certains périodes d'inactivité.

2.9 Compromis entre les ressources et facilité d'implantation

Comme nous l'avons vu précédemment, le besoin d'avoir un bloc indépendant du reste du circuit impose de simplifier l'architecture, donc toute réutilisation de signaux externes d'haute fréquence n'est pas permise. Cependant des niveaux logiques de configuration et pilotage du bloc sont acceptés.

Les architectures basées sur des oscillateurs possèdent des caractéristiques intéressantes en termes de simplicité, laissant la partie plus large sur le post traitement. Les systèmes chaotiques, par contre, peuvent utiliser des éléments qui peuvent augmenter la complexité (notamment les réalisations en temps continu) avec des éléments qui sont couteux en terme d'implantation (inductances par exemple), qui doivent être remplacés par leurs équivalents dynamiques.

Les systèmes en temps continu possèdent les mêmes avantages des systèmes sous oscillateurs, donc des blocs simples autour d'une fonction non linéaire qui peut être choisi parmi une grande variété de gabarits à implanter.

Les PUF sont dépendants des variations des processus de fabrication, donc nous n'avons pas vraiment une assurance entre le portage d'une configuration entre deux technologies différentes. Il existe en même temps un bloc d'analyse et extraction qui n'est pas négligeable en complexité embarquée avec ces éléments.

2.9.1 Conclusion.

Au terme de ce chapitre, après une description des différents types de supports (de carte à puce), comme des différents aspects sécuritaires embarqués sur ces derniers, l'importance des TRNG a semblé comme étant incontournable. Les TRNG dont les implémentations sont variables en fonction de l'application souhaitée, une étude des paramètres de quantification a permis de trancher sur les avantages et inconvénients de chacune d'elles.
Ainsi, après une évaluation des paramètres vis-à-vis des contraintes que l'on s'était posé, nous allons pousser l'étude vers les systèmes basés sur les oscillateurs chaotiques à temps discret qui seront détaillés dans le chapitre suivant. Les avantages de simplicité des architectures,de la facilité d'implantation sur silicium dans une technologie donnée et de contrôlabilité de l'aléa, qui ressemblent au bruit mais sans les contraintes d'amplification, justifient ce choix.

Chapitre 3

Systèmes dynamiques et chaos

Introduction

L'étude des modèles mathématiques des phénomènes physiques remonte au temps de Galilée, qui proposa un modèle pour la chute des corps et le mouvement de la terre autour du soleil. La présence de la variable *temps* permet de parler des systèmes dynamiques.

Au XVIII siècle, Sir Isaac Newton établit le déterminisme sur les phénomènes physiques où tout devient parfaitement prédictible et causal. Muni des équations différentielles du modèle, par exemple, il est possible de représenter et estimer la trajectoire d'un corps dans le temps.
Une fois appliquée avec succès sur d'autres phénomènes physiques, cette approche finie par imposer pendant longtemps l'idée dogmatique d'un monde parfait, régit par les mêmes lois et donc prédictif. Lié à cette idéologie, tout phénomène proche du hasard apparaît comme un système que l'on n'était pas encore capable de modéliser.

Au début du XXème siècle, le mathématicien français Henry Poincaré, en analysant la modélisation des mouvements des corps célestes, met en évidence que la physique de Newton, n'est pas tout à fait une règle générale applicable à tout phénomène. Il constate l'existence de solutions particulières et apériodiques pour des trajectoires. Beaucoup plus tard ces travaux donneront naissance à la théorie du chaos.

Figure 3.1 – (a)Sir Isaac Newton,(b)Henri Poincaré,(c)Edward Lorenz.

Dans les années soixante, Edward Lorenz, professeur de mathématiques au **MIT** (*Massachusetts Institute of Technology*) s'intéresse aux phénomènes météorologiques. De tels phénomènes répondent à un nombre très conséquent de variables. Cependant, avec une approche newtonienne, il est possible de simplifier le calcul en considérant un nombre 'compacte' de variables, tout en gardant des équations complexes. Il

utilisa ainsi un modèle du système à trois équations - trois inconnues, de résolution par calcul qui restait impossible à faire à la main. Ainsi, muni d'un calculateur numérique[1], il commença le traitement de données historiques pour trouver une approximation numérique aux solutions des équations modélisées.

C'est dans ce contexte qu'il découvrit le chaos, d'une manière très fortuite, à cause de la précision d'un calculateur. En effet, un phénomène qui suit une règle de fonctionnement, peut avoir une variation des résultats en jouant sur la précision de la résolution du calculateur, mais en gardant une notion de convergence vers une valeur connue.

Ainsi, Edward Lorenz réalisa une série de calculs en modifiant la résolution, de six à trois chiffres décimales, dans le but de gagner du temps de calcul tout en gardant une précision tolérable. Lorsqu'il compara les deux séries de résultats, il obtint des résultats totalement différents, il imagina en tout premier lieu qu'il existait un dysfonctionnement dans l'ordinateur. Faux !

Il venait de constater le comportement chaotique d'un système non linéaire, où les infimes différences dans les conditions initiales entraînaient des résultats complètement différents. Cette théorie sera baptisée plus tard comme *la théorie du chaos*[2].

Un autre aspect intéressant a également été observé par Lorenz via une représentation graphique des solutions d'un système d'équation ; les orbites obtenues sont caractéristiques d'une classe de systèmes. Partant des conditions initiales légèrement différentes, les courbes liées au système divergent progressivement mais conservent une trajectoire avec une forme globale similaire. Dans le modèle météorologique, les courbes obtenues ressemblent aux ailes déployées d'un papillon (figure 3.2).

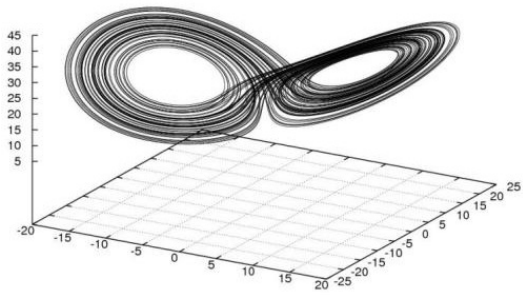

FIGURE 3.2 – L'attracteur de Lorenz ou "papillon".

Lorenz emploie une métaphore qui exprime clairement l'effet de la sensibilité aux conditions initiales et contribue au succès médiatique de la théorie du chaos :
"... le simple battement d'ailes d'un papillon au Brésil pourrait déclencher

1. le *Royal McBee LGP-300*
2. par les mathématiciens Li Tien-Yi et James A. Yorke, "Period three implies chaos" 1975.

une tornade au Texas ... "
A la fin du XX^{ieme} siècle début du XXI^{ieme}, l'étude du chaos s'est étendue à toutes les disciplines, due à leur omniprésence dans les phénomènes physiques.

Ainsi, par exemple, en électronique, les oscillateurs chaotiques sont utilisés dans la réalisation des générateurs de bruit, ainsi que des systèmes à automates à réseaux de neurones. Dûs à leur nature erratique et non prédictible, mais restreints sur une marge dynamique, ils sont des sources d'entropie facilement réalisables avec des circuits électroniques.

Ce chapitre introduit les concepts élémentaires des systèmes dynamiques. A partir des caractéristiques de ceux ci, une étude des systèmes chaotiques d'un point de vue générique sera présentée avec une mise en avant des solutions à temps discrets. En effet, ces derniers seront à la base de la source d'entropie requise pour la réalisation d'un générateur aléatoire.

3.1 Systèmes dynamiques.

Un système dynamique est la représentation dans le temps d'un phénomène physique quelconque, qui répond à une ou plusieurs entrées en interaction avec un ensemble de variables, avec une ou plusieurs sorties. Après une étude des différents types de systèmes dynamiques, nous nous attarderons sur la représentation de ces derniers.

3.1.1 Classification des systèmes dynamiques.

Les systèmes dynamiques se classifient en fonction de leur façon d'évoluer dans le temps. Ainsi, il existe trois types de systèmes dynamiques :

- ***Les systèmes stochastiques***, qui évoluent au hasard dans tout l'espace sans qu'aucune équation ne les régissent. Aucune prévision exacte dans le temps n'est possible.
- ***Les systèmes déterministes***, régis par des lois mathématiques bien connues et dont on peut donc prévoir exactement leur l'évolution dans le temps.
- ***Les systèmes chaotiques***, qui possèdent un comportement complexe. Ils sont irrésistiblement attirés par une caractéristique géométrique de structure également complexe, sur laquelle ils se comportent erratiquement, au hasard, mais sans jamais la quitter, ni repasser deux fois par le même point.
  Ces systèmes, semblent suivre à la fois des lois déterministes et des lois aléatoires, ce qui rend toute prévision à long terme impossible.

Parmi ces systèmes, nous pouvons retrouver une bonne partie des systèmes physiques courants (calcul de trajectoires : d'un élément mobile, d'une planète, etc...) ;

mais on y retrouve aussi l'évolution des populations, des automates cellulaires, la météorologie et bien d'autres.

3.1.2 Représentation des systèmes dynamiques.

Partant de ces différentes classes de systèmes dynamiques, nous pouvons représenter l'évolution d'un système de deux formes possibles :

- systèmes en temps continu
- systèmes en temps discret

Une étude de ces deux formes sera présentée ci dessous.

3.1.2.1 Systèmes en temps continu.

Les systèmes à temps continu sont caractérisés par l'utilisation d'équations différentielles décrivant l'évolution des variables dans le temps.

Les équations utilisées, reposant sur une approximation au premier ordre, possèdent la forme :

$$\dot{X} = \dot{X}_{(t)} = f_{(X;t,v)} \tag{3.1}$$

avec $X_{(t)}$ représentant l'évolution du système dans le temps et $\dot{X}_{(t)}$ correspondant à l'état instantané du système. La fonction $\boldsymbol{f}$ dépends du temps, ainsi que des paramètres du système, v. Dans un système à N variables, l'expression (3.1) devient :

$$\begin{aligned} \dot{X_1} &= f_1(X_1, X_2, ..., X_N; t, v) \\ \dot{X_2} &= f_2(X_1, X_2, ..., X_N; t, v) \\ \vdots &\ \vdots\ \vdots \\ \dot{X_N} &= f_N(X_1, X_2, ..., X_N; t, v) \end{aligned} \tag{3.2}$$

Dont le système, X_1,...,X_N possèdent des conditions initiales connues x_{1o},...,x_{No}.

3.1.2.2 Systèmes en temps discret.

Pour les systèmes à temps discret, le système est décrit en utilisant une modélisation dont les instants sont répartis dans le temps de façon équidistante.
Afin de répondre aux critères de discrétisation du système dans le temps, deux possibilités s'offrent à nous :

1. les caractéristiques du système imposent leurs caractères discrets,
2. le système est une version échantillonnée d'un système en temps continu.

Mais dans les deux cas, leurs représentations mathématiques utilisent des fonctions de récursivité. Une mise en équation via un système de premier ordre, est caractérisée de la façon suivante :

$$X_{(n+1)} = f_{(X_n)}, n \geq 0 \tag{3.3}$$

Avec une condition initiale connue $X_{(0)} = x_0$.

Dans le cas d'un système d'ordre supérieur ($r \geq 2$) :

$$X_{(n+r)} = f_{(X_n, X_{n+1}, \ldots, X_{n+r-1})}, n \geq 0 \tag{3.4}$$

3.1.3 L'espace des phases.

Étant donné l'implémentation possible d'un modèle décrivant l'évolution temporelle d'un système à partir des équations caractéristiques, nous pouvons donc construire un ensemble de variables d'état associé à ce système.

Un espace mathématique, appelé *espace des phases* est affecté à l'ensemble des variables d'états précédemment introduits. Cet espace mathématique est donc multidimensionnel et constitue toutes les valeurs possibles que peut prendre la sortie d'un système en fonction de ses paramètres.

3.1.4 Le portrait de phases.

C'est une représentation géométrique de la dynamique d'un système. Il peut s'agir d'une représentation uni, bi, ou tridimensionnelle, voire même multidimensionnelle.

Chaque axe compris dans le portrait de phase correspond à une variable d'état à un instant donné. L'ensemble de points représentant l'évolution du système génère des trajectoires caractéristiques du système. En utilisant le portrait de phase nous pouvons identifier l'attracteur correspondant à un système.

3.1.5 Les attracteurs.

Nous appelons *attracteur* la région de l'espace de phases précis d'un système, vers laquelle le système évolue de façon irréversible en absence de perturbations. Au sein de cette région de l'espace de phases, l'attracteur définit une forme géométrique. Il est à noter que nous pouvons trouver des attracteurs dans beaucoup de systèmes dynamiques chaotiques ou non chaotiques.

Etant donné qu'une forme géométrique caractérise un *attracteur*, autour de laquelle le système évolue à long terme, nous pouvons classifier les différents attracteurs suivants en fonction des types définis ci-dessous (cf. Fig. 3.3) :

- ***Le point fixe*** *(d'équilibre)* : cet attracteur caractérise un système atteignant un état stationnaire (exemple : balancer un pendule au bout d'une ficelle).

- ***Périodique*** *(de trajectoire fermée)* : il caractérise un système atteignant un état périodique. (exemple : en tournant un cerceau autour de la taille, il décrit une trajectoire imparfaite mais après sa trajectoire va se stabiliser et former

des cercles parfaits tant qu'on l'impulse de façon constante.)

- ***L'attracteur spatial*** : il décrit une surface. (exemple : l'attracteur torique, qui représente les mouvements résultant de deux oscillations indépendantes dont les trajectoires s'enroulent autour d'un tore).

- ***L'attracteur étrange*** : sa structure contient deux tendances apparemment antagonistes : l'attraction des trajectoires vers un type de comportement et leur divergence sur une trajectoire constante.

  Les trajectoires observées ne se coupent jamais, et pourtant semblent évoluer au hasard formant des sortes de boucles pas tout à fait concentriques, pas tout à fait sur le même plan, mais formant des figures indiscutablement reconnaissables.

  L'un des exemples classiques est l'attracteur de Lorentz ou "papillon".

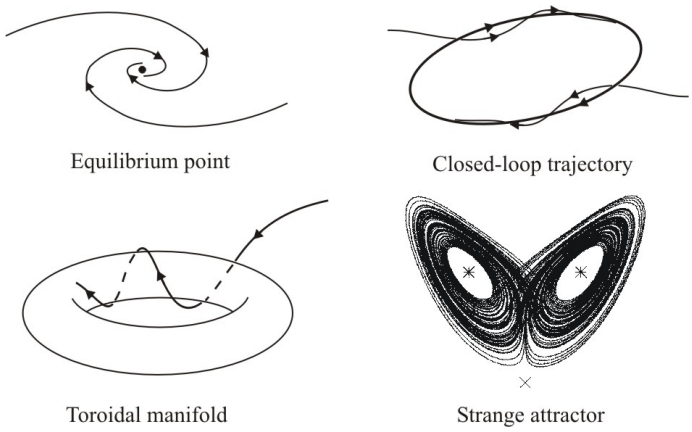

FIGURE 3.3 – Les différents types d'attracteurs

Dans le cas des systèmes en temps discret, la représentation est plutôt conformée par une suite de points. L'orbite qui représente l'évolution du système s'obtient par *itération graphique.*

3.2 Caractérisation des systèmes dynamiques discrets.

Dans le cas général, un système dynamique discret (*SSD*) est décrit par un système d'équations aux différences finies, qui génère une fonction de récurrence appliquée à chaque point du système.

Par simplicité, nous allons étudier les systèmes de premier ordre, unidimensionnels (éq. 3.3), ce modèle représente un système à boucle fermée avec une seule sortie.

Nous appelerons **orbite ou trajectoire** de f_{X_n} la représentation X_{n+1} vs X_n, du système à la suite générée par l'itération du système, à partir d'une condition initiale :

$$O_{x_0} = \{x_0, X_1, X_2, \ldots\} \tag{3.5}$$

Une manière très utile dans la visualisation de l'évolution d'un système dynamique en temps discret consiste à utiliser l'itération graphique.

L'itération graphique s'effectue en superposant deux courbes : la fonction sous étude ($y = f_{(x)}$), et la diagonale de pente unitaire ($y = x$). Nous allons utiliser la diagonale pour retrouver le point obtenu sur l'axe des y, dans l'axe des x, afin d'exécuter graphiquement une itération de f, à partir d'une condition initiale x_0 (figure 3.4).

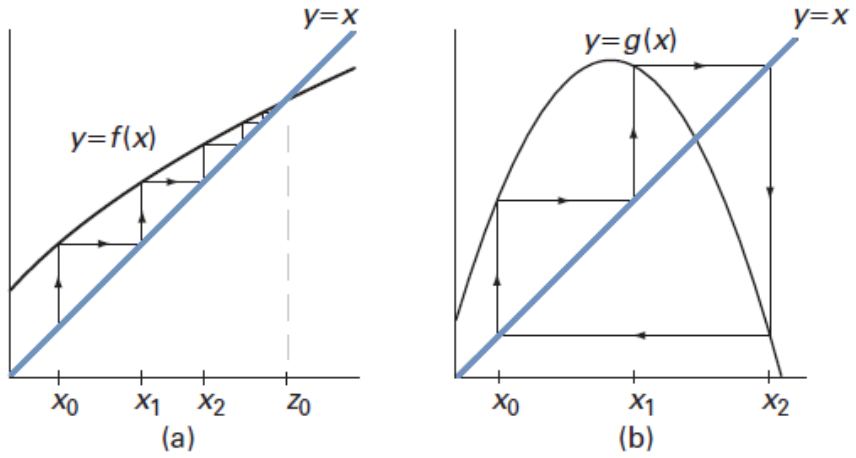

FIGURE 3.4 – Iteration graphique d'une trajectoire 1-D

La figure 3.4 illustre l'itération graphique de deux fonctions, f et g. La fonction $f_{(x)}$ (3.4.a) converge au bout des itérations vers un point fixe, z_0, tant que $g_{(x)}$ continue les itérations autour d'une trajectoire fermée de façon indéfinie (3.4.b).

En appliquant les définitions des types d'attracteurs, nous pouvons obtenir des relations utiles pour l'analyse des orbites :

– la condition du point fixe, est atteinte quand un point x_k verifie :

$$f_{(x_k)} = x_k \tag{3.6}$$

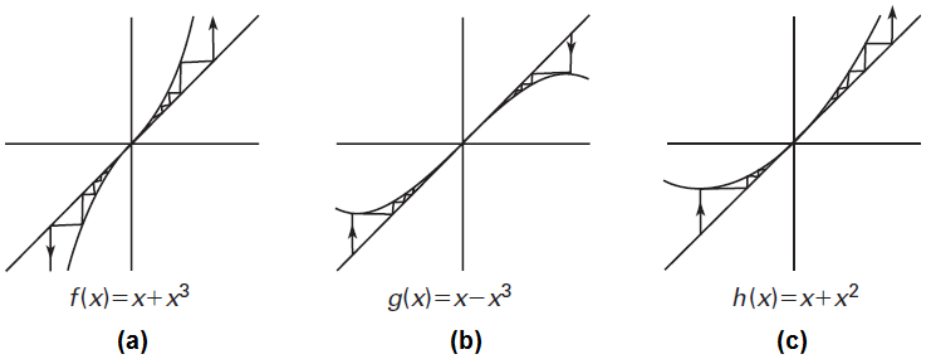

FIGURE 3.5 – Types de point fixe

Un critère d'identification de cette caractéristique consiste à analyser la première dérivée de la fonction autour du point fixe. Une illustration est proposée en Fig.3.5. Soit x_0 un point fixe de f, avec la fonction f soumise à un paramètre λ :

- Si $|f'(x_0)| > 1$, x_0 est un point répulsif, à partir de ce point il y a différents points adjacents divergents vers des valeurs différentes de ce point après un certain nombre d'itérations(3.5a).

- Si $|f'(x_0)| < 1$, x_0 est un point attracteur, les points adjacents convergent vers ce point après un nombre d'itérations (3.5b).

- Si $f'(x_0) = \pm 1$, x_0 est un point invariant où nous ne pouvons rien indiquer en particulier concernant x_0 (3.5c).

– La condition de périodicité, (cycle d'ordre p) d'une orbite se vérifie quand elle revient vers la valeur de départ au bout de p itérations :

$$X_{(n+p)} = X_{(n)}, p > 0. \tag{3.7}$$

Les points périodiques vérifient :

$$f^p_{(x)} = \underbrace{f(f(f(f(...f}_{p\ \ fois}(x))...))) = x \tag{3.8}$$

$$X_{(n+kp)} = X_{(n)}, \forall k \tag{3.9}$$

3.2.1 Le phénomène de bifurcation.

Le phénomène de bifurcation indique un changement significatif sur la dynamique d'un système, qui se produit lorsque ses paramètres changent. Donc en fonction du changement d'un paramètre, un système peut aller vers une caractéristique monotone, périodique ou chaotique. La valeur pour laquelle l'effet de bifurcation se

produit est nommée *point de bifurcation.*

Il existe une bifurcation (un changement dans le nombre de points fixes) si f_λ possède un point fixe invariant, pour une valeur de λ donné. Si, par rapport aux voisinages de λ, il présente la transition d'un point attracteur vers un répulsif, le système présente un dédoublement de la période.

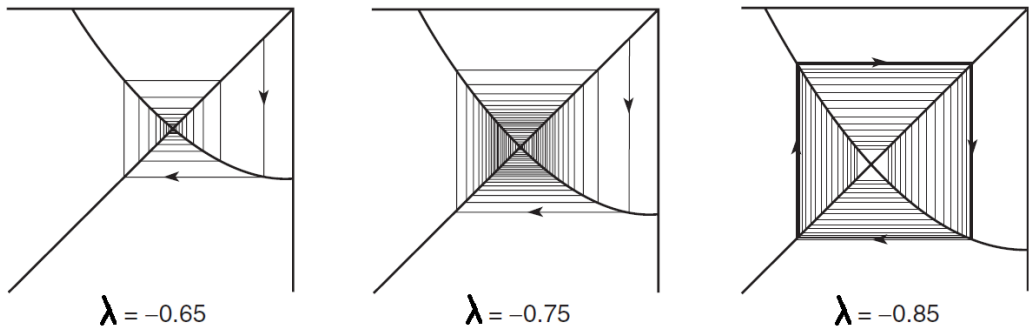

FIGURE 3.6 – Exemple de dédoublement de la période.

La figure 3.6 illustre un exemple de dédoublement de la période à l'aide des itérations graphiques sur une fonction. La fonction, ici sous étude, est :

$$f_{(x)_\lambda} = x^2 + \lambda \tag{3.10}$$

possédant un point fixe x_o sur $\frac{1}{2} \pm \frac{\sqrt{1-4\lambda}}{2}$.

En analysant $f'_{(x)_\lambda} = 2x$, autour du point $\lambda_o = -3/4$, sur une des solutions $x_o = \frac{1}{2} - \frac{\sqrt{1-4\lambda}}{2}$, nous obtenons $f'_{(x_o)_{\lambda_o}} = -1$, indiquant que pour cette valeur de λ, il existe un point invariant. En analysant les voisinages, nous trouvons que $\lambda > -3/4$ implique un comportement d'attracteur et $\lambda < -3/4$ un comportement de point répulsif. Le point $\lambda = -3/4$ correspond à la définition d'un point de bifurcation, avec dédoublement de la période dans la dynamique du système.

Graphiquement, nous pouvons constater que si l'orbite croise la diagonale de symétrie, son itération graphique effectuera un chemin concentrique qui sera ouvert ou fermé en fonction des conditions initiales.

3.3 Le chaos déterministe.

Le chaos possède un comportement non prédictible, lié à l'instabilité et à la non linéarité des systèmes dynamiques déterministes.

Ceci se manifeste comme ayant une sensibilité élevée aux changements des conditions initiales, provoquant une croissance des erreurs de prédiction entre deux événements,

que l'on appelle *chaos*. Ce phénomène introduit du hasard dans un système physique, même s'il est décrit par des équations parfaitement déterministes. L'effet d'une incertitude sur un état initial peut donc, donner lieu pendant l'évolution temporelle d'un système à des instabilités, tout en restant dans les limites de l'espace de phase.

3.3.1 Transition vers le chaos.

L'un des aspects de recherche est une définition claire et formelle des conditions dans lesquelles un système devient chaotique. Afin de simplifier l'approche, une modélisation simplifiée via un nombre réduit de paramètres, permet de réaliser une observation du passage vers le chaos.

Via cette approche et en fonction d'un paramètre du modèle, un système peut passer d'un état stationnaire vers un périodique et faire postérieurement une transition vers le chaos. L'évolution d'un point fixe vers le chaos est caractérisée par des changements discontinus appelés *bifurcations*. Les bifurcations, décrites précédemment, marquent le passage d'un régime à un autre.

Le passage vers le chaos peut suivre trois scénarios d'évolution définis ci desous.

3.3.1.1 Doublement de période.

En changeant un paramètre, la fréquence du régime, initialement périodique, est doublée consécutivement (x2, x4, x8, ...). Les doublements de fréquence deviennent de plus en plus proches, avec une tendance d'accumulation vers l'infini. C'est dans cet instant où le système devient chaotique.
Un exemple de ce comportement est représenté par la suite logistique qui sera décrite postérieurement.

3.3.1.2 Apparition d'intermittences.

Il présente des apparitions erratiques du chaos qui reviennent vers un comportement périodique du système Il se déstabilise brutalement, puis il s'éloigne de la valeur critique qui l'a amenée à un tel état, en continuant vers un comportement périodique puis en revenant successivement. Un exemple de ce comportement s'observe dans le système de Rössler.

3.3.1.3 Quasi périodicité.

Elle est illustrée par le modèle de Lorentz. Elle résulte de la concurrence de différentes fréquences dans le système, de façon à osciller entre un état dans la limite de la périodicité, qui permet de transiter vers le chaos.

3.3.2 Sensibilité aux conditions Initiales.

Cette caractéristique permet de reconnaître un système chaotique, celle ci étant une caractéristique intrinsèque de ces derniers.

Par exemple, deux feuilles mortes tombant d'un même point de départ, possèdent des trajectoires qui vont se séparer pour devenir totalement dissemblables. Cette caractéristique, la *sensibilité aux conditions initiales*, permet d'identifier un système chaotique, régit par un attracteur étrange.

La divergence exponentielle des deux trajectoires reste cependant un phénomène local. Au même titre que pour les attracteurs qui disposent de dimensions finies, il est alors impossible d'avoir les deux trajectoires divergentes de manière infinie, de sorte que l'attracteur doit se replier sur lui-même à un moment ou à un autre.

Plusieurs types d'attracteurs étranges ont été trouvés dans de nombreux systèmes complexes, comme dans l'écoulement des fluides, dans la propagation du laser ou dans les battements cardiaque, etc. Il faut considérer le nombre de dimensions (variables) utilisées sur l'analyse du système dynamique. En effet, vu que nous sommes limités par les trois dimensions de l'espace euclidien pour une représentation visuelle des attracteurs, il faut choisir les variables pertinentes parmi l'ensemble multidimensionnel de variables dans notre modèle.

3.3.3 Les exposants de Lyapunov.

L'évolution d'un système chaotique est difficile à quantifier, vu la divergence entre les trajectoires sur un attracteur. Une méthode utilisée pour mesurer la vitesse de la divergence ou convergence dans un système, consiste au calcul des exposants de Lyapunov.

Il nous permet d'avoir une mesure du degré de stabilité du système. Le nombre d'exposants de Lyapunov est égal à la dimension de l'espace de phases, et indexes en fonction de leur valeur. Il existe deux façons d'obtenir de tels exposants, soit directement de l'expression formelle du système, soit en traitant des échantillons obtenus sur la réponse temporelle du système.

Ainsi, dans le but d'illustrer la démarche présentée, nous allons appliquer l'analyse sur une itération à une dimension :

$$X_{(n+1)} = f(X_n) \tag{3.11}$$

Cette fonction non linéaire génère un attracteur chaotique, de condition initiale x_0. Nous allons voir l'évolution des itérations devant une perturbation initiale δx_0 :

$$X_{(1)} + \delta x_1 = f(x_0) + \delta x_0 = f(x_0) + f'(x_0)\delta x_0 \tag{3.12}$$

La déviation est donnée lors de la première itération par :

$$\delta x_1 = f'(x_0)\delta x_0 \tag{3.13}$$

Au bout de la déuxième itération, la déviation est :

$$\delta x_2 = f'(x_1)f'(x_0)\delta x_1 \tag{3.14}$$

Qui forme une règle de récurrence :

$$\delta x_n = (\prod_{m=0}^{n-1} f'(x_m))\delta x_0 \tag{3.15}$$

Une approximation propose :

$$\delta x_n \simeq (\gamma)^n |\delta x_0| \tag{3.16}$$

étant γ un indicateur de l'évolution à chaque itération.

$$(\gamma)^n = (\prod_{m=0}^{n-1} f'(x_m)) \tag{3.17}$$

En appliquant un logarithme à l'équation (3.17), nous obtenons l'expression suivante :

$$\lambda = log(\gamma) = \lim_{n\to\infty} \frac{1}{n} \sum_{i=0}^{n-1} log(|f'(x_i)|) \tag{3.18}$$

Nous appelons l'élément λ exposant de Lyapunov.
Un exposant de Lyapunov positif indique que selon la direction qu'il représente la divergence entre deux trajectoires voisines augmente exponentiellement avec le temps.
Il s'agit donc bien là d'une caractérisation d'un attracteur étrange. Pour une application bidimensionnelle, la présence d'au moins un exposant de Lyapunov nous permet d'avoir une idée de l'évolution du système.

La valeur de l'exposant de Lyapunov (LLE, largest Lyapunov Exponent) indique le degré de chaos du système :

- $LLE = 0$, caractérise une orbite marginalement stable,
- $LLE < 0$, caractérise une orbite périodique ou un point fixe,
- $LLE > 0$, caractérise une orbite chaotique.

Une technique de calcul des exposants de Lyapunov qui n'est pas affectée par le nombre d'échantillons analysés, est la méthode de Rosenstein, énoncée en détail sur [38].

3.4 Exemples de systèmes chaotiques.

Nous allons citer quelques exemples de systèmes chaotiques en temps continu et en temps discrèt.

3.4.1 Modèle de Lorenz.

Le modèle de Lorenz a joué un important rôle historique puisque son évolution temporelle fait apparaître un comportement chaotique. De plus, il constitua le premier et le célèbre système différentiel dissipatif permettant d'observer un attracteur étrange pour certaines valeurs des paramètres.

Dans sa version exprimée en paramètres et variables réduits, le système de trois équations différentielles couplées s'écrit :

$$\begin{aligned} \dot{x} &= \sigma(y - x) \\ \dot{y} &= rx - y - xz \\ \dot{z} &= xy - bz) \end{aligned} \tag{3.19}$$

Les variables qui participent au modèle sont :

- les variations de température horizontale et verticale (y, z),
- le taux d'échauffement par convection (x)

accompagné des paramètres (tous les trois positifs) :

- Le nombre de Prandtl (σ),
- le nombre de Rayleigh (r)
- la taille physique du système (b)

Les valeurs des paramètres utilisées par Lorenz étaient : σ=10,b=8/3 et r=28 comme illustré sur la Fig.3.7.

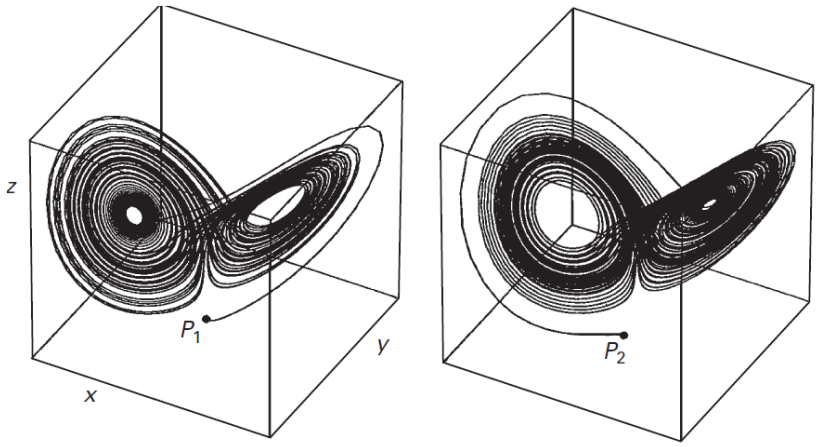

FIGURE 3.7 – Représentation 3D de l'attracteur de Lorenz avec deux conditions initiales différentes.

3.4.2 Modèle de Henon.

Ce modèle provient d'une simplification du modèle de Lorenz sur deux dimensions. Il s'exprime de la façon suivante :

$$\begin{aligned} x &= y + 1 - ax^2 \\ y &= bx \end{aligned} \tag{3.20}$$

avec comme paramètres, a qui contrôle la non linéarité et b la dissipation à chaque itération. Les valeurs habituellement utilisées pour mettre en évidence le phénomène sont : a=1.4 et b=0.3, comme illustré en Fig.3.8.

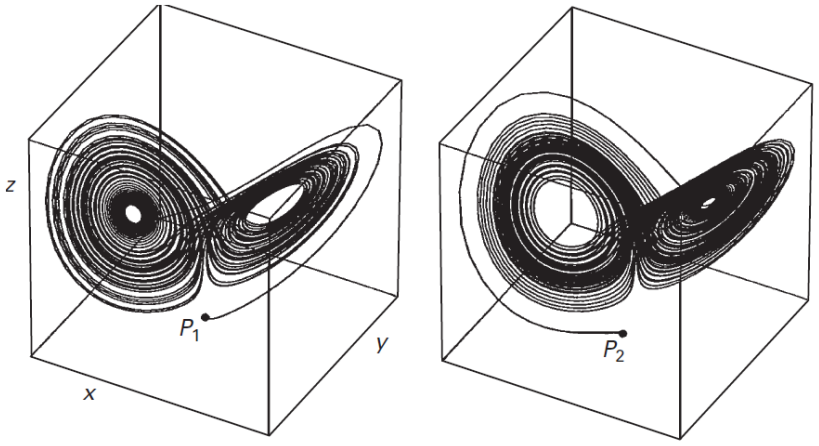

FIGURE 3.8 – Attracteur d'Hénon : a : diagramme de bifurcation b :atracteur.

L'aspect bidimensionnel du système nous permet de représenter plus facilement la dynamique du système versus l' attracteur ainsi que d'illustrer l'évolution des bifurcations en fonction du paramètre de contrôle en utilisant un diagramme de bifurcations (figure 3.8).

3.4.3 Modèle logistique

Le modèle logistique est un modèle en temps discret qui permet de modéliser l'évolution d'une population.

$$X_{(n+1)} = \lambda X_n(1 - X_n) \tag{3.21}$$

X_n indique la population en fin d'année "n" et λ ($\lambda > 0$) est un paramètre de croissance. En fonction de λ et de x_0, le système peut varier entre un point fixe, un système périodique et un système chaotique.

Une étude de ce système nous donne un point fixe ayant deux solutions :

1. pour $x_0 = 0$ et pour toutes les valeurs de λ

2. pour $x_0 = (1 - \frac{1}{\lambda})$. En appliquant la première dérivée sur le point fixe :

$$f'(x_0) = \lambda(1 - 2x_0) \tag{3.22}$$

en remplaçant dans la deuxième solution x_0 :

$$f'(x_0) = 2 - \lambda \tag{3.23}$$

Cette équation caractérise en fonction de la valeur de λ le système, à savoir :

- Pour $\lambda \in < 1; 3 >$, le système se comporte comme un attracteur.
- Pour $\lambda = 3$, il existe une bifurcation.
- Pour $\lambda < 3; 3.4 >$, le système se comporte comme un rejecteur.

Pour $\lambda = 4$, il existe un point critique car $f_4(1/2) = 1 \rightarrow f'_4(1/2) = 0$. Par rapport à la deuxième itération, $f_4^2(1/2) = f(1) = 1$ et c'est récursif pour toutes les autres n itérations.

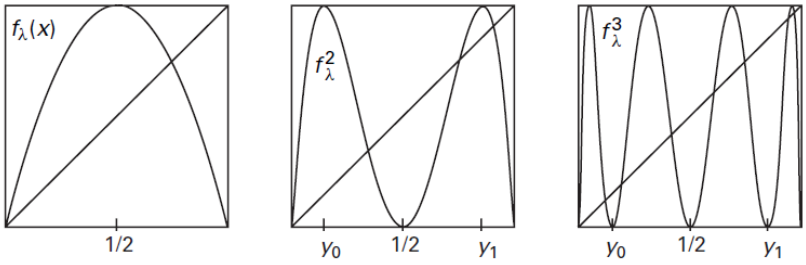

FIGURE 3.9 – Analyse des itérations de la fonction logistique.

Nous vérifions en traçant l'itération graphique de $f_4(x)$ qu'il existe, pour deux différentes valeurs initiales, des trajectoires différentes. Ceci nous permet de vérifier le chaos sous l'apparition d'un doublement de la période.

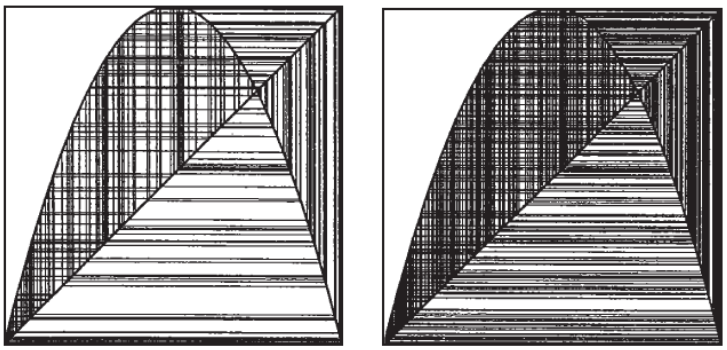

FIGURE 3.10 – Analyse des itérations de la fonction logistique.

3.4.4 Modèle de doublement.

Soit la fonction discontinue $D : [0, 1 >$ définie par :

$$\begin{aligned} D(x) &= 2x, & 0 \leq x < 1/2 \\ D(x) &= 2x - 1, & 1/2 \leq x < 1 \end{aligned} \tag{3.24}$$

La représentation graphique de D^n possède 2^n droites de pente 2^n au long de l'intervalle $[0, 1]$ en générant des sous-intervalles $[k/2^n, (k+1)/2^n]$, pour $k = 0, 1, \ldots 2^n - 2$. Ceci indique un doublement de la période à chaque itération comme le montre la Fig.3.11.

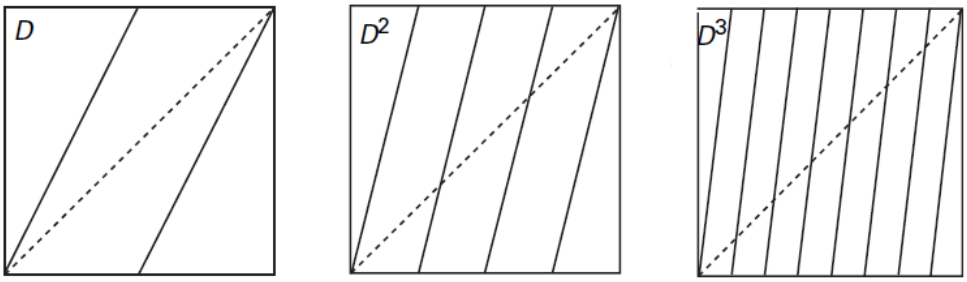

FIGURE 3.11 – Analyse des itérations de la fonction de doublement.

3.4.5 Modèle type "Tente".

Soit la fonction discontinue $D : [0, 1 >$ définie par :

$$\begin{aligned} T(x) &= 2x, & 0 \leq x < 1/2 \\ T(x) &= -2x + 2, & 1/2 \leq x < 1 \end{aligned} \tag{3.25}$$

La représentation graphique de T^n suit le même comportement observé sur D^n, avec un doublement de la période à chaque itération comme illustré en Fig.3.12.

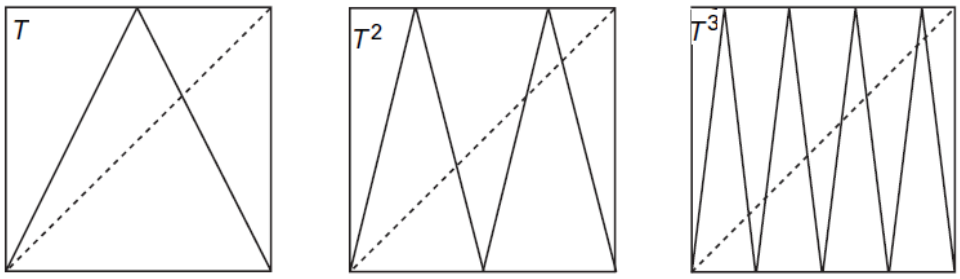

FIGURE 3.12 – Analyse des itérations de la fonction tente.

Nous pouvons constater encore une fois que pour toutes les fonctions disposant de régions de discontinuité qui croissent les deux moitiés de la diagonale de symétrie entre les deux axes, ce système peut amener vers une situation de dédoublement de la période.

3.5 Conclusion.

Nous allons alors essayer de nous approcher d'une fonction "tente" ou du moins à un modèle logistique, en utilisant des transistors pour générer une région d'au moins deux régions de discontinuités, centrées autour de la moitié de la dynamique du signal, de façon à avoir des points d'aller-retour sur les itérations graphiques de l'orbite généré, et pouvoir calibrer le système pour basculer entre une orbite périodique, ponctuelle et finalement chaotique.

Les modèles antérieurement définis sont de type mathématique, qui ne prennent en compte dans leur structure itérative une implémentation physique (notamment la ré-alimentation et l'itération sont considérées automatiques).

L'objectif sera donc dans la suite l'étude des systèmes chaotiques en temps discret, vis à vis d'une modélisation sous la forme d'un système discret, réalimenté, et conçu autour d'une fonction non linéaire avec les discontinuités décrites qui garantissent une orbite fermée, tenant en compte l'impact de l'implémentation choisie sur la dynamique du système total.

Chapitre 4

Mise en oeuvre du système chaotique en temps discret

Introduction

Après une introduction des phénomènes chaotiques, nous allons maintenant nous attarder sur l'implantation physique d'un oscillateur chaotique en temps discret. Cet oscillateur chaotique sera le point central du générateur aléatoire proposé au sein de cette étude.

La première étape de l'analyse consistera à modéliser un système chaotique en temps discret autour d'un élément non linéaire. Le modèle récursif vu dans le chapitre 3 du système assume implicitement une structure récursive, en boucle fermée.

Cette structure théorique doit se matérialiser vers un diagramme de blocs fonctionnels. Nous allons intégrer sur le modèle équivalent du système une fonction non linéaire plus le chemin de signal nécessaire pour réaliser les itérations, pilotées par un signal d'horloge qui cadencera les retours de boucle.

Après avoir défini le modèle en blocs du système, chacun des blocs sera modélisé puis matérialisé à base de transistors, tenant en compte une phase de co-simulation accompagnant l'élément non linéaire afin de constater les effets ajoutés dus à la dynamique de fonctionnement des transistors participants à chaque élément du système réalimenté. Le système sera accompagné, bien évidemment, par des circuits accessoires de polarisation et cadencement qui seront implantés autour du système chaotique.

Finalement, l'ensemble du système sera analysé en comparant le modèle compact prévu et les effets induits par l'utilisation des transistors.

4.1 Modèle discret équivalent réalimenté.

Comme nous l'avons vu précédemment, le modèle d'un oscillateur chaotique à temps discret correspond à une fonction non linéaire qui s'exécute récursivement à des intervalles réguliers.

Un système récursif de premier ordre possède, théoriquement, la forme :

$$X_{n+1} = F_{(X_n)} \tag{4.1}$$

dont X_n représente l'état du système dans le n^{esime} instant observé $n = 0, 1, 2, \ldots$.

Dans le but d'une réalisation matérielle non algorithmique, la récursivité de cette fonction itérative peut être réalisée via un système en boucle fermé (figure 4.1(a)), possédant une étape qui contrôle le retour de la boucle, cadencé par le signal d'horloge CK et qui gère le chemin du signal à travers des interrupteurs complémentaires.

Afin d'éviter des possibles pertes sur le chemin de retour, ainsi que de dissocier la valeur à itérer, nous allons inclure un élément intermédiaire sur le retour de la boucle. Nous pouvons utiliser un élément non inverseur, de gain unitaire, sur la chaîne de retour (figure 4.1(b)). Afin de mieux visualiser l'évolution de X_k en nous basant de la solution précédente (figure 4.1(b)), nous avons redessiné le montage, illustré sur la figure 4.1(c).

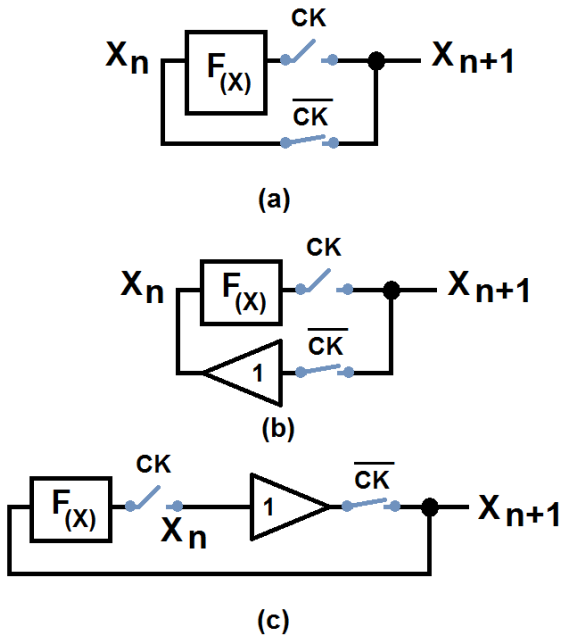

FIGURE 4.1 – Générateur chaotique en temps discret :
(a) Schéma général (b) Équivalent avec étage non inverseur (c) Équivalent modifié.

Nous allons nous servir de la dernière solution qui possède l'avantage de disposer d'un unique bloc contenant la fonction non linéaire et d'une étape de gain unitaire qui n'altère pas significativement le signal, mais qui sert de passerelle, adaptation dans le chemin du signal commuté, à effets d'utiliser des capacitances parasites comme éléments de mémoire (figure 4.2b). La présence des éléments de mémoire permet au système d'effectuer les itérations dans les instants où le chemin du circuit

commuté ne communique pas une étape avec la suivante, permettant l'évaluation analogique de la fonction par étapes.

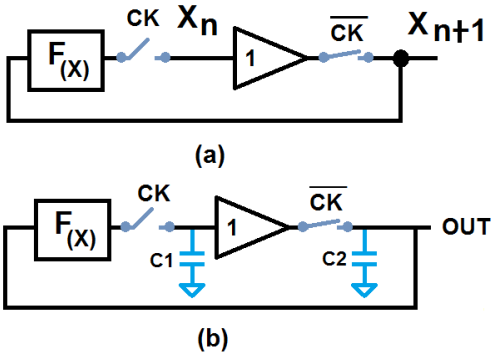

FIGURE 4.2 – (a) Modèle de l'oscillateur chaotique (b) avec représentation des éléments de mémoire capacitif.

4.2 Caractérisation des éléments.

Nous allons représenter la topologie retenue (figure 4.2) vers un système avec cinq blocs élémentaires, représentés dans le diagramme de la figure 4.3 :

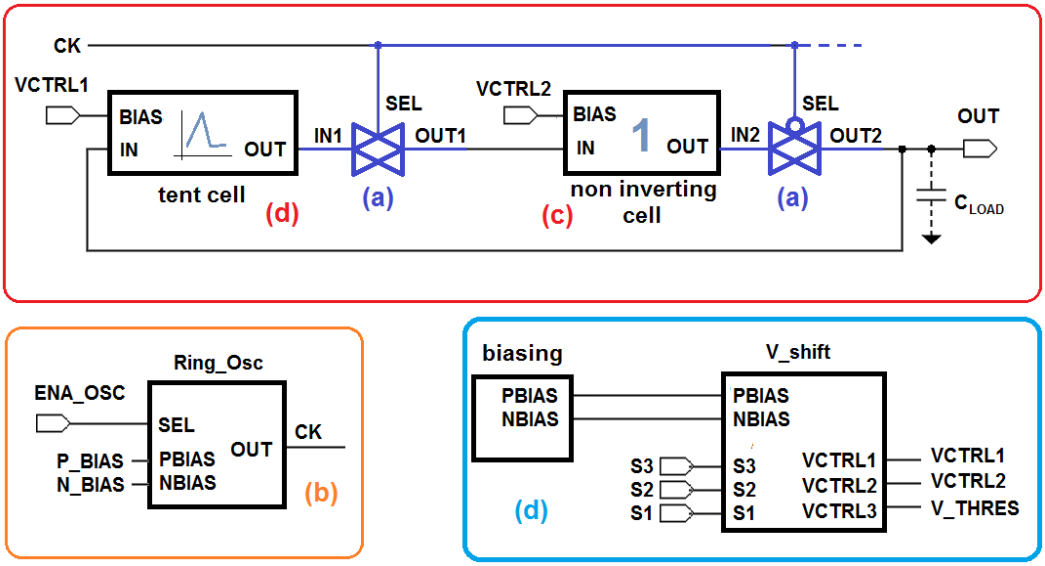

FIGURE 4.3 – Représentation en blocs de l'oscillateur chaotique.

(a) Un système d'interrupteurs.

(b) Un générateur d'horloge de synchronisation des interrupteurs.

(c) Un étage de gain unitaire.

(d) Une fonction non linéaire.

(e) Un circuit de polarisation.

4.2.1 Interrupteurs complémentaires.

Le but des interrupteurs est de retenir la valeur du signal instantanée lors des phases d'ouverture / fermeture des interrupteurs sur le chemin de la boucle. Le modèle du système assume un effet mémoire à l'entrée de chaque étape (figure 4.2(a)).

Ainsi, des éléments capacitifs peuvent modéliser cet effet mémoire entre les entrées et sorties de chaque bloc comme illustré en figure 4.2(b). Cet élément capacitif modélisé prend en compte l'effet capacitif en entrée de chaque porte (usuellement la capacité grille/source de chaque inverseur) mais également les effets capacitifs parasites de ligne. Enfin, la nature complémentaire du système permet de compléter le retour de la boucle après un cycle d'horloge.

4.2.2 Générateur d'horloge.

Comme son nom l'indique, ce bloc fournit un ou plusieurs signaux carrés en sortie. Il a pour fonction de cadencer le système d'interrupteurs dans des intervalles constants sur le système, ainsi que pour d'autres circuits synchrones si nécessaire. Pour une utilisation dans l'oscillateur chaotique, nous avons besoin de deux sorties complémentaires, CK et $\overline{CK}$.

4.2.3 Étage de gain unitaire.

L'étage de gain, idéalement unitaire sert de stockage temporaire de l'information. Cette dernière est fournie par la fonction non linéaire de la boucle de l'oscillateur chaotique. D'un point de vue théorique, cette étape ne doit apporter aucune dégradation ni déphasage au signal, toutefois, ceci n'est pas toujours le cas lors du passage à l'aspect pratique.

Ce bloc est susceptible de générer des tensions d'offset par rapport au signal d'entrée et un gain inférieur mais proche de l'unité, qui peut ajouter des effets complémentaires sur le comportement du signal sortant de la fonction non linéaire. Ces effets et autres non linéarités possibles seront analysés postérieurement dans la modélisation du circuit, vis-à-vis de sa réalisation à base de transistors.

4.2.4 Élément non linéaire

Une cellule non linéaire permet d'altérer la dynamique d'un système à boucle fermée vers le chaos.

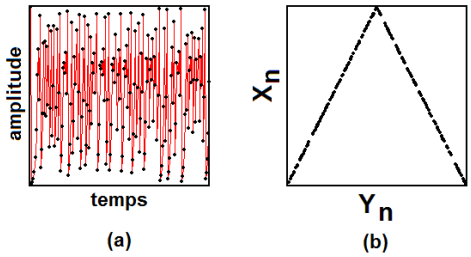

FIGURE 4.4 – "*tent map*" :(a) réponse transitoire (b) itération graphique.

Idéalement, la fonction rampe est définie par deux fonctions linéaires chacune couvrant une partie du rang de valeurs de la dynamique, de pente égale à 2(-2). Mais ce n'est pas limité à pente. Tant que les deux fonctions intervennant dans l'étude peuvent croiser l'axe symétrique $x = y$ dans l'évolution de la fonction, nous pouvons attendre un comportement qui pourra converger vers une orbite fermée.

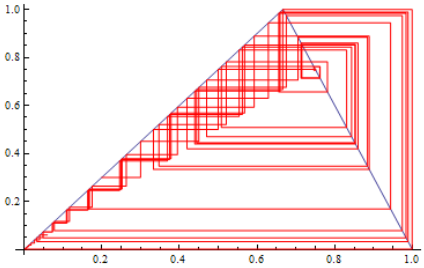

FIGURE 4.5 – Iteration graphique sur une fonction du type "tente".

Dans le cas d'une réalisation plus réaliste, nous allons considérer des limitations dans la dynamique, telles que l'asymétrie de la fonction caractéristique, ainsi que de possibles cas de saturation et d'offset pouvant apparaitre sur le chemin de retour du signal. La figure 4.5 montre l'itération graphique de la fonction, présentant une évolution vers un comportement chaotique.

Nous allons essayer de nous approcher de ce type de gabarit et de réponse avec des montages à base de transistors MOS.

Après avoir brièvement introduit l'intérêt des différents blocs, nous allons maintenant nous focaliser sur la mise en oeuvre de chaque élément décrit du système chaotique. Afin de rendre la lecture plus abordable, une présentation bloc sera proposée via une explication des différentes fonctionnalités et de leurs intérêts.

Ensuite, une description au niveau transistor sera introduite allant jusqu'au dimensionnement retenu ainsi que des conditions de fonctionnement. Ainsi, tout lecteur souhaitant aborder le sujet ou approfondir ses connaissances dans les générateurs de valeurs aléatoires pourra se servir de ce manuscrit comme d'une base de travail.

4.3 Description des blocs du système chaotique.

4.3.1 Système d'interrupteurs complémentaires.

Ces interrupteurs permettent de retenir à la sortie, à chaque demi période d'horloge, l'état du bloc précédant dans le circuit réalimenté. Cela permet d'établir une évolution discrète du système en terme du *sampling and hold*, étant implicites l'effet mémoire des capacités d'entrée-sortie de chaque bloc dans la boucle.

4.3.1.1 Description détaillée du circuit.

Ce circuit utilise deux transistors NMOS (**MN2**, **MN3**) comme interrupteurs de signal, commandés par le signal **SEL**. Étant donné le positionnement des interrupteurs, ceux ci devant fonctionner en opposition de phase, une action complémentaire a été réalisée en utilisant un inverseur logique (**MP1**, **MN2**) qui inverse le signal **SEL**. Ainsi, le second interrupteur reçoit un signal en opposition de phase vis-à-vis du premier.

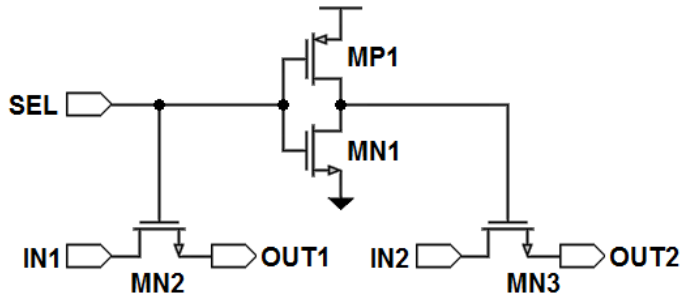

FIGURE 4.6 – Schématique des interrupteurs complémentaires.

4.3.1.2 Dimensionnement des composants.

Les tailles des transistors prévus pour le montage présenté en Fig.4.6 sont proposés dans le tableau ci dessous.

Mode	Not Gate		Switches	
Transistors	MP1	MN1	MN2	MN3
Dimensions [μm]	$\frac{2\times0.4}{5}$	$\frac{0.4}{5}$	$\frac{1.5}{0.5}$	$\frac{1.5}{0.5}$

TABLE 4.1 – Dimensionnement des interrupteurs complémentaires.

L'utilisation de deux transistors MOS jouant le rôle d'interrupteur, introduit de légères pertes aux bornes des interrupteurs. Celles-ci sont liées à la tension drain-source de saturation des transistors MOS. Cette perte est de l'ordre de 100 à 200mV sur une technologie CMOS 0.35μm. Toutefois, cette perte est réduite via l'utilisation d'une structure bouclée sur elle-même comme proposée précédemment.

4.3.2 Générateur d'horloge.

Le générateur d'horloge permet de fournir un signal carré de période défini ayant pour finalité le cadencement des interrupteurs précédemment introduits. La figure 4.7 représente l'architecture simplifiée de l'oscillateur retenu.

4.3.2.1 Architectures de générateur d'horloge.

L'oscillateur est basé sur une topologie dérivée des oscillateurs en anneau. Comme nous l'avons vu précédemment, cette structure repose sur l'utilisation d'un enjeu d'inverseurs en nombre impairs rebouclés sur eux mêmes.

De plus, la topologie retenue correspond au type "*current starving*", centrée sur l'utilisation d'une source de courant alimentant chacun des inverseurs, permettant le contrôle du courant consommé par le circuit en anneau. Cette caractéristique permet d'imposer la consommation du circuit, ceci étant très judicieux pour des applications basses consommations. Pour y parvenir, la topologie dérivée possède une source de courant commune pour les inverseurs logiques (figure 4.7).

De plus, la topologie possède une entrée qui permet la mise en mode *stand-by* du circuit. Ainsi, via l'utilisation d'un signal de contrôle **ENA**, pilotant un inverseur de remise en forme du signal (**U6**), le signal est ensuite utilisé pour piloter l'interrupteur (**SW**), placé de sorte à couper l'alimentation en courant des inverseurs. Ainsi, le circuit possède une entrée qui permet d'activer-désactiver le circuit.

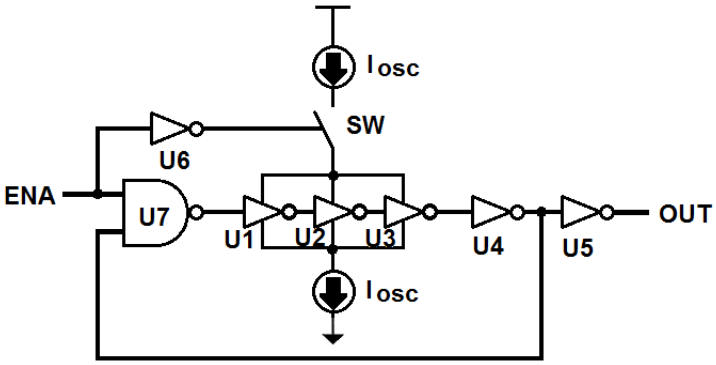

FIGURE 4.7 – Diagramme simplifié du générateur d'horloge.

L'oscillateur de base, "*shared current bias*" est réalisé à l'aide de trois étages d'inverseurs (**U1**,**U2**,**U3**) polarisés par une source de courant unique par branche, I_{OSC} (figure 4.7). Cette configuration possède l'avantage d'utiliser un courant de polarisation réduit qui par son unicité permet de mieux masquer les transitions des portes logiques. D'un autre côté, il délivre un signal de sortie de faible amplitude et basse capacité de charge (figure 4.8).

Le choix de ce type d'oscillateur répond au fait que, comparativement aux autres solutions existantes de type "*current starved*", ce montage présente une faible déviation de fréquence en fonction des variations de température et de tension d'alimentation. Si un comparatif est réalisé avec les autres solutions usuellement employées, telles que :

- *current starved with output switching*
- *current starved with power switching*
- *current starved with current shared bias*

Nous pouvons constater un comportement semblable en fonction de la température et de la tension d'alimentation. Toutefois, les autres variantes, reposant sur des circuits *current starved* ayant des sources de courants indépendantes pour chaque inverseur (figure 4.9), présentent l'inconvénient d'une plus faible immunité aux variations.

Le tableau 4.2 présente une synthèse des différentes performances des différentes architectures précédemment introduites mettant en avant par rapport à nos contraintes, la solution de l'oscillateur de type *current starved-shared source*, en utilisant la méthodologie proposée par [44].

Type "Current starved with	"Output Switching"	"Power Switching"	"Shared Current Bias"
Température	0.4 % /°C	0.37 % /°C	0.267 %/°C
Alimentation	-42.5 % /V	-11.25 % /V	13.3 % /V

TABLE 4.2 – Dérivation en fréquence de chaque type d'oscillateur.

Enfin, une porte *NAND* a été placée en entrée (**U7**) de la structure afin de mettre en *stand-by* la fonction oscillateur (en parallèle avec la mise à l'état bas du niveau de courant d'alimentation de l'oscillateur). Un inverseur (**U4**) a été également positionné dans la boucle de l'oscillateur afin de compléter le nombre impair de cellules inverseuses (1 NAND + 4 inverseurs).

Finalement, un inverseur (**U5**) est placé en sortie de la structure afin de récupérer un signal numérique augmentant par la même occasion la capacité maximale de sortie du générateur. Il est à noter également l'intérêt de l'ajout de portes logiques standards afin de compenser le *jitter* de l'oscillateur et la déviation globale de fréquence.

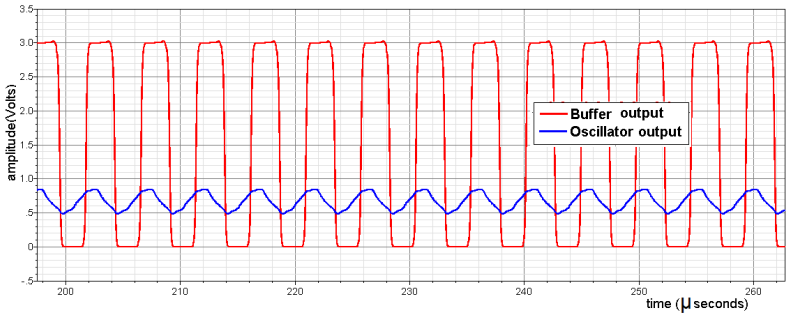

FIGURE 4.8 – Oscillateur csi (sans modification).

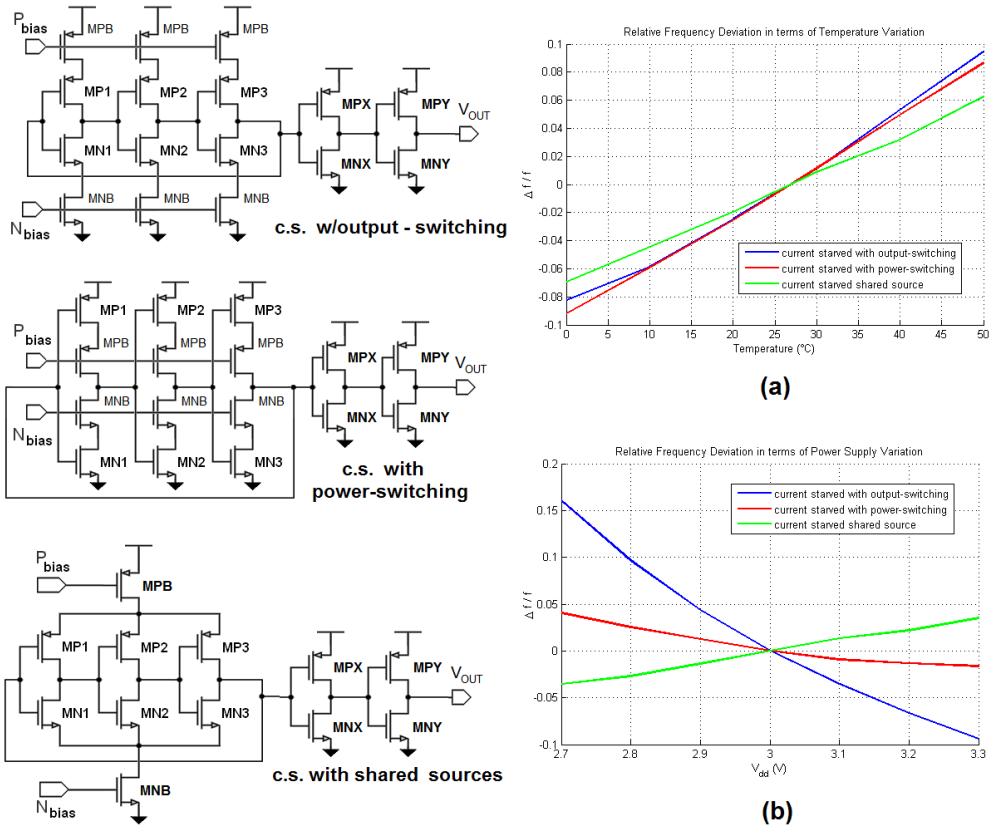

FIGURE 4.9 – Variantes de la topologie *current starved* et déviation relative de fréquences en fonction de : (a) la température (b) la tension d'alimentation.

4.3.2.2 Description détaillée du circuit.

La figure 4.10 illustre l'implémentation de l'oscillateur à base de transistors MOS. Si l'on reprend les blocs précédemment introduits constituant le générateur d'horloge, nous trouvons donc :

– Les sources de courant, I_{OSC} dans la rame supérieure et inférieure sont respectivement représentées par les deux transistors (**MP9**) et (**MN9**). Les deux transistors sont polarisés par deux tensions **PBIAS** (pour le transistor PMOS) et (**NBIAS** (pour le transistor NMOS). Le circuit de polarisation assurant ces deux tensions sera explicité ultérieurement. Toutefois, les deux transistors polarisés par une tension fixe jouent le rôle de miroir de courant simple permettant ainsi une décorrélation entre la fonction polarisation (**PBIAS** et **NBIAS**) et la fonction alimentation I_{OSC}.

– L'interrupteur **SW** est remplacé par son équivalent à base de transistor, soit le transistor **MP10**. Celui-ci était piloté via le signal **ENA** inversé via la structure inverseuse (**MP8,MN8**). Cet interrupteur permet la mise en veille de la structure oscillateur via la mise a l'état bas du signal **ENA**.

– L'oscillateur en anneau, comme introduit précédemment, est réalisé via trois inverseurs identiques **U1,U2,U3** réalisés avec les paires de transistors PMOS et NMOS (**MP1-3,MN1-3**). La porte NAND **U7** en entrée est réalisée par les transistors (**MP5-6,MN5-6**) et l'inverseur **U4** ajouté dans la boucle de retour afin d'assurer un nombre impair, illustrée par les transistors (**MP4,MN4**).

– L'inverseur de sortie, **U5**, implémenté via les transistors (**MP7,MN7**), assure la remise en forme du signal de sortie.

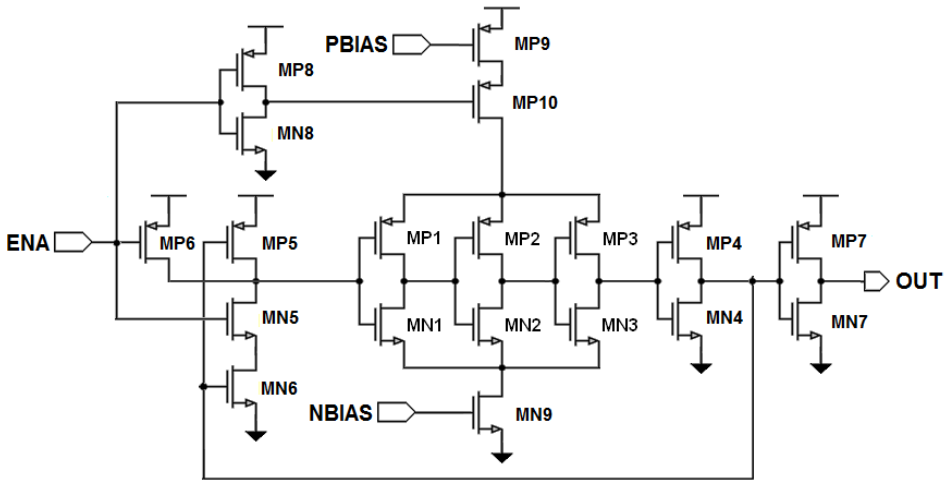

FIGURE 4.10 – Schématique du générateur d'horloge.

4.3.2.3 Dimensionnement des composants.

Afin de rendre ce manuscrit de thèse exploitable dans le futur, les tailles des transistors ont été données afin de pouvoir permettre aux futurs concepteurs de générateurs de sources chaotiques de disposer d'une base de travail. Ainsi, les tailles

des transistors pour le bloc de générateur d'horloge sont fournies dans le tableau ci dessous.

Transistors	Dimensions [μm]	Transistors	Dimensions [μm]
MP(1 :3)	$\frac{3\times0.5}{5}$	MN(1 :3)	$\frac{2\times0.5}{5}$
MP9	$\frac{4\times1}{7.5}$	MN9	$\frac{4\times0.5}{5}$
MP10	$\frac{2}{0.35}$	MN6	$\frac{4\times0.4}{25}$
MP4	$\frac{2.5}{15}$	MN4	$\frac{0.4}{15}$
MP(5 :6)	$\frac{0.4}{10}$	MN5	$\frac{0.4}{25}$
MP7	$\frac{2\times1}{2}$	MN7	$\frac{1}{2}$
MP8	$\frac{2.5}{15}$	MN8	$\frac{1}{15}$

TABLE 4.3 – Dimensionnement du générateur d'horloge.

La Fig.4.11 montre l'implémentation retenue.

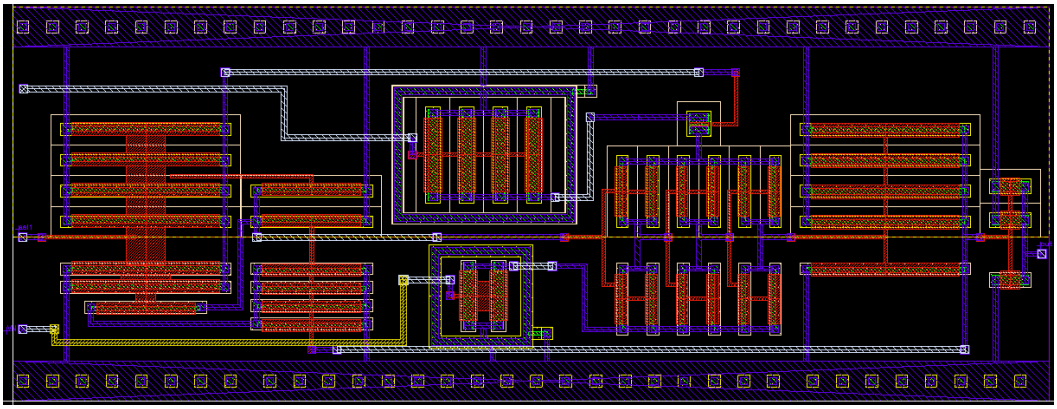

FIGURE 4.11 – Layout correspondant au bloc oscillateur.

4.3.2.4 Caractéristiques obtenues

La figure 4.12 illustre le signal de retour de la boucle ainsi que le signal de la sortie de l'oscillateur en anneau conçu. La figure 4.8 montre l'effet regénératif de l'ajout des portes standards dans le chemin de l'oscillateur de base sur la dynamique interne de l'oscillateur.

Le générateur d'horloge fonctionne sous les contraintes suivantes :

- Alimentation du circuit V_{DD}=3.3V
- Signal carré en sortie variant entre 0-3.3V
- Fréquence de sortie de 250 kHz.
- Courant de polarisation I_{OSC}=250 nA.

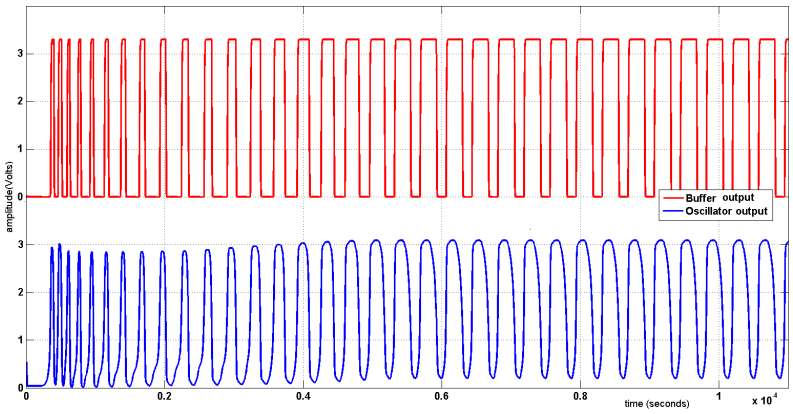

FIGURE 4.12 – Formes d'ondes générées par l'oscillateur.

4.3.3 Étage de gain unitaire.

Un étage de gain unitaire est utilisé comme passerelle du signal dans le chemin de la boucle. Il doit donc assurer un gain unitaire en tension en restant sur une consommation faible.

4.3.3.1 Description détaillée du circuit

Un montage usuellement employé pour réaliser cette fonction est le *NMOS Source Follower (NSF)* (ou amplificateur drain commun) comme le montre la figure 4.13. Ce bloc suiveur est formé à partir de deux transistors **NMOS**.

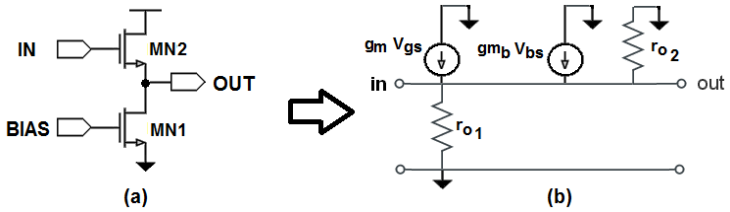

FIGURE 4.13 – NSF (a) schématique (b) schéma petit signal.

Dans ce montage, le transistor **MN2** apporte l'élément de gain par transconductance ; quant au transistor **MN1**, il opère comme source de courant constant. L'utilisation d'une source de courant constant se réalise par l'application d'une tension de polarisation **BIAS** à la grille du transistor **MN1**, permettant d'imposer la consommation du circuit.

Dès la mise en fonctionnement du circuit, pour un signal d'entrée **IN** dépassant la tension de seuil de **MN2**, la tension de sortie s'exprime sous la forme suivante :

$$V_{out} = V_{in} - V_{gs_2} \tag{4.2}$$

Ainsi, en considérant une opération en saturation :

$$V_{gs_2} = V_{Tn} - \sqrt{\frac{I_{BIAS}}{K_n W/L}} \tag{4.3}$$

Sur AMS 0.35μm, $K_n \simeq 115\mu A/V^2$. Dans notre cas, nous allons polariser ce circuit avec une source de courant de 50nA. Le deuxième terme de l'équation 4.3 peut être négligé et nous avons finalement :

$$V_{out} \simeq V_{in} - V_{tn_2} \tag{4.4}$$

Nous retrouvons bien un fonctionnement linéaire entre l'entrée et la sortie avec un déclenchement à partir de la tension V_{tn_2}. La fonction suiveuse est donc bien garantie en DC. Concernant le gain, qui va affecter aussi la dynamique, nous devons analyser le circuit de la figure 4.13b, correspondant au schéma petit signal du circuit proposé.

Les substrats de tous les NMOS étant tous reliés à la masse, la tension substrat-source (*Bulk-Source* V_{bs}) du transistor **MN1** sera nulle alors que pour le transistor **MN2**, elle sera définie par la relation : $V_{bs} = -V_{out}$.

En mode petit signal, l'introduction d'une tension substrat-source non nulle, entraîne la prise en compte de la transconductance gm_b du transistor **MN2**. La fonction de transfert du circuit prend alors la forme suivante :

$$\frac{V_{out}}{V_{in}} = \frac{gm_2 r_{o_2}}{1 + (gm_2 + gm_b) r_{o_2} + \frac{r_{o_2}}{r_{o_1}}} \tag{4.5}$$

$$\simeq \frac{gm_2 r_{o_2}}{1 + (gm_2 + gm_b) r_{o_2}} \tag{4.6}$$

expression qui se simplifie en considérant r_{o_1} très élevé. Également avec r_{o_2} de grande valeur, l'expression précédente se simplifie sous la forme :

$$\frac{Vout}{Vin} \simeq \frac{gm_2}{gm_2 + gm_b} = \frac{1}{1+\gamma} \tag{4.7}$$

L'expression 4.7 indique ainsi un gain proche de l'unité tout en restant inférieur à l'unité. Nous restons bien ainsi dans une fonction de suiveur de tension.

4.3.3.2 Modélisation.

D'abord, nous avons imposé un courant de polarisation proche de 50nA, de façon à répliquer facilement via la recopie des références du système. Sous cette considération, nous avons obtenu, en utilisant des paramètres du transistor par extraction pseudo-empirique [17],[32],[27], [7], [27] une tension de seuil d'approx. 0.45V pour V_{tn} de MN2. La relation 4.7 indique que le gain est proche à l'unité.
Dans le cas du circuit utilisé, une approximation linéarisée possède la forme suivante :

$$g(x) = \begin{cases} 0, & \text{si } x<0.45 \\ 0.85(x - 0.45) & \text{si } 0.45 \leq x \leq 3.3 \end{cases} \tag{4.8}$$

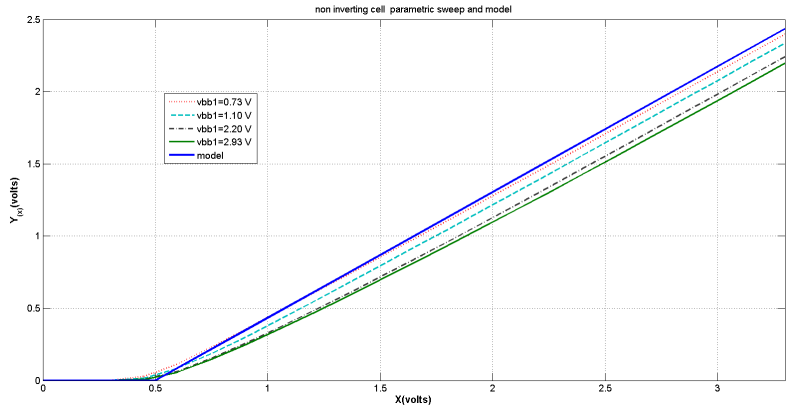

FIGURE 4.14 – Modèle utilisé versus simulation paramétrique de l'étage non inverseur.

Le modèle, simulé en figure. 4.15 est très proche des résultats de simulation.

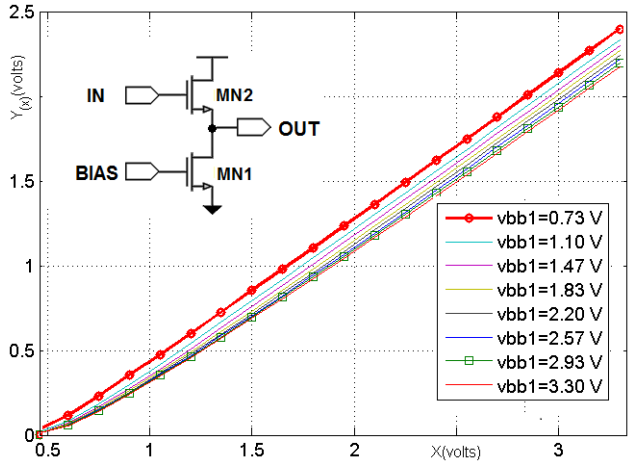

FIGURE 4.15 – Simulation paramétrique de l'étage non Inverseur.

4.3.3.3 Dimensionnement des composants.

Les tailles retenues par la structure sont présentées dans le tableau ci dessous.

Transistors	Dimensions
MN1	$\frac{1}{5}$
MN2	$\frac{10}{0.35}$

TABLE 4.4 – Dimensionnement de l'étage non inverseur.

4.3.4 Fonction non linéaire.

4.3.4.1 Description du circuit.

La fonction non linéaire est réalisée avec l'utilisation d'une topologie très élémentaire réalisée à partir de trois transistors, **MP1**, **MP2** et **MN1** (fig. 4.16). Ces trois transistors sont activés à des instants bien définis de la dynamique. Cela génère des discontinuités qui ressemblent à la fonction *tente* introduite précédemment.

Ainsi, les discontinuités de la fonction *tente* générées sont réalisées par l'utilisation des transitions des points de fonctionnement du circuit. Cette solution permet d'avoir l'effet de deux configurations différentes sur le même circuit. Ces deux modes sont activés de façon séquentielle en fonction de l'évolution de l'amplitude du signal d'entrée. Du fait des caractéristiques propre des transistors MOS, une région additionnelle de discontinuité est observable.

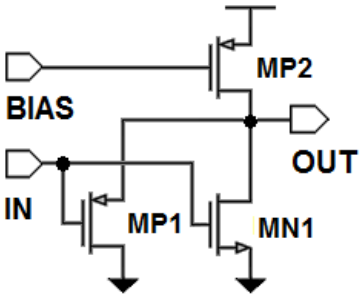

FIGURE 4.16 – Fonction non linéaire.

La fonction *tente* est composée par une caractéristique à pente positive suivie d'une caractéristique à pente négative. La pente positive de la courbe correspond à la fonction *PSF* (PMOS source follower) réalisée par la polarisation des deux transistors **MP2** et **MP1**. Durant cet instant, le transistor **MN1** est polarisé comme étant inactif, jusqu'à que le signal d'entrée atteigne la valeur de sa tension de seuil.

Ainsi, la tension V_{gs} du transistor **MN1** augmente progressivement, atteint puis dépasse sa tension de seuil. Il évolue alors progressivement vers la région de saturation, en formant un inverseur avec **MP2**. Durant cet instant, la tension V_{gs} du transistor **MP1** diminue progressivement jusqu'à désactiver complètement le transistor. Cette partie se distingue comme un changement sur la pente de la caractéristique du système, prenant une valeur négative liée à la transition entre la désactivation du PSF.

Enfin, le transistor **MN1** quitte son fonctionnement en saturation pour atteindre un mode de fonctionnement de type triode ou linéaire. Dans ce cas, la pente négative de la fonction *tente* devient moins forte.

Le transistor **MP2** est configuré comme source de courant pour les deux configurations (deux pentes de la tente) de fonctionnement du circuit. La polarisation de **MP2**, versus l'entrée **BIAS**, permet ainsi de varier le courant total consommé par le circuit. Il est possible de calibrer les différentes pentes de l'oscillateur mais également de passer d'un oscillateur périodique à chaotique.

La caractéristique DC de fonctionnement du circuit est représentée sur la figure 4.17, pour différentes valeurs de tension **BIAS**(vbb).

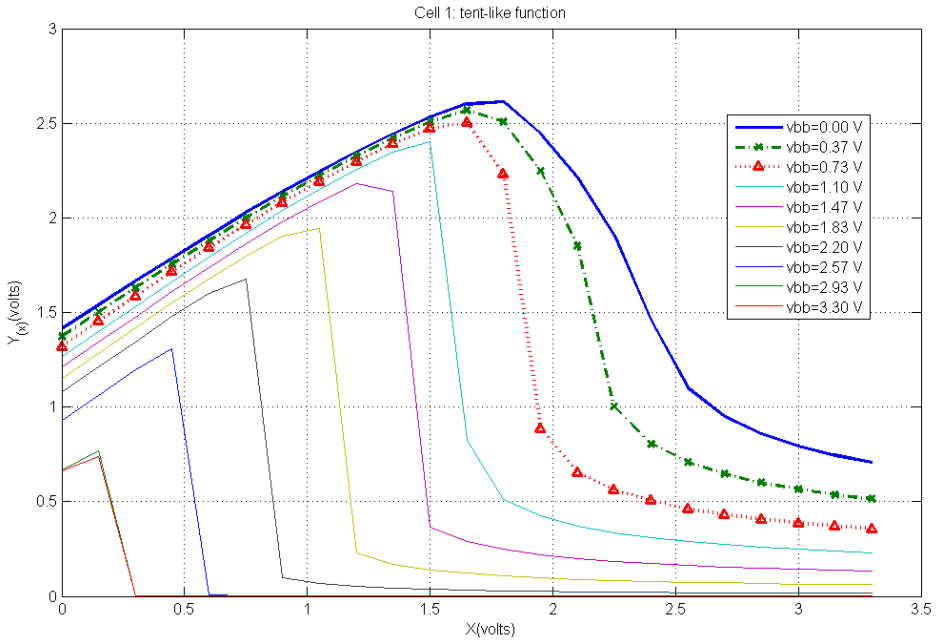

FIGURE 4.17 – Simulation paramétrique du circuit non linéaire type "tente"

4.3.4.2 Modélisation.

Afin de pouvoir calibrer notre circuit, une phase de modélisation a été nécessaire. Ainsi, nous avons utilisé une caractéristique qui s'approche du modèle proposé de la tente. Pour cela, nous allons préféré utiliser un gabarit qui s'approche de celui de la fonction tente symétrique, avec les deux discontinuités de base sur la dynamique totale. Un autre aspect du modèle linéarisé implique la représentation de chaque région de discontinuité par des droites.

En utilisant une approche similaire à celui du suiveur pour la cellule non linéaire, nous avons une caractéristique du suiveur telle que :

$$V_{out} \simeq V_{in} + |V_{tp_2}| \tag{4.9}$$

Initialement centré sur un élément de recopie du *biasing* du système, nous avons été obligés de redimensionner cet élément de récopie, pour favoriser une proximité

à la moitié de la dynamique, tant sur la sortie du système que sur les points de transition vers la fonction inverseuse sur le système. Donc nous avons une tension de polarisation arbitraire et inférieure à 1V.

Ainsi, nous devons réutiliser cette source de courant pour polariser le deuxième élément, correspondant à l'inverseur. Avec les effets induits par le *PSF*, due à la transition progressive de MP1, nous avons une pente plus élevée, laquelle progressivement se stabilisera sur une valeur constante vers la fin de la dynamique.

Nous avons accompagné la modélisation avec une co-simulation paramétrique, ayant comme but de vérifier l'évolution de la fonction due à la polarisation et aux effets non considérés dans les transitions du circuit (la figure 4.17 met en évidence la transition progressive de l'activation des deux transistors et les effets sur le comportement attendu).

Afin de conserver une approche de fonctions de premier ordre sur le modèle de la fonction de transfert (pour faciliter les démarches de calcul, itération), nous avons utilisé un modèle avec trois régions, de façon à approximer avec des droites le gabarit décrit antérieurement. Le modèle approximé retenu initialement est défini via le jeu d'équations :

$$f(x) = \begin{cases} 0.86x + 1.37, & \text{si } \mathrm{x} < 1.73 \\ -9x + 18.43, & \text{si } 1.73 \leq \mathrm{x} < 2 \\ -0.1x + 0.66, & \text{si } 2 \leq \mathrm{x} \leq 3.3 \end{cases} \tag{4.10}$$

Il faut remarquer que le modèle proposé peut contenir une imprécision sur sur la pente de la deuxième région de discontinuité (figure 4.18). Avec un nombre plus grand de données, nous constatons que la deuxième région de discontinuité possède une pente encore plus élevée.

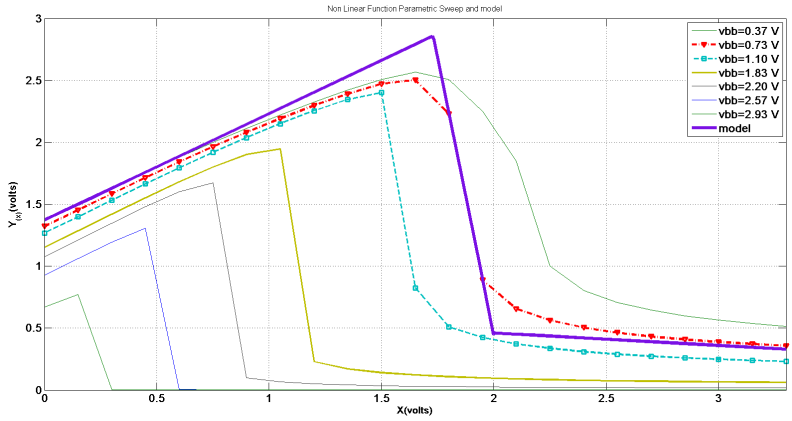

FIGURE 4.18 – Modèle simplifié de la fonction de transfert.

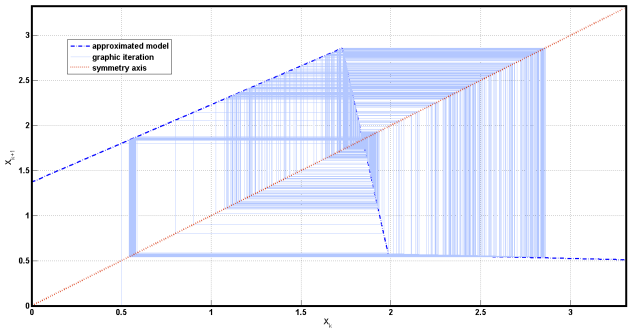

FIGURE 4.19 – Modèle linéarisé du comportement de la fonction discontinue avec itération graphique.

La simulation du modèle décrit (Eq. 4.10) avec le traçage des orbites par itération graphique est illustrée dans la figure 4.19. Nous pouvons apprécier que l'orbite décrite par les itérations n'est pas convergente et donc l'apparition d'un comportement chaotique centrée sur la moitié de la dynamique. Ceci correspondant à nos attentes sachant que même si les transitions entre les 3 régions d'opération peuvent être de deuxième ordre, leur impact va provoquer une réduction dans la dynamique mais en conservant une orbite fermée. Le circuit a été validé puis dimensionné.

Finalement, il faut tenir en compte de l'association des deux blocs principaux dans la boucle de l'oscillateur chaotique (le non inverseur et l'élément non linéaire). L'ensemble, sous un cas d'interrupteurs idéaux, produit l'orbite illustrée dans la figure 4.20. Cette orbite présente un comportement chaotique (la dynamique étant affectée par l'effet de la non linéarité du *NSF*).

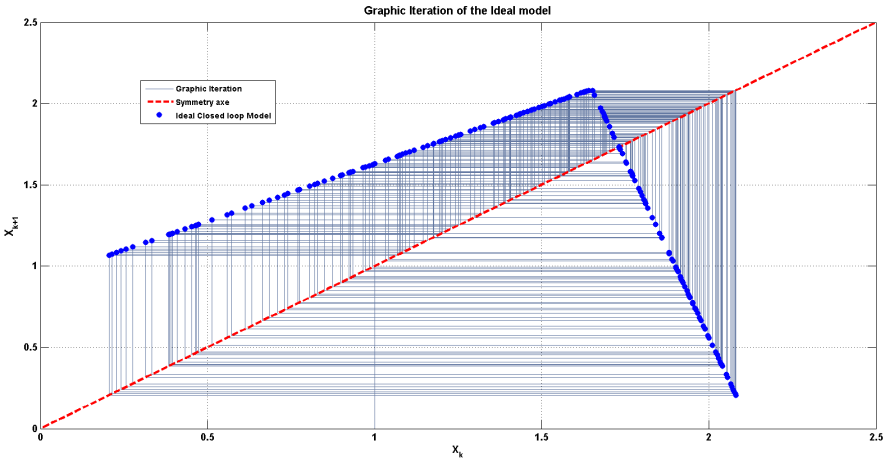

FIGURE 4.20 – Itération graphique du système idéal en boucle fermée.

4.3.4.3 Dimensionnement des composants.

Avec une utilisation croisée entre la modélisation approximée et la caractérisation par simulations paramétriques, les dimensions des transistors de la fonction non-linéaire ont été testées puis ajustées via un jeu de simulation successif afin de tendre vers une représentation des plus stabilisées. Ainsi, les tailles retenues sont proposées au sein du tableau ci-dessous.

MP1	MP2	MN3
$\frac{2}{0.45}$	$\frac{0.55}{5}$	$\frac{0.5}{0.45}$

TABLE 4.5 – Dimensionnement du circuit non linéaire

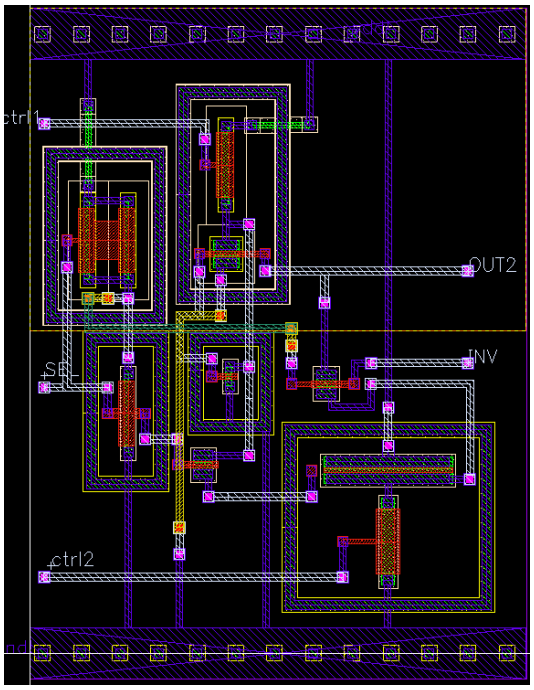

FIGURE 4.21 – Layout correspondant à l'oscillateur chaotique, incluant les interrupteurs, le suiveur et le bloc non linéaire.

4.3.5 Circuit de polarisation - Démarrage.

4.3.5.1 Principe.

Le circuit de polarisation a pour fonctionnalité de générer les niveaux de tension-courant nécessaires pour le fonctionnement du reste du système. Il est donc essentiel et une attention toute particulière doit y être apportée afin de ne pas avoir un système complet dysfonctionnant uniquement pour des raisons de niveaux mal placés.

Il est généralement constitué de références de tension et de courant devant avoir des performances optimales permettant au système de pouvoir osciller autour d'un point fixe.

Les références doivent avoir comme contraintes d'être indépendantes des variables physiques qui interviennent dans son fonctionnement, telles que la tension d'alimentation, la température et la technologie de fabrication (usuellement appelé indépendance en PVT pour *Process-Voltage-Temperature*)PVT . Cette caractéristique est d'autant plus souhaitable pour le bon fonctionnement des oscillateurs, miroirs de courant et amplificateurs. Une source de courant (ou référence de courant) est généralement une dérivée d'une référence de tension, réalisée à travers un circuit de conversion tension-courant. Ce dernier est, très succinctement abordé comme précédemment, réalisé via la polarisation d'une grille d'un transistor par une tension constante pour en obtenir un courant de drain constant.

Les semiconducteurs sont sensibles aux variations de la température, tension d'alimentation et aux processus de fabrication. L'indépendance à la tension d'alimentation est souvent très facilement réalisée via l'utilisation de fonction de type convoyeur de courant. Quant aux dépendances liées au processus de fabrication, des règles de placement-routage, lors du dessin des masques, permettent de réduire leurs influences. Une contrainte résiste dans la dépendance en température.

Il existe un grand nombre de références de tension disponibles dans la littérature scientifique, pouvant aller de références de type *bandgap*, de références aux différentielles de tension de seuil, référence à compensation d'effet... ([3], ++) Bien que les références de type *Bandgap* soient un incontournable, elles possèdent comme caractéristique une limitation de leurs tensions de sortie de référence à une valeur autour de la valeur de 1,23V. Cette référence dont la dénomination est liée à la tension de Gap des transistors possède comme limitation une tension de sortie élevée (pour des applications fonctionnant maintenant à des tensions inférieure au Volt) ainsi qu'une consommation et un encombrement pouvant être conséquents.

Toutefois, ces références *bandgap*, comme montrée en figure 4.23, sont à l'origine en terme de fonctionnement des références actuellement développées sur le marché. En effet, une référence de type *bandgap* repose sur l'association de deux cellules 4.23 (i) une référence PTAT dont la variable de sortie est proportionnelle à la température et (ii) une référence CTAT dont la variable est inversement proportionnelle à la température. Ainsi, la tension de sortie étant fonction de ces variables PTAT et CTAT devient indépendante de la température.

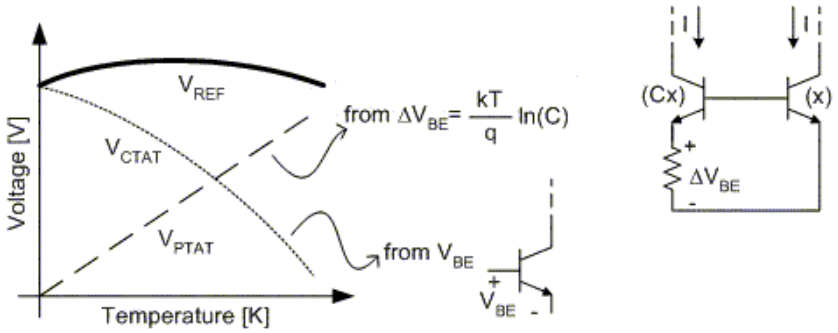

FIGURE 4.22 – Effet CTAT et PTAT et leur synthèse dans un système bandgap.

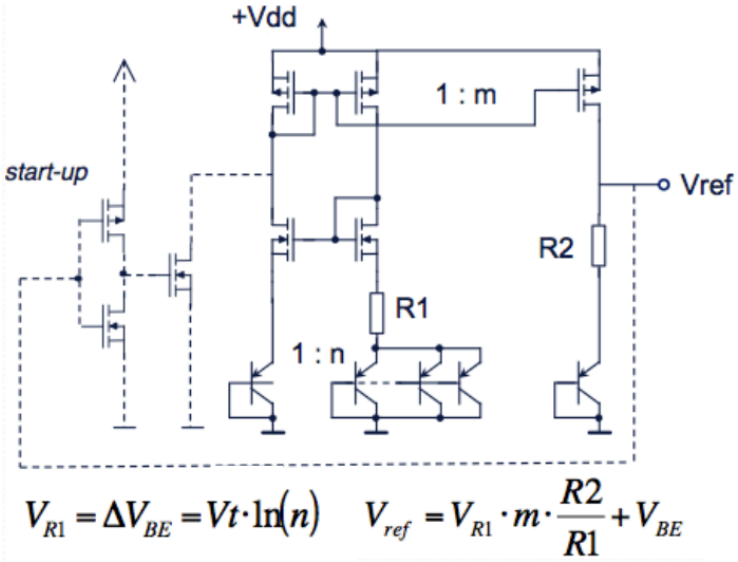

FIGURE 4.23 – Exemple classique de reference bandgap BiCMOS.

4.3.5.2 Référence PTAT.

Ainsi, la solution consiste à réaliser une référence de courant de type PTAT (*Proportional To Absolute Temperature*) qui va polariser une structure de type CTAT permettant de minimiser les dépendances en température. La référence de type PTAT est basée sur une structure classique, Fig.4.24.a à base de résistance.

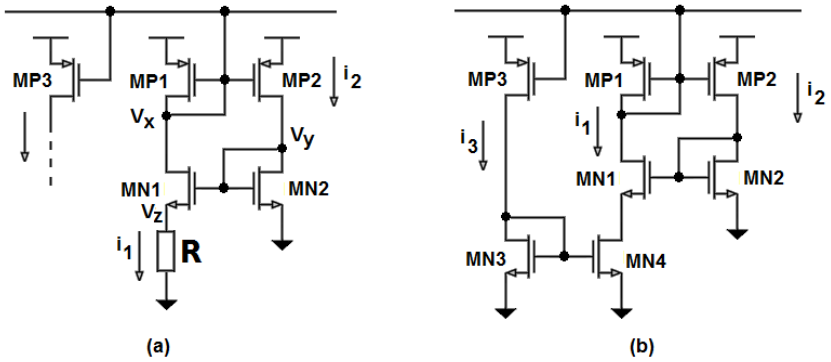

FIGURE 4.24 – (a) référence "classique" (b) référence Oguey.

Dans un souci de réduction de la consommation de la structure, les deux transistors PMOS, **MP1** et **MP2**, polarisés en faible inversion sont montés en structure de type miroir de courant. Les deux transistors **MN1** et **MN2** polarisés en faible inversion, permettent avec la résistance **R** de fixer le niveau de courant de la référence. Ceci est lié à l'architecture des composants **MN1**, **MN2** et **R** montés en convoyeurs de courant de premier ordre plus usuellement appelée *Miroir de Widlar*.
Les transistors polarisés en faible inversion, ont une modélisation dont le courant de drain du transistor est lié aux tensions de grille, de source de de drain avec un comportement de type exponentiel comme le montre l'expression ci dessous :

$$I_D = I_o \frac{W}{L} e^{\frac{V_G - V_T}{\eta U_T}} (e^{\frac{-V_S}{U_T}} - e^{\frac{-V_D}{U_T}}) \tag{4.11}$$

En réalisant le ratio des courants I_1 et I_2, via la modélisation proposée en Eq. (4.11) des deux transistors **MP1** et **MP2**, nous obtenons le ratio suivant :

$$\frac{I_{DP1}}{I_{DP2}} = \frac{I_{o_p} \frac{Wp_1}{Lp_1} e^{\frac{V_x - V_T}{\eta U_T}} (e^{\frac{-V_{dd}}{U_T}} - e^{\frac{-V_y}{U_T}})}{I_{o_p} \frac{Wp_2}{Lp_2} e^{\frac{V_x - V_T}{\eta U_T}} (e^{\frac{-V_{dd}}{U_T}} - e^{\frac{-V_x}{U_T}})} \tag{4.12}$$

$$= \frac{\frac{Wp_1}{Lp_1} (e^{\frac{-V_{dd}}{U_T}} - e^{\frac{-V_y}{U_T}})}{\frac{Wp_2}{Lp_2} (e^{\frac{-V_{dd}}{U_T}} - e^{\frac{-V_x}{U_T}})} \tag{4.13}$$

De même pour les transistors **MN1** et **MN2**, le ratio s'exprime sous la forme :

$$\frac{I_{DN1}}{I_{DN2}} = \frac{I_{o_n} \frac{Wn_1}{Ln_1} e^{\frac{V_y - V_T}{\eta U_T}} (e^{\frac{-V_z}{U_T}} - e^{\frac{-V_x}{U_T}})}{I_{o_n} \frac{Wn_2}{Ln_2} e^{\frac{V_y - V_T}{\eta U_T}} (e^0 - e^{\frac{-V_y}{U_T}})} \tag{4.14}$$

$$= \frac{\frac{Wn_1}{Ln_1} (e^{\frac{-V_z}{U_T}} - e^{\frac{-V_x}{U_T}})}{\frac{Wn_2}{Ln_2} (1 - e^{\frac{-V_y}{U_T}})} \tag{4.15}$$

Les deux équations précédentes (Eq. (4.12) et Eq. (4.14)) se simplifient sachant que le courant $I_1 = I_{DN1} = I_{DP1}$ et $I_2 = I_{DN2} = I_{DP2}$. Dans ce cas, nous avons alors :

$$\frac{\frac{Wp_1}{Lp_1}}{\frac{Wp_2}{Lp_2}} = \frac{\frac{Wn_1}{Ln_1}(e^{\frac{-V_z}{U_T}} - e^{\frac{-V_x}{U_T}})}{\frac{Wn_2}{Ln_2}(1 - e^{\frac{-V_y}{U_T}})} \tag{4.16}$$

Si nous nous mettons dans le cas où les tailles de transistors **MP1** et **MP2** sont équivalentes, nous avons alors $W_{p_1}L_{p_2}/W_{p_2}L_{p_1} = 1$. De même, si nous fixons un ratio pour les deux transistors **MN1** et **MN2** avec $W_{n_1}L_{n_2}/W_{n_2}L_{n_1} = M$. Enfin dans le cas d'une polarisation optimale avec V_x=V_y, l'équation précédente Eq. (4.16) se simplifie sous la forme de la relation proposée ci-dessous :

$$\frac{\frac{Wp_1Wn_2}{Lp_1Ln_2}}{\frac{Wp_2Wn_1}{Lp_2Ln_1}} = \frac{1}{M} = \frac{e^{\frac{-V_z}{U_T}} - e^{\frac{-V_x}{U_T}}}{1 - e^{\frac{-V_x}{U_T}}} \tag{4.17}$$

$$Me^{\frac{-V_z}{U_T}} - Me^{\frac{-V_x}{U_T}} = (1 - e^{\frac{-V_x}{U_T}}) \tag{4.18}$$

$$Me^{\frac{-V_z}{U_T}} = 1 + (M-1)e^{\frac{-V_x}{U_T}} \tag{4.19}$$

En assumant que : $M >> 1$ et $V_x >> U_T$

$$Me^{\frac{-V_z}{U_T}} = 1 + Me^{\frac{-V_x}{U_T}} \tag{4.20}$$

$$e^{\frac{-V_z}{U_T}} = \frac{1}{M} - e^{\frac{-V_x}{U_T}} \simeq \frac{1}{M} \tag{4.21}$$

Finalement :

$$V_z = U_T ln(M) = U_T ln(\frac{Wn_1Ln_2}{Wn_2Ln_1}) \tag{4.22}$$

Sachant que : $U_T = \frac{KT}{q}$ et en calculant le courant qui traverse la résistance R, défini par $V_z = R.I_1$, le courant de référence fourni par la référence PTAT est défini par :

$$I_{ref} = I_1 = \frac{V_z}{R} = \frac{KT}{Rq} ln(\frac{Wn_1Ln_2}{Wn_2Ln_1}) \tag{4.23}$$

La formule 4.23 nous permet d'apprécier la caractéristique PTAT
Enfin, ceci est d'autant plus problématique car pour une valeur de courant très réduite, il est nécessaire d'avoir une valeur de résistance très élevée. Ainsi, en dehors d'une dépendance en température de cette dernière d'autant plus conséquente, la surface silicium du composant sera d'autant plus conséquente.

4.3.5.3 Référence PTAT du type Oguey.

Une solution proposée par Oguey [11] consiste à remplacer la résistance précédemment problématique par son équivalent à transistor jouant le même rôle de définition du niveau de courant de la référence.

Une architecture très élémentaire, proposée dans la figure 4.24.b consiste à remplacer la résistance R par un transistor **MN4** polarisé en mode triode ou linéaire. Le transistor **MN4** est polarisé par le transistor **MN3** via un courant I_3, copie du courant $I_1 = I_{ref}$.

La Fig. 4.25 intègre à la référence d'Oguey, des blocs de démarrage et de recopie de courant et changement de tension de polarisation lui assurant son fonctionnement dans l'application finale.

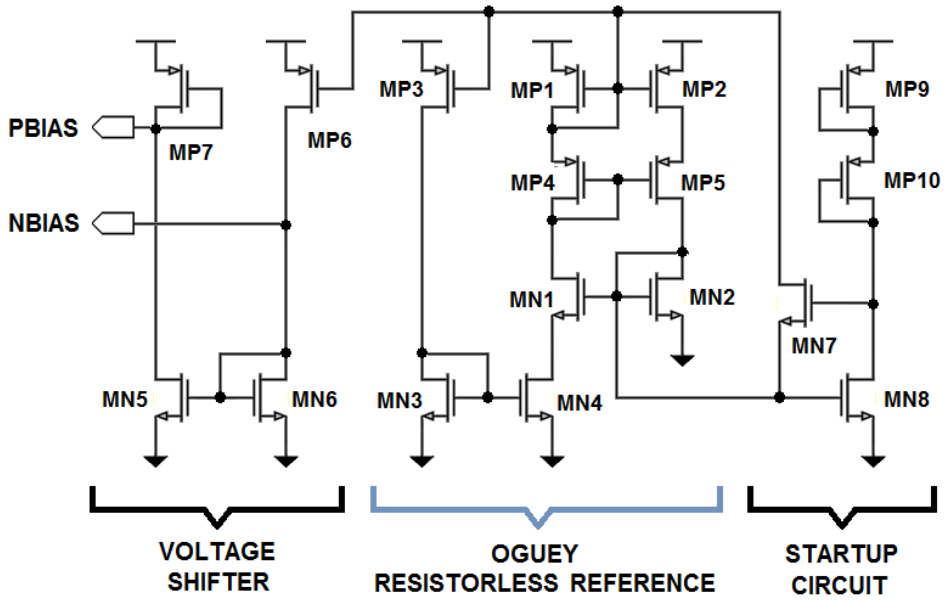

FIGURE 4.25 – Circuit de polarisation-démarrage

Ainsi, comme illustré dans la figure 4.25, la référence PTAT de type Oguey possède les trois éléments :

- **Un bloc de démarrage du système.(*start-up circuit*).** Il permet au circuit de ne pas rentrer dans un état metastable de courant nul lors du démarrage. Dans un état initial, **MN7** est activé et permet d'injecter du courant au reste du circuit. Puis, progressivement, la tension sur la grille de **MN8** augmente pour atteindre la tension de seuil, permettant son activation. Dans ce cas, la tension de drain du transistor **MN8** devient très faible, ce qui désactive finalement **MN7**. Pendant le reste du temps, **MN7** reste désactivé n'ajoutant pas une consommation supplémentaire au circuit.
- **Le circuit de référence Oguey**. Sur le circuit de base de la Fig. 4.24.b initialement introduit, les miroirs de courant simple de type PMOS ont été remplacés par des étages de type cascode afin de modifier l'impédance de sortie. En utilisant la première référence obtenue, nous avons changé le régime

d'opération en utilisant les éléments de décalage en tension (*voltage shifter*) et en conservant le courant de référence. La sortie de cette nouvelle référence est réutilisé pour l'ajuster au reste des blocs, qui seront répliqués par des miroirs de courant.

– **Un bloc de recopie des courants - changement de niveaux de tension (*voltage shifter*).** Cet étage permet de dissocier la fonction référence de la fonction charge (**NBIAS, PBIAS**). Ceci est réalisé via l'ajout de miroir de courant **MP1**/**MP6** et **MN6**/**MN5**.

4.3.5.4 Dimensionnement des composants.

Les tailles retenues pour le montage référence PTAT de type Oguey sont proposés dans le tableau ci dessous.

Voltage Shifter				Oguey Référence									Startup Circuit			
MP7	MN5	MP6	MN6	MP3	MN3	MP1	MP4	MN1	MN4	MP2	MP5	MN2	MP9	MP10	MN7	MN8
$\frac{2}{15}$	$\frac{1}{5}$	$\frac{0.4}{100}$	$\frac{1}{5}$	$\frac{0.4}{100}$	$\frac{1.2}{1.5}$	$\frac{0.4}{100}$	$\frac{0.4}{100}$	$\frac{20x0.4}{10}$	$\frac{20x1.2}{1.5}$	$\frac{0.4}{100}$	$\frac{0.4}{100}$	$\frac{0.4}{10}$	$\frac{200}{0.35}$	$\frac{0.4}{100}$	$\frac{0.4}{100}$	$\frac{0.4}{100}$

TABLE 4.6 – Dimensionnement du circuit de polarisation

Les contraintes attendues pour ce circuit sont la génération d'un courant de polarisation fixé à I_{BIAS}= 25nA. Les références de tension également générée par cette référence de courant sont égales à P_{BIAS}=2.5V et N_{BIAS}=0.5V. Le temps de démarrage de la structure est de l'ordre de 50μs.

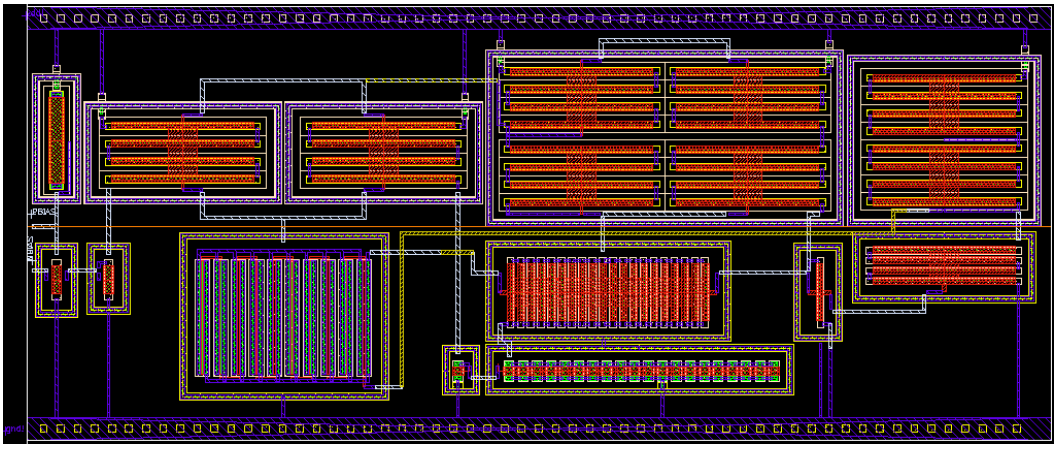

FIGURE 4.26 – Layout correspondant au circuit de polarisation-démarrage.

4.4 Résultats.

Après une phase de modélisation permettant de calibrer la structure afin d'en déduire la taille des composants, la structure a été simulée en phase de pré et post-Layout. Les deux étapes sont essentielles pour observer l'impact des éléments parasites rajoutés pendant le routage. Bien évidemment, nous ne prenons pas en compte l'effet des plots ou l'implantation mécanique du composant dans cette étape.

Concernant la réponse transitoire nous pouvons observer une diminution sur la valeur nominale du courant de polarisation équilibré sur les deux sources de courant-tensions de biasing. Il existe un incrément sur le temps d'établissement.

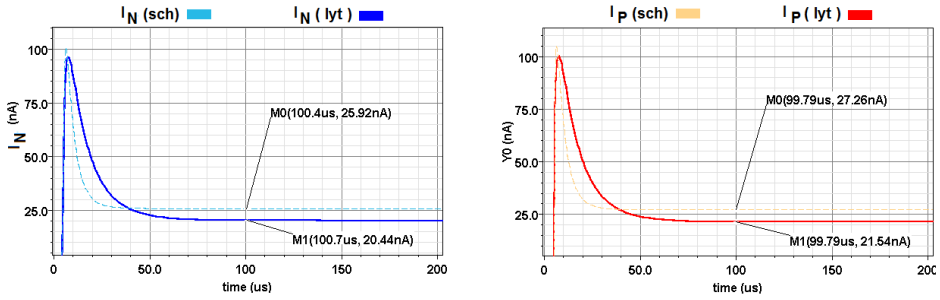

FIGURE 4.27 – Démarrage du circuit
(simulations transitoires pre -post Layout).

Par rapport aux variations dues à la température, le circuit possède la caractéristique linéaire attendue. Cependant, les variations avec la température sont plus prononcés.

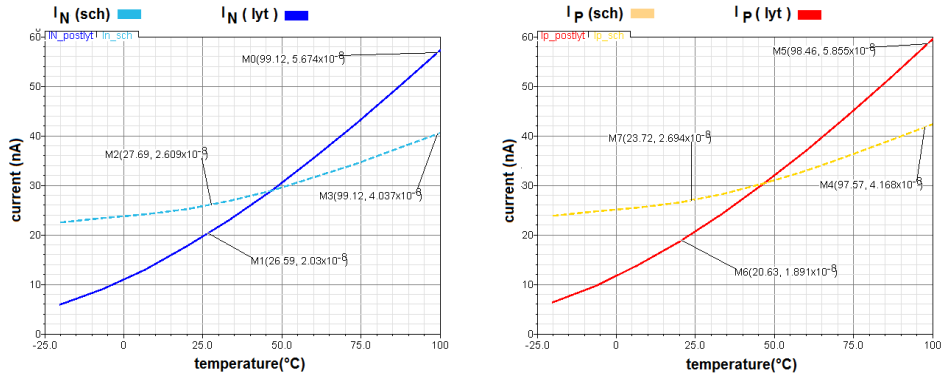

FIGURE 4.28 – Variation du courant avec la température
(simulations pré-post Layout).

De l'autre coté, les tensions utilisées pour polariser les grilles des autres transistors mis en région de saturation sont plus stables, présentant une déviation de 10% sur leurs valeurs lors du routage, donc une pente $\Delta V/\Delta T$ moins prononcée.

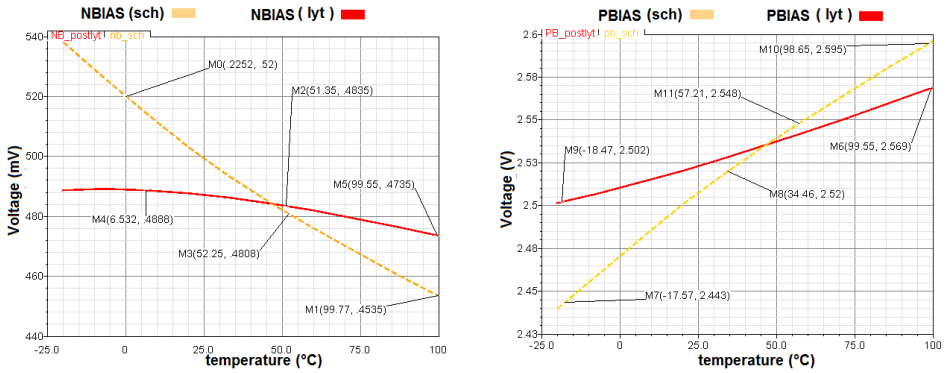

FIGURE 4.29 – Variation des tensions versus température (pré-post Layout).

4.5 Circuit de pilotage de l'oscillateur chaotique

Le bloc précédemment introduit permet la polarisation des courants de référence nécessaires au bon fonctionnement de la plupart des blocs du système, incluant l'oscillateur chaotique.

L'oscillateur chaotique en temps discret possède des cellules qui peuvent avoir une sensibilité aux variations du au routage et à l'implantation sur la puce de test (notamment l'effet des plots, tant que le circuit de base est conçu pour gérer une étape de faible consommation et faible capacitance de charge, en profitant de la partie de conversion vers le numérique pour monter en capacitance).

Ce fait impliquera que sur la réalisation d'un testchip avec des plots pour l'observation des différents valeurs de signaux que doivent rester normalement internes au système, ces plots ammènent des variations sur les tensions de polarisation des cellules intégrant l'oscillateur chaotique.

Le rôle de ce bloc est donc de nous permettre d'avoir l'option de faire une calibration *à posteriori* si nécessaire sur les tensions du circuit.

Au défaut, le circuit possède des éléments de polarisation de grille prédéfinis.

4.5.1 Description détaillée du circuit.

Ce circuit permet de gérer les tensions de polarisation requises pour l'oscillateur chaotique et le comparateur de sortie.

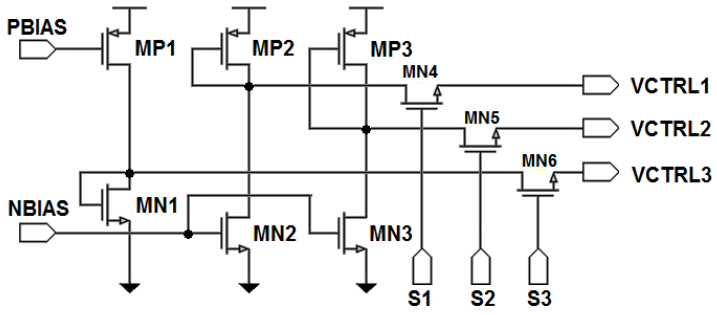

FIGURE 4.30 – Schématique du circuit de calibration.

Le bloc (figure 4.30) est constitué de trois circuits de *"level shifting"*, représentés par les paires respectives des transistors **(MP[1-3],MN[1-3])**. Sur chacune des paires est placé un interrupteur **(MN[4-6])**, les interrupteurs assurent la connexion des éléments de polarisation interne aux étages suivants ou alors le pilotage en externe des niveaux de polarisation.

L'activation des signaux de pilotage s'effectue à travers des entrées **S1-S3**. Un niveau bas permet d'utiliser la référence interne et un niveau haut permet d'utiliser un signal de tension externe pour la calibration.

4.5.2 Dimensionnement des composants.

Les tailles retenues pour les transistors ont été sélectionnés afin de fournir des courants de 50nA par voie, avec des tensions de pré-calibration obtenues par simulation paramétrique du pré et post Layout. Nous avons choisi dans la plage de valeurs générés les suivantes : $V_{CTRL1} : 0.7V$, $V_{CTRL2} : 0.7V$, $V_{CTRL3} : 0.5V$.

Transistors	Dimensions	Transistors	Dimensions
MP1	$\frac{2}{15}$	MN1	$\frac{3}{5}$
MP2	$\frac{0.5}{25}$	MN2	$\frac{1.1}{5}$
MP3	$\frac{2}{15}$	MN3	$\frac{1}{5}$
MN(4 :6)	$\frac{1}{0.5}$		

TABLE 4.7 – Dimensionnement du circuit programmable de références.

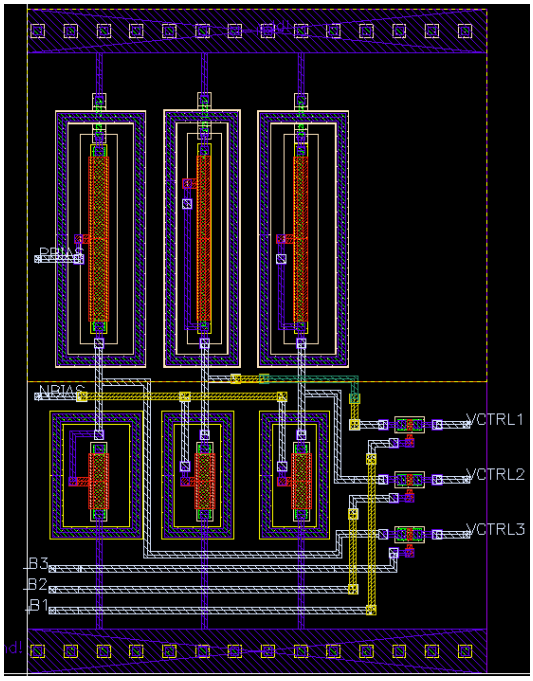

FIGURE 4.31 – layout du circuit de calibration.

4.5.3 Mise en marche de l'oscillateur chaotique.

4.5.3.1 Modélisation du fonctionnement.

Comme indiqué dans la description du bloc non linéaire (figure 4.20), le système présente des orbites chaotiques. La figure 4.35 résume un comportement en temps continu, représenté dans un cas idéal avec un effet d'échantillonneur- bloqueur idéal, qui au même temps utilise une dynamique approximée des trois régions du modèle considéré.

Ce modèle comportemental de base ne prend en compte que des sur tensions-sous tensions dues aux transitions de commutations qui peuvent générer des saturations instantanées sur la dynamique de l'oscillateur (en conséquence introduisant encore plus d'incertitudes sur les trajectoires de retour). Ainsi, les effets capacitifs sur l'établissement du signal sont négligés en assumant une valeur stable à chaque itération de la boucle. Le modèle idéal présente donc une dynamique correspondant à approximativement 2 volts (figure 4.35).

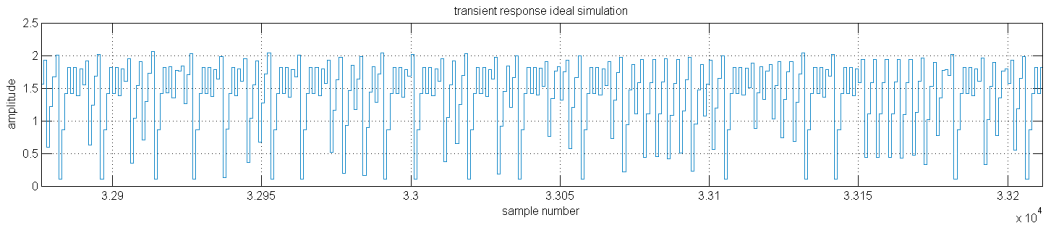

FIGURE 4.32 – Simulation transitoire du modèle idéal sous Matlab.

4.5.3.2 Simulations du système chaotique

Les figures 4.35 et 4.33 nous permettent de visualiser les variations non périodiques des signaux de sortie. Le modèle idéal ne contient pas de paramètres liés à la propagation du signal, ce qui justifie un affichage comparatif en termes d'échantillons et pas d'instant de temps précises. Ainsi, comme indiqué précédemment, les deux systèmes sont proches mais c'est le modèle de $BSIM$ qui contient le vrai modèle non linéaire avec les transitions de la fonction *tente* correctement définis.

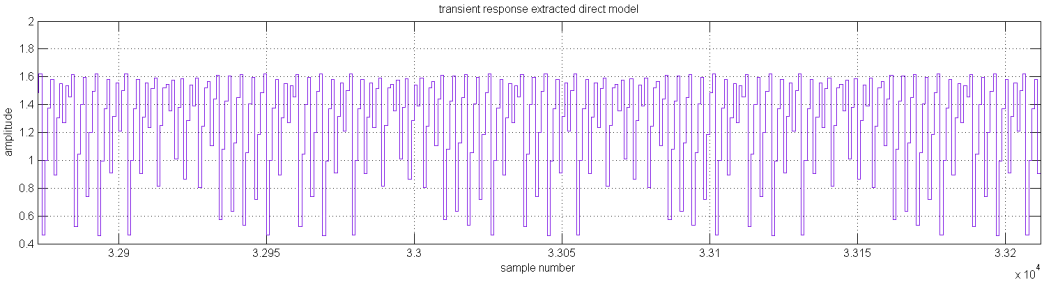

FIGURE 4.33 – Simulation transitoire (BSIM4).

La figure 4.34 met en évidence les effets de transition de commutation inexistants sur le modèle comportemental, pouvant provoquer des effets de saturation ou déviations sur le tracé de la caractéristique des itérations attendu (nous avons superposé aussi une version prennant en compte les valeurs stabilisés du signal).

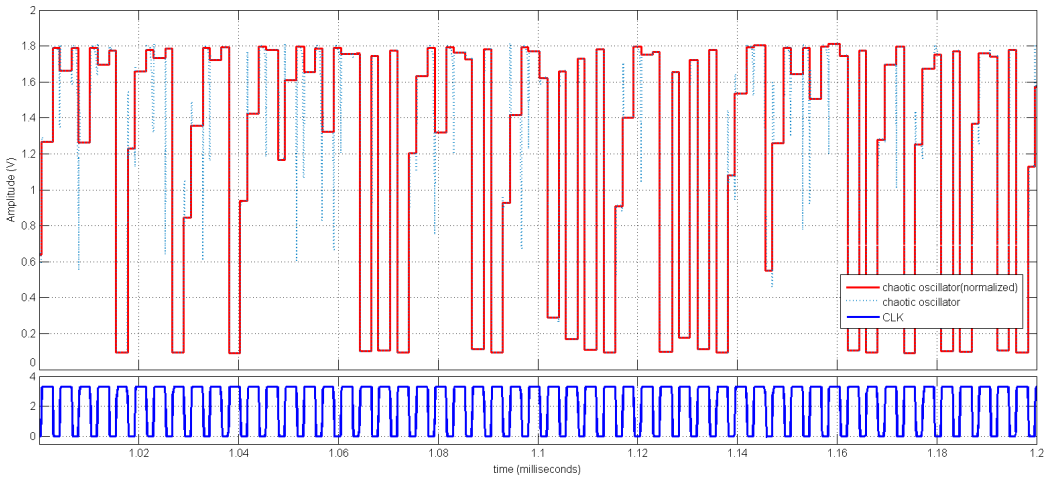

FIGURE 4.34 – Détail de la réponse transitoire indiquant les effets de commutation.

A partir des données obtenues du système, nous avons tracé le portrait de phases correspondant. La figure 4.35 nous permet d'illustrer les différences existantes par rapport au modèle. La première différence importante correspond, comme attendu, à la pente et le point de la dynamique liée à la transition correspondant à la désactivation du *psf*. L'autre différence consiste à la réduction de la dynamique.

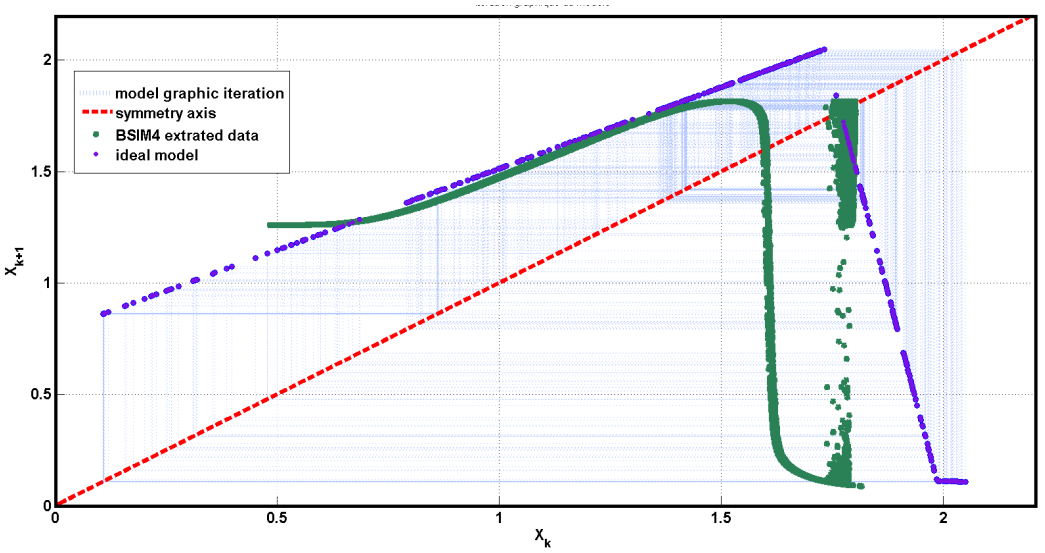

FIGURE 4.35 – comparaison des espaces générés par le modéle idéal et le modèle sous simulation.

4.5.3.3 Caractéristique chaotique du signal.

Afin de vérifier la caractéristique chaotique de l'oscillateur, nous allons utiliser la méthode des exposants de Lyapunov, étudiée dans le paragraphe 3.3.3.

Le calcul des exposants de Lyapunov (LLE : Largest Lyapunov Exponent) permet d'avoir une idée qualitative de la dynamique d'un système. Dans le cas d'un espace de phases unidimensionnel :

- $LLE = 0$, caractérise une orbite marginalement stable
- $LLE < 0$, caractérise une orbite périodique ou un point fixe
- $LLE > 0$, caractérise une orbite chaotique.

Accessoirement, nous avons pris différents blocs parmi les échantillons récupérés par simulation (normalisés sur l'état stable à chaque coup d'horloge pour réduire l'espace mémoire requis pour le calcul). Nous avons exécuter l'algorithme de Lyapunov-Rosenstein, proposé par [29], pour le calcul du plus grand exposant de Lyapunov à partir d'une série temporelle.

L'implémentation proposée de cet algorithme possède la particularité de ne pas dépendre de la fréquence d'échantillonnage des données pour apporter des informations sur la sensibilité au changement des conditions initiales.

L'évolution de $\log(\gamma)$, en fonction du nombre d'échantillons pris lors de l'analyse converge vers une valeur déterminée au bout d'un nombre d'échantillons, ce qui permet de réduire le nombre de points utilisés sur l'analyse pour obtenir des résultats.

L'algorithme indique un LLE supérieur à zéro (approx. 0.67), indiquant la présence du chaos sur le signal de sortie.

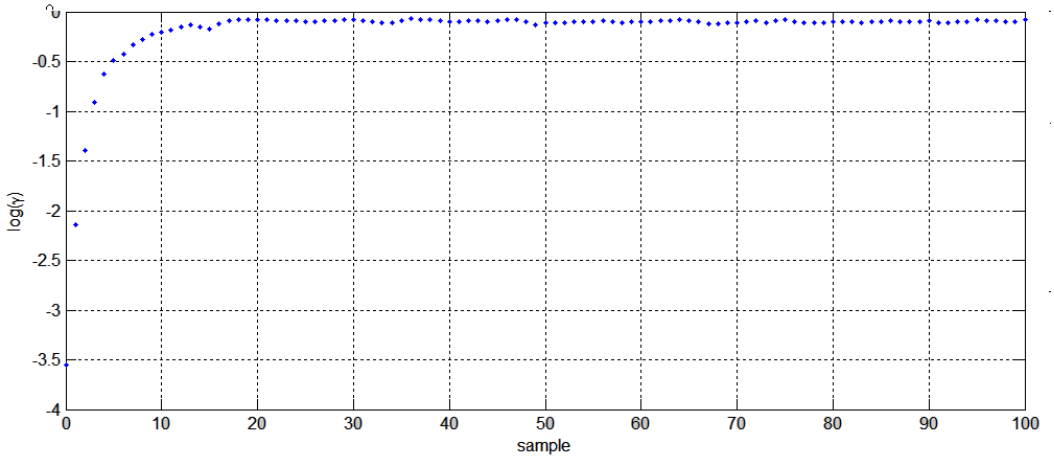

FIGURE 4.36 – Application de l'algorithme de Rosenstein.

4.6 Conclusion.

Nous avons présenté les démarches de dimensionnement et les caractéristiques obtenues pour l'oscillateur chaotique. Une étape de modélisation mathématique et comportementale a été réalisée afin d'avoir une idée de l'ordre de grandeur des résultats, toutefois, un modèle néglige une quantité de caractéristiques intrinsèques mais sert comme référence du comportement général du système.

Après une étape de simulation, une élaboration à base de transistors a été effectuée. Une telle démarche ajoute des éléments non linéaires intrinsèques aux caractéristiques des composants, changeant le gabarit de la dynamique prévue dans le modèle de base, sans modifier par autant le comportement global.

Les simulations apportent des informations qui permettent de confirmer la tendance du système, sous les valeurs de calibration, vers des oscillations non périodiques. La vérification se manifeste dans le gabarit de la dynamique de l'espace de phases unidimensionnel ainsi que sur le calcul du plus grand exposant de Lyapunov (LLE).

Nous allons utiliser ce circuit pour la réalisation du circuit d'échantillonnage pour l'obtention des signaux binaires dans le générateur de valeurs aléatoires.

Chapitre 5

Implémentation du TRNG

5.1 Introduction.

Nous avions vu dans le chapitre 2 les différentes solutions de réalisation d'un TRNG (amplification directe du bruit, échantillonnage d'oscillateurs, utilisation de fonctions physiquement non-clonables, échantillonnage de signaux chaotiques) avant de conclure sur l'intérêt de la solution d'échantillonnage de signaux chaotiques.

Ainsi, étant donné l'importance de la génération de signaux chaotiques, une étude des différentes architectures possibles répondant à nos contraintes de basse consommation a été proposée au chapitre 4. De cette étude, une solution optimale versus le cahier des charge a été testée et validée par simulation. Celle-ci est donc intégrée dans la structure TRNG proposée dans ce chapitre à laquelle viendront se rajouter quelques blocs dont la description sera proposée.

Enfin, une validation de la structure sera réalisée et mise en avant par rapport aux solutions concurrentes.

5.2 Architecture du TRNG Proposé.

Le principe d'une architecture d'oscillateur chaotique, utilisant une topologie incluant une fonction non linéaire discrète a été déjà documenté dans la littérature (i.e. [6],[35],[20]).

Nous allons utiliser dans le circuit comme générateur du signal à échantillonner la sortie de l'oscillateur chaotique et les circuits associés décrits dans le chapitre 4.

Nous associons un étage de conversion de cette source d'incertitude afin d'obtenir des patrons binaires de caractéristique aléatoire.

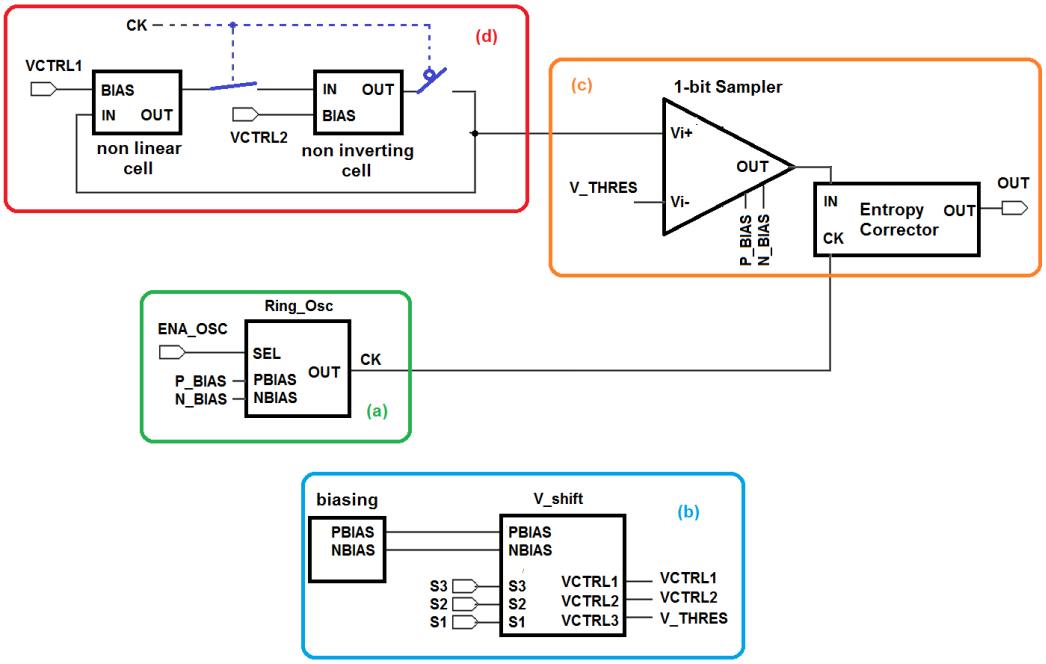

FIGURE 5.1 – Diagramme en blocs du TRNG proposè

5.2.1 Description du système.

Cette architecture en temps discret est constituée par des éléments de base présentés dans la figure 5.1 qui sont :

(a) Un oscillateur périodique, générant un signal de cadencement du système.

(b) Un circuit de polarisation du système.

(c) Un oscillateur chaotique générant un signal avec des variations d'amplitude non périodiques, utilisé comme source d'entropie.

(d) Un convertisseur du signal analogique chaotique en un signal binaire aléatoire. Il est accompagné d'un correcteur numérique d'entropie qui assure une distribution binaire uniforme des niveaux logiques '1' et '0'.

Enfin la nature chaotique, non prédictif de la sortie de l'oscillateur chaotique, permet une utilisation comme une source qui ressemble au bruit amplifié qui sera transformé postérieurement vers des variations binaires erratiques.

5.2.2 L'oscillateur chaotique.

L'oscillateur chaotique, avec le circuit de *biasing*, correspond au système décrit au sein du chapitre 4. Ainsi, comme introduit dans la figure 4.3, l'oscillateur chaotique à temps discret est constitué :

- D'un générateur d'horloge générant un signal d'horloge **CK** oscillant entre les tensions 0 et 3,3V pour une fréquence de 250kHz.
- D'un élément non linéaire et un étage suiveur qui forment une boucle de mémorisation cadencé par **CK**, en réalisant des itérations qui emmènent le circuit dans un état chaotique.
- D'une référence de tension qui permet de fournir au convertisseur 1 bit deux références de tension P_{BIAS} et N_{BIAS} dont les valeurs sont respectivement égales à 2,5V et 0,5V ainsi que les tensions de polarisation de la boucle non linéaire.

5.2.3 Convertisseur à 1 bit.

Le convertisseur à 1 bit est réalisé à partir d'un comparateur de tension. Ainsi, les signaux provenant de la sortie du bloc oscillateur chaotique, versus la sortie **OUT** sont discriminées à l'aide du convertisseur via une comparaison avec la tension de seuil V_{THRES} fournie par une référence de tension. Ainsi, les signaux générés sont binaires et peuvent être ensuite exploités par des routines numériques de quantification.

Avant de s'attaquer à la conception en tant que telle du circuit convertisseur 1 bit, une phase de modélisation a été réalisée afin de justifier de l'approche orientée par la solution proposée.

5.2.3.1 Modélisation comportementale du convertisseur 1 bit.

Afin d'appréhender la modélisation comportementale d'un convertisseur 1 bit, un certain nombre d'hypothèses doivent être définies en amont. Ainsi, nous allons considérer un comparateur possédant des caractéristiques idéales en bande passante-vitesse, soit un temps de propagation négligeable. En revanche, nous allons considérer la caractéristique d'hystérésis comme faisant partie du comparateur.

En effet, le fait de modéliser l'hystérésis nous permettra de prévoir le comportement du comparateur dans un statut de pire cas (*"worstcase"*). Cela est particulièrement identifiable lorsque le comparateur doit gérer les pics de surtension (*"overshoots"*) lors de la commutation du signal mais également lorsqu'il fait face à des variations instantanées pouvant modifier l'état de sortie

Le phénomène d'hystérésis se manifeste comme une dépendance du signal de sortie sur l'état actuel de l'entrée, ainsi que de l'antérieur à celui-ci. Dans le cas du comparateur de tension, ceci signifie le fait d'avoir deux seuils de comparaison : un seuil haut (V_{TRIP+}) et un seuil bas (V_{TRIP-}), utilisés respectivement lors de la croissance et décroissance du signal d'entrée. (figure 5.2)

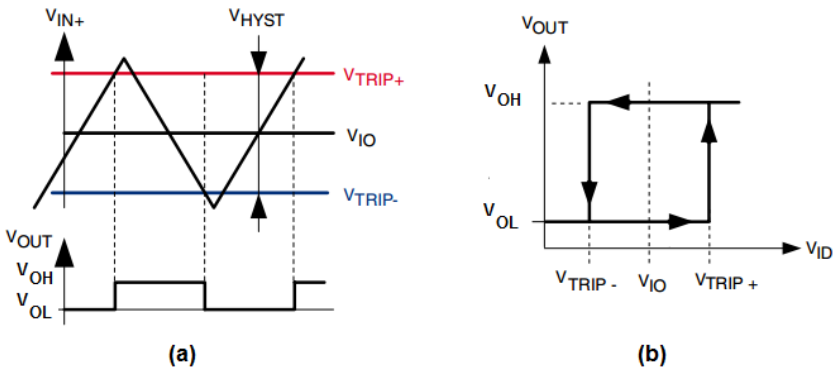

FIGURE 5.2 – Comparateur (a) Modèle (b) Caractéristique entrée V_{ID}-sortie V_{OUT}

Pour notre application (Fig.5.2.(a)), une approximation linéarisée est définie via la mise en équation suivante :

$$V_{OUT(V_{IN})} = \begin{cases} V_{OL}, & \text{si } V_{IN} \leq V_{TRIP-} \\ V_{OL}, & \text{si } V_{TRIP-} < V_{IN} < V_{TRIP+} \text{ et } V_{IN_{actuel}} < V_{IN_{anterieur}} \\ V_{OH} & \text{si } V_{TRIP-} < V_{IN} < V_{TRIP+} \text{ et } V_{IN_{actuel}} > V_{IN_{anterieur}} \\ V_{OH}, & \text{si } V_{IN} \geq V_{TRIP+} \end{cases} \tag{5.1}$$

Si à partir des équations précédents, nous exprimons la taille de la bande d'hystérésis ΔV_{TRIP} définie par :

$$\Delta V_{TRIP} = V_{TRIP+} - V_{TRIP+} \tag{5.2}$$

Nous pouvons exprimer les tensions de seuil haute et basse, ΔV_{TRIP+} et ΔV_{TRIP-}, en fonction de la tension de seuil moyenne V_{IO} via les deux équations suivantes :

$$V_{TRIP+} = V_{IO} + \frac{\Delta V_{TRIP}}{2} \tag{5.3}$$

$$V_{TRIP-} = V_{IO} - \frac{\Delta V_{TRIP}}{2} \tag{5.4}$$

Afin de répondre aux contraintes du cahier des charges précédemment fixé par les états fournis en sortie de l'oscillateur chaotique, la modélisation comportementale du comparateur 1 bit doit posséder les caractéristiques suivantes :

- V_{IO} défini par un élément externe.
- $\Delta V_{TRIP} = 20mV$
- $V_OL = 0V \rightarrow$ "0"
- $V_OH = 3.3V \rightarrow$ "1"

La bande d'hystérésis a été volontairement réduite à une valeur inférieure à 100mV (sur les simulations comportementales, nous utilisons une valeur de 20mV, i.e.), ceci représentant moins d'un dixième de la dynamique. Ainsi, la modélisation prend en

compte l'effet de l'hystérésis et son impact sur des conditions extrêmes de variations instantanées sur la dynamique (figure 5.4).

A fin d'économiser de la mémoire dans l'analyse de donnés, nous prenons des valeurs stables de tension à la sortie de l'oscillateur chaotique, à intervalles réguliers fixés par la fréquence de l'oscillateur de référence. L'inconvénient de ce modèle est l'absence des transitions dûs à la commutation des interrupteurs.

La présence de telles variations de durée instantanée peuvent avoir deux effets possibles sur la sortie du comparateur. Dans un cas idéal, des transitions instantanées instables se présentent dans ces instants. Dans une autre possibilité, l'hystérésis peut provoquer un état de mémorisation par rapport à l'évolution montée ou descente des variations de commutation.

Afin d'évaluer et de valider le comportement du modèle, nous avons considéré de telles variations. La figure 5.3 illustre le signal provenant de l'oscillateur chaotique vu dans le chapitre 4, avec des sur-sous tensions instantanées, aussi que la version plus optimiste du modèle.

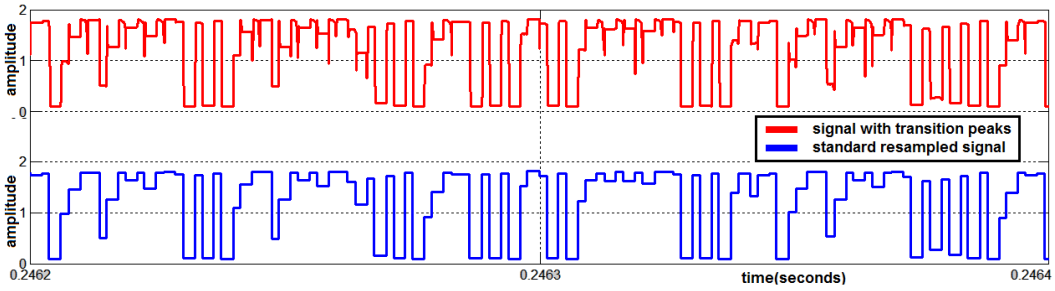

FIGURE 5.3 – Signal d'entrée du comparateur (en considérant ou pas les transitions de commutation).

La figure 5.4 illustre, sur le signal optimiste du modèle, l'action du comparateur avec et sans la bande d'hystérésis. Avec le seuil placé proche de la limite supérieure de la dynamique, $V_{THRES} = 1{,}75\text{V}$ (où les surtensions transitoires apparaissent le plus souvent). Nous pouvons observer la différence entre les deux modes qui se traduit par un effet mémoire sur les transitions consécutives, qui représente un incrément sur la quantité de zéros (ou de uns) sur la suite binaire de sortie.

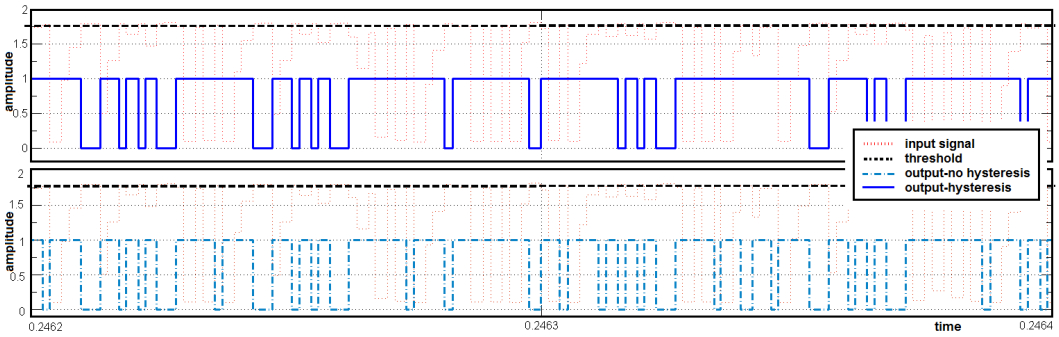

FIGURE 5.4 – Effet de l'hystérésis sur le modèle du comparateur (V_{THRES} = 1,75V).

Afin de vérifier le comportement devant des variations plus écartées , nous allons placer le seuil dans une région vers le milieu de la dynamique ($V_{THRES} = 1$V). Nous pouvons constater (figure 5.5) que l'opération du comparateur est plus classique. La modélisation de l'hystérésis n'a pas d'impact que ce soit positif ou négatif.

Nous obtenons alors bien le comportement classique d'un comparateur comme le montre la figure 5.5 , ceci étant justifié par une gamme de variation assez grande de la sortie de l'oscillateur chaotique. Pour une valeur basse, de la tension de seuil, les résultats sont identiques.

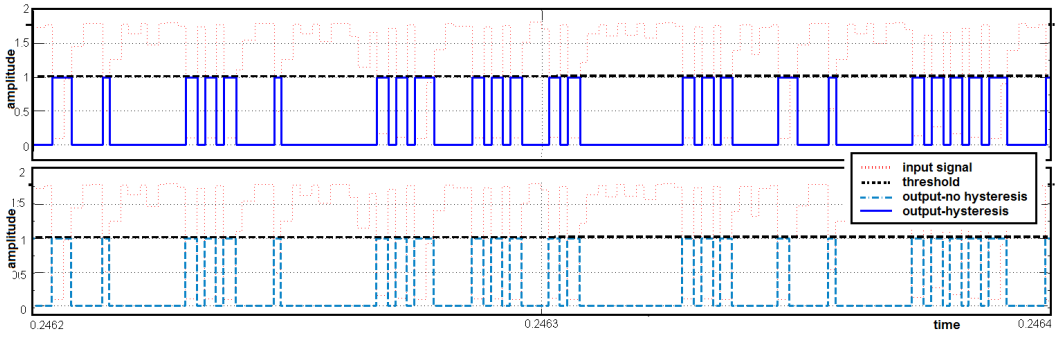

FIGURE 5.5 – Effet de l'hystérésis sur le modèle du comparateur.

La figure 5.6 illustre en termes des distributions binaires, une moyenne de résultats sur les signaux générés par le modèle de l'oscillateur chaotique.

Nous pouvons constater que la distribution binaire suit l'évolution de la tension de seuil. Nous avons donc une région près de la moitié de la dynamique possédant une distribution uniforme mais de variations très peu fréquentes, donc ayant plus de possibilités d'avoir des séquences proches du périodique. Nous allons donc favoriser la plage supérieure de la dynamique avec des variations abruptes et erratiques, qui permettront de favoriser une sortie plus équilibrée en utilisant la compensation numérique après.

L'autre point important correspond aux différences entre l'ajout d'hystérésis. La distribution pressente une variation plus importante dans le cas de seuillage dans la partie supérieure de la dynamique. Dans ce cas, l'incrément de 1's , comme vu dans la figure 5.4 implique une transition perdue et pas un 1 incrusté entre un bloc de 0's. Cette situation correspond au *worstcase*, dans lequel nous perdons de l'information en augmentant des séries de valeurs monotones.

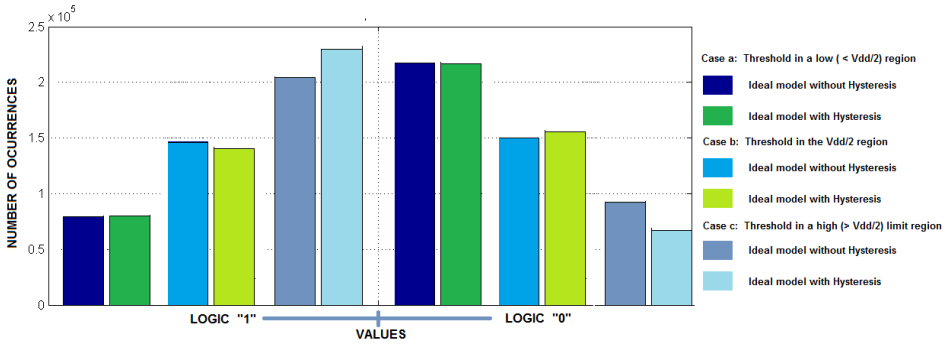

FIGURE 5.6 – L'effet de l'hystérésis dans trois placements du seuil.

Étant donné les résultats précédemment énoncés, nous avons décidé de favoriser le modèle avec hystérésis et réaliser des tests permettant d'en définir le pire cas (*"worstcase"*) de fonctionnement. Ce pire cas, nous permettra ensuite de définir la configuration à appliquer à un correcteur d'entropie placé en aval du comparateur 1 bit et dont la présentation sera abordée dans la suite de ce manuscrit.

Aux simulations précédemment abordées, les pics de commutations instantanées n'ont pas été pris en compte dans les transitions du signal. Ces pics devraient rajouter beaucoup plus d'aléa mettant en avant le modèle avec hystérésis. nous allons comparer à postériori l'hypothèse avec le fonctionnement obtenu sur les signaux de sortie de l'oscillateur chaotique, lors de la phase de simulation du circuit proposé dans la section suivante.

5.2.3.2 Conception du circuit.

Le convertisseur à 1 bit (fig. 5.7) est constitué d'un comparateur de tension auquel s'ajoute en sortie un étage de *buffer*. Cette architecture permet d'obtenir une sortie binaire de données à partir du signal analogique fourni par l'oscillateur chaotique.

Pour répondre aux contraintes de faible consommation, de rapidité, nous pouvons lister trois types de topologies de comparateurs pouvant être utilisées, soient :

- un amplificateur en boucle ouverte, sans compensation interne.
- un système avec alimentation positive, qui utilise une structure une structure similaire à une *flip-flop* afin d'augmenter la vitesse de comparaison
- un système avec capacités commutées.

De ces trois topologies, seule la structure en boucle ouverte semble répondre à nos contraintes. En effet, les autres topologies requièrent d'un signal d'horloge plus rapide que celui de la référence que nous utilisons pour l'oscillateur chaotique. Ainsi, en terme de consommation mais aussi de complexité aussi bien sur la phase de conception que de modélisation des états métastables, ces deux dernières solutions ont été mises de coté au profit de la solution en boucle ouverte.

Ainsi, pour réaliser cette structure en boucle ouverte, nous avons utilisé un comparateur contenant un OTA du type ***rail-to-rail***, basé sur une architecture "symétrique" [5], comme le montre la figure 5.7. Cette architecture utilise deux amplificateurs différentiels (N et P) en parallèle. Ainsi, pour de faible tension d'entrée, l'amplificateur N est inactif alors que l'amplificateur P est opérationnel. A tension d'entrée élevée, c'est l'opération inverse qui se déroule.

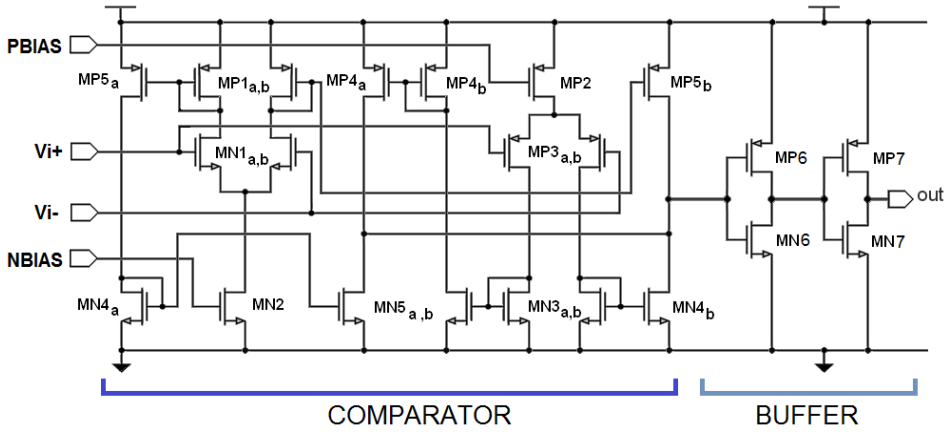

FIGURE 5.7 – OTA-Comparateur (a) schématique (b) symbole

La règle garantissant un fonctionnement en mode *rail-to-rail* est de disposer d'une tension d'alimentation suffisante pour avoir l'équation ci dessous vérifiée :

$$V_{DD} \geq V_{GS_p} + V_{GS_n} + 2V_{D_{sat}} \tag{5.5}$$

Où les tensions V_{GS_n} et V_{GS_p} correspondent respectivement aux tensions grille-sources des transistors de chaque paire différentielle d'entrée, $MN1[a,b]$ et $MP3[a,b]$. La tension $V_{D_{sat}}$ correspond à la tension minimale de drain dans les transistors qui opèrent comme sources de courant ($MN2$ et $MP2$).

Le seuil de référence est établi par une tension provenant du circuit de références programmable introduit dans le chapitre précédent. Dans un but d'optimisation de la consommation, l'amplificateur utilise une faible courant de polarisation. Il est donc nécessaire d'ajouter un étage d'adaptation en sortie du comparateur.

Les variations à la sortie de l'amplificateur en boucle ouverte sont finalement adaptées afin de convenir aux niveaux digitaux. Ceci a été réalisé à l'aide d'un *buffer* de sortie formé par deux inverseurs CMOS (constitués des transistors **MN6/MP6** et **MN7/MP7**). Il permet en même temps d'augmenter la capacité de sortie du circuit.

5.2.3.3 Dimensionnement des composants.

Le dimensionnement des composants du comparateur est délicat dans le sens ou les deux amplificateurs N et P doivent fonctionner de façon alternée. En effet, si un recouvrement de la zone de fonctionnement apparaît, cela permet d'ôter les problématiques de linéarité sur la zone centrale mais au profit d'une sur-consommation sensible. Par ailleurs, le fonctionnement de ce comparateur a été calibré pour fonctionner à basse consommation, d'où les tailles des transistors possédant des longueurs plus conséquentes que les largeurs. Ainsi, le tableau 5.1 résume les tailles ayant été retenues.

Transistors	Dimensions	Transistors	Dimensions
MP1(a :b)	$\frac{2\times2}{15}$	MN1(a :b)	$\frac{7\times1}{5}$
MP2	$\frac{4\times2}{15}$	MN2	$\frac{4\times1}{5}$
MP3(a :b)	$\frac{4}{0.5}$	MN3(a :b)	$\frac{2\times1}{5}$
MP4(a :b)	$\frac{2\times1}{5}$	MN4(a :b)	$\frac{2\times2}{15}$
MP5(a :b)	$\frac{2\times2}{15}$	MN5(a :b)	$\frac{2\times1}{5}$

TABLE 5.1 – Dimensionnement du comparateur

Les deux cellules inverseuses placées l'une derrière l'autre, assurent la fonctionnalité de *buffer*. Ils assurent ainsi la remise en forme du signal pour être exploité d'un point de vue digital pour la suite du circuit. La taille des composants est proposée dans le tableau 5.2

Cellules inverseuse 1		Cellule inverseuse 2	
MP1	MN1	MP2	MN2
$\frac{2\times0.45}{15}$	$\frac{0.5}{15}$	$\frac{2\times0.5}{5}$	$\frac{0.5}{5}$

TABLE 5.2 – Dimensionnement du buffer de sortie

5.2.3.4 Résultats.

Après avoir validé d'un point de vue simulation le fonctionnement du comparateur, celui ci a été réalisé sur silicium. L'importance de la symétrisation de la structure

est importance via la prise en compte de notion de *matching*. Ceci a été bien pris en compte via la taille des transistors retenue, comme le montre la figure 5.8

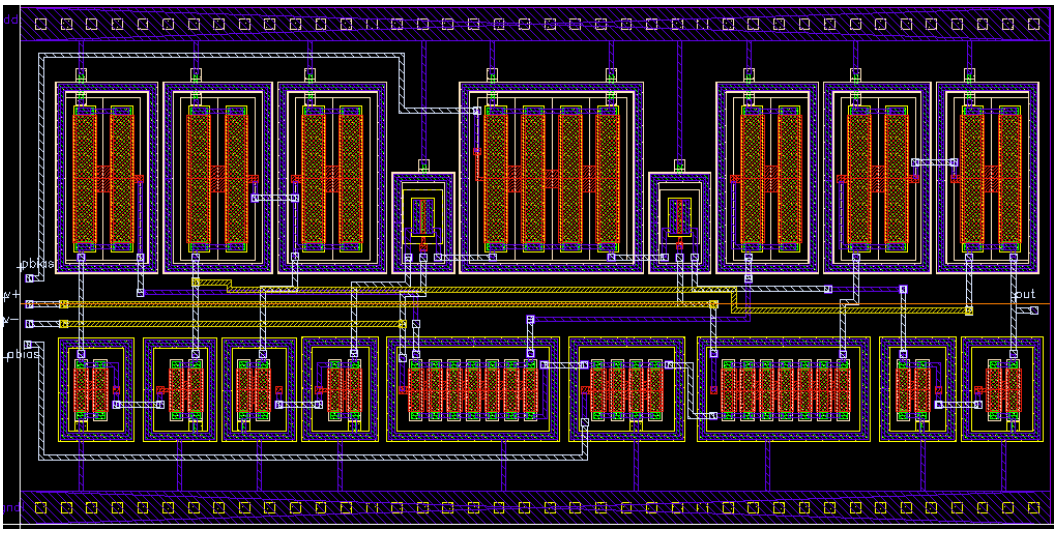

FIGURE 5.8 – Layout de la cellule OTA.

La figure 5.9 indique la réponse, en fréquence en magnitude et en phase de l'OTA en boucle ouverte sous différentes valeurs de charge. Nous allons travailler sur une faible capacité de charge avant de connecter vers le *buffer*, donc la plage de 200fF correspond correctement à nos besoins (même si dans le système lae marge de phase n'apporte aucun intérêt lorsqu'on est en boucle ouverte).

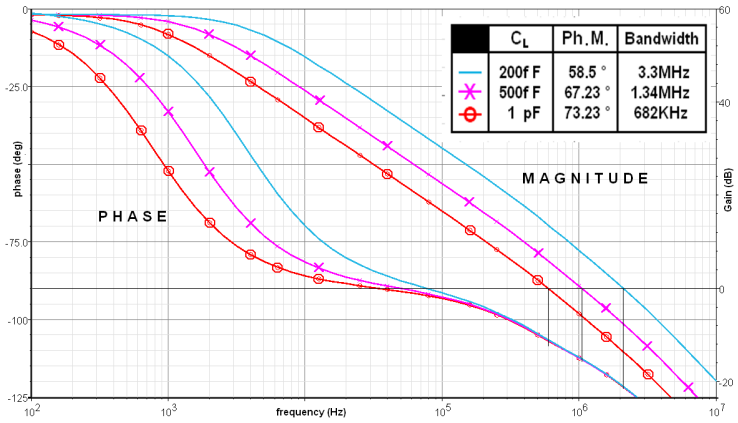

FIGURE 5.9 – Réponse en fréquence de l'OTA.

La figure 5.10 indique les formes d'onde à la sortie du circuit et du *buffer*, sous un signal d'entrée sinusoïdale d'amplitude 3V et fréquence 200kHz, et une capacité de charge de 500fF. La figure indique un déphasage du signal en sortie du comparateur ainsi qu'une dégradation du signal attendue (on commence à sortir de la zone de gain constant), mais avec le buffer de sortie nous arrivons à récupérer le signal carré

souhaité en phase. En termes de consommation, le système requiert une moyenne de 600nA, et en repos 440nA, avec une surtension de commutation de 1μA.

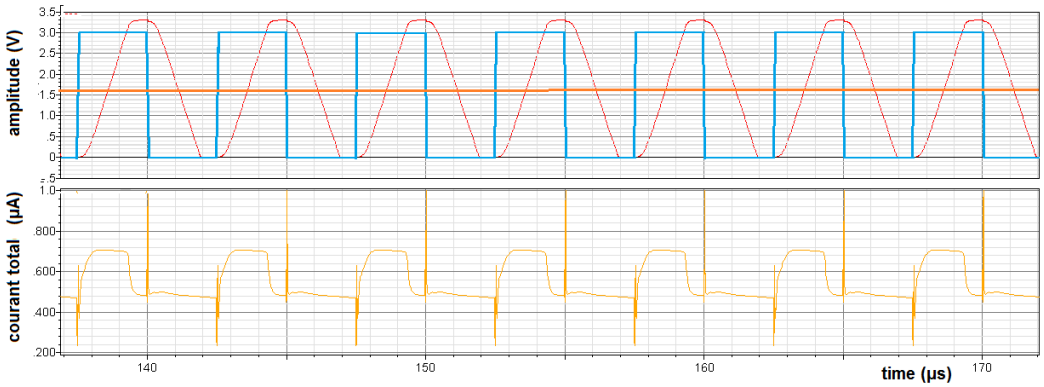

FIGURE 5.10 – Réponse transitoire du comparateur.

5.2.4 Correcteur d'entropie.

Comme décrit précédemment, un générateur vraiment aléatoire doit posséder une distribution uniforme.

Dans le cas présent, nous obtenons des donnés binaires directement de la sortie chaotique, en appliquant un comparateur de tension. Lors de cette conversion de l'analogique chaotique vers des transitions binaires, le signal obtenue échantillonné (synchronisé), peut avoir une distribution biaisé (non uniforme) des '0's et '1's logiques.
En effet, cette disparité est liée à la dynamique de sortie du montage ainsi qu'à la tension de seuil appliquée au comparateur. Afin de rééquilibrer cette disparité, un correcteur d'entropie a été ajouté à la sortie du comparateur 1 bit.

Les causes qui peuvent générer ces distributions non uniformes sont :
- L'entropie de la source n'est pas assez grande.
- L'extraction d'aléa réalisée initiale n'est pas optimale.
- Les valeurs obtenues lors de l'extraction sont corrélés.

Le rôle du bloc de post-traitement est d'améliorer les propriétés statistiques de la séquence binaire. D telles améliorations impliquent une réduction sur le débit de sortie.
Parmi les correcteurs numériques nous pouvons citer les suivants :

Correcteur XOR.

C'est le correcteur le plus simple. Il consiste à appliquer une fonction *ou exclusif* sur des blocs de taille fixe avant les envoyer à la sortie. L'avantage de cette technique est la quantité réduite de composants nécessaires, par contre, le débit de sortie est

réduit proportionnellement à la taille du groupe de bits du correcteur utilisés pour générer un bit debiasé.

Correcteur de Von Neumann.

Ce type de correcteur utilise une fonction non linéaire qui réalise une discrimination pour réduire les séries de valeurs répétitives et consécutives.
Une telle discrimination permet idéalement d'enlever le *biasing* et obtenir une distribution uniforme. L'inconvénient de cette solution c'est la réduction du débit de sortie ainsi que la faiblesse contre des signaux avec des traces de périodicité (des groupes de séquences aléatoires qui se répètent, comme dans le cas des PRNG-LFSR).

Correcteur du type LFSR.

Le *LFSR*[1] est un correcteur basé sur une fonction linéaire. Ce type de circuit est souvent utilisé dans la création de PRNGs, du au fait que :

- L'implémentation matérielle d'un LFSR est relativement simple
- les séquences de sortie possèdent de bonnes propriétés statistiques
- Leur comportement peut être modélisé mathématiquement et algorithmiquement.

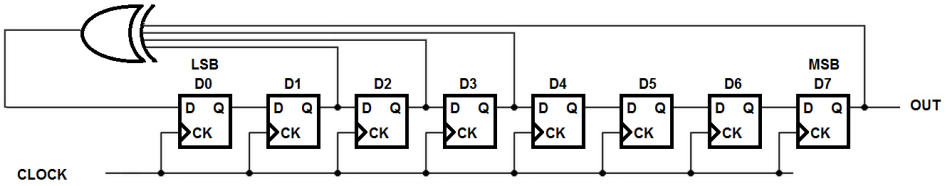

FIGURE 5.11 – Correcteur du type LFSR avec *taps* externes

Un *LFSR* de taille L consiste en L registres en cascade de 1 bit, synchronisés par un même signal d'horloge (fig. 5.12). Cette association permet de stocker les L-1 dernières données à la sortie de chaque registre au bout de chaque coup d'horloge.
Les opérations que véhiculent les registres cascadés choisis sur un polynôme générateur (*taps*) correspondent à la somme (modulo 2)[2].
Le polynôme générateur du LFSR possède la forme :

$$f_{(X)} = a_1 X_1 \oplus a_2 X_2 \oplus \cdots \oplus a_{L-1} X_{L-1} \tag{5.6}$$

De forme qu'à chaque coup d'horloge ($a_L = [0, 1]$),

$$X_1 = f_{(x)} \tag{5.7}$$
$$X_2 = X_1 \ldots \tag{5.8}$$
$$X_{L-1} = X_{L-2} \tag{5.9}$$

1. Linear Feedback Shift Register
2. opérateur ou exclusif

Par exemple, pour un LFSR à 8 bits (figure 5.12.a), le polynôme générateur de ce LFSR associé, d'ordre 7 est :

$$X_7 \oplus X_3 \oplus X_2 \oplus X_1 = f_{(X)} \tag{5.10}$$

Cette architecture, associée aux solutions proposées par le polynôme, produit un cycle de 128 valeurs pseudo aléatoires. Ce patron se répète cycliquement, comme le montre la figure 5.13.(a).

L'intérêt de ce type de système réalimenté est d'assurer la possibilité de disposer d'une distribution binaire équiprobable ou près de l'équiprobable.

Afin de profiter de la caractéristique de la distribution uniforme d'un LFSR, nous allons exécuter une opération additionnelle, avec un bit provenant d'une source aléatoire biaisée (*random seed*).

Dans ce cas, la séquence de sortie du LFSR devient non prédictive et non reproductible, avec une distribution binaire uniforme (figure 5.12).

5.2.5 Choix retenu.

Afin de rester sur un système simple, nous nous sommes orientés sur l'intégration d'un montage de type compression linéaire, autour d'un *LFSR* comme correcteur d'entropie et générateur de valeurs aléatoires.

Ainsi, la première étape a été de quantifier la taille de l'équation caractéristique du LFSR et donc du nombre de registres nécessaires à son implémentation. Pour cela, un jeu de simulation comportementales réalisés à partir des données récupérées des simulations (BSIM) de l'oscillateur chaotique, suivi du comparateur nous a permis de valider l'effet du correcteur ainsi que son architecture.

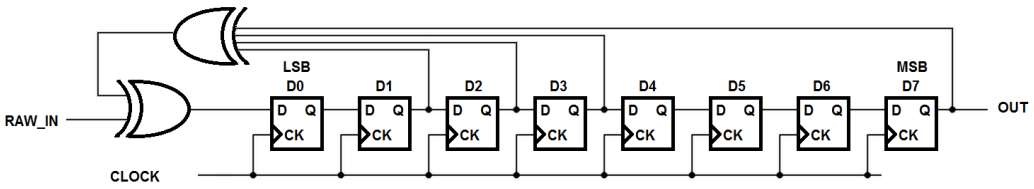

FIGURE 5.12 – Correcteur du type LFSR avec *taps* externes.

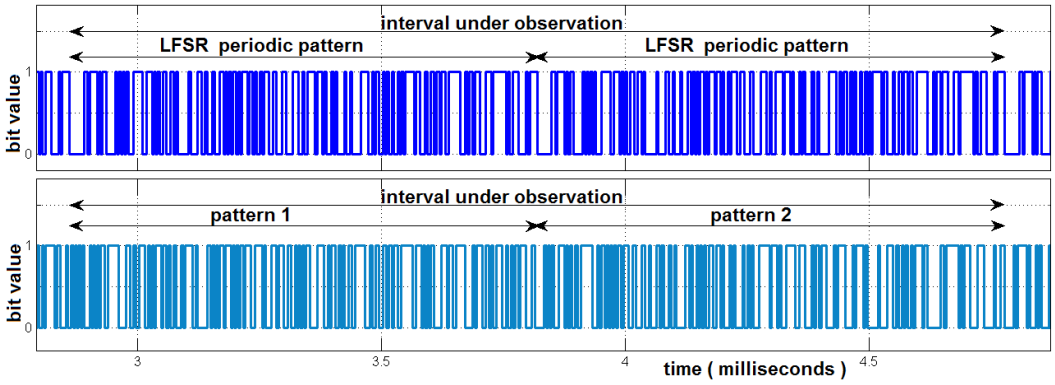

FIGURE 5.13 – LFSR
(a)independant-PRNG (b)avec injection du signal biaisé.

Une représentation statistique des résultats est illustrée en figure 5.14, montrant de façon complémentaire que l'effet d'injecter un signal biaisé aléatoire sur le LFSR n'affecte pas la distribution uniforme caractéristique d'un LFSR, mais ne fait qu'annuler l'effet cyclique des patrons aléatoires, inhérents d'un LFSR.

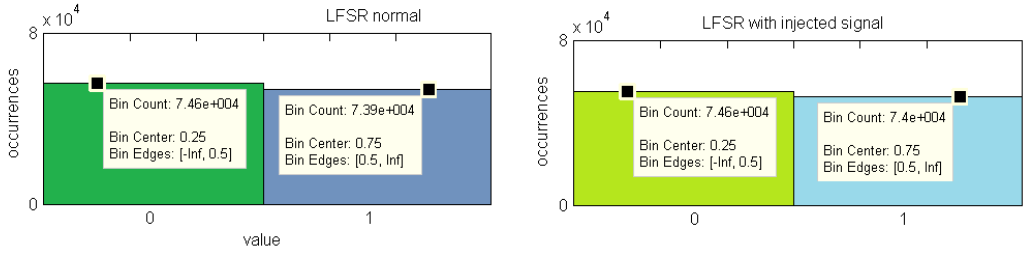

FIGURE 5.14 – Distribution binaire (a)LFSR solo (b) avec injection du signal biaisé.

Finalement, une topologie avec un LFSR de 8 bits avec la fonction affecté sur les taps **[8 6 5 4 1]** sera utilisée.

5.3 Réalisation sur silicium.

Une fois l'architecture de l'oscillateur chaotique validée, comme il a été montré au chapitre 4, auquel s'est ajoutée la validation de la structure convertisseur 1 bit (section 5.2.3), corrigée par le correcteur d'entropie (section 5.2.4), l'intégration de l'ensemble de ces blocs a été possible via la réalisation sur silicium en technologie AMS 0.35μm.

Ainsi, à l'aide des outils de Cadence, via les chaines Virtuoso et Layout XL, une architecture complète a été réalisée comme le montre la figure 5.15.

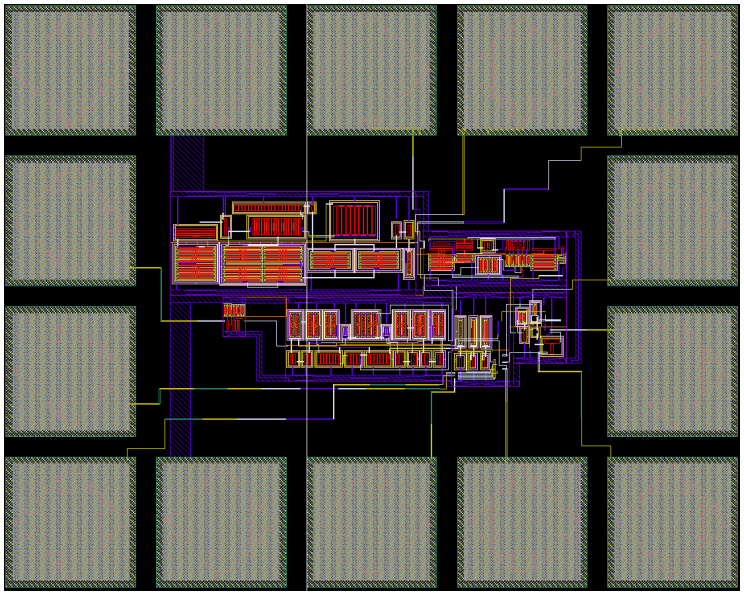

FIGURE 5.15 – Layout du subsystème.

La figure 5.16 illustre le circuit obtenu ayant une taille de $535\mu m \times 425\mu m$, incluant les plots.

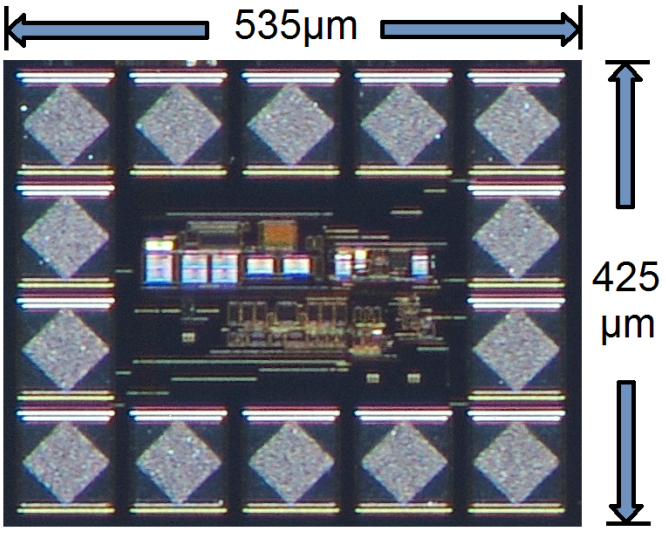

FIGURE 5.16 – Photographie du test chip réalisé.

Configuration Physique des broches.

La figure indique la distribution de pins sur le circuit integré fabriqué, mis sour boitier du type DIL14.

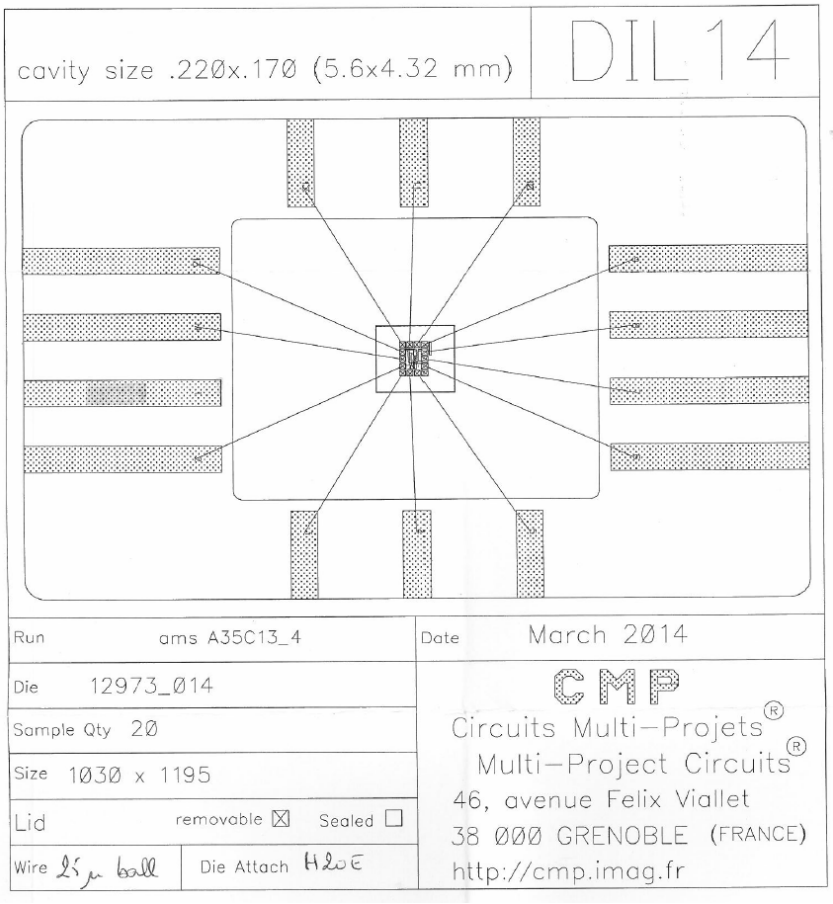

FIGURE 5.17 – Distribution des broches sur le chiptest

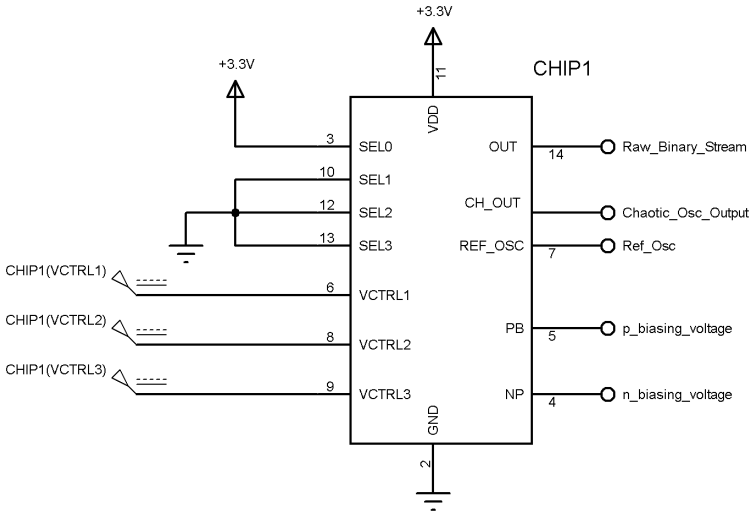

FIGURE 5.18 – Configuration de test sur le chiptest

5.4 Influence des conditions de tests.

De la même façon que sur l'expérience de Lorenz ([25]), nous avons constaté que le changement des paramètres de précision du calculateur entrainait des résultats ayant des caractéristiques différentes pendant la phase de simulation.

Notamment, l'ajout d'un délai avant le démarrage du système ou le changement du pas de simulation (paramètre *tstrobe*), ceci dans le but d'accélérer le temps de calcul pour avoir une quantité plus élevé de données génère des résultats de BSIM sur le signal chaotique différents à chaque changement de conditions.

Dans la partie de simulations pré-post Layout, l'une des caractéristiques du simulateur spectre est d'assigner par défaut des pas d'échantillonnage dynamique, en fonction de la convergence sur le signal de sortie. Cette particularité génère une quantité de points sur les transitions brusques, présents sur l'oscillateur chaotique qui fait des transitions plus des surtensions de commutation.

Afin d'avoir des pas réguliers sur les transitions plus stables du signal, nous sommes obligés de "normaliser" les données, permettant par la même occasion de réduire l'espace mémoire dans le logiciel d'analyse que nous utilisons (MatLab). Ceci est réalisé via une synchronisation du signal avec les transitions du signal d'horloge de référence.

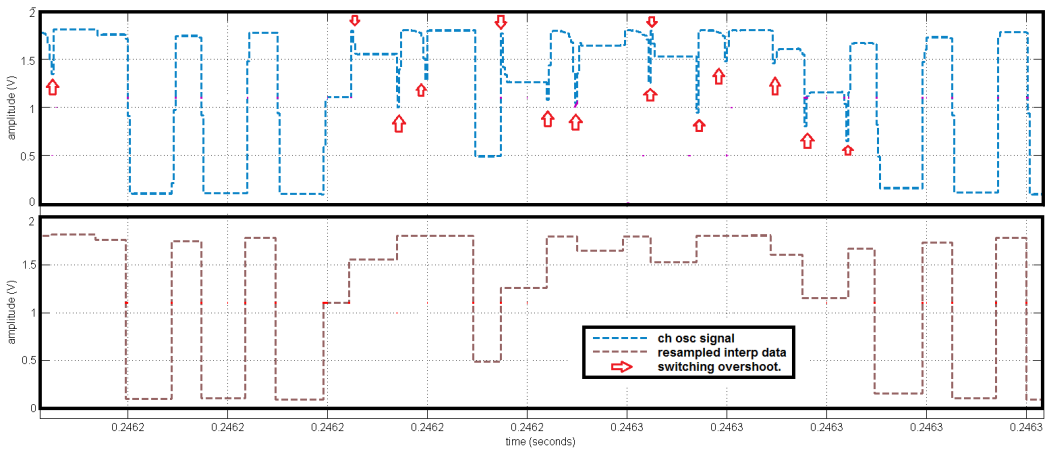

FIGURE 5.19 – Discrimination des surtensions du à la commutation du système lors de l'interpolation et re-échantillonnage du signal acquise.

Nous réalisons ensuite une interpolation-reéchantillonnage de données avec un pas constant, afin de normaliser et discrétiser sur des transitions . Nous perdons alors les transitions de commutation de l'oscillateur chaotique (figure 5.19), mais nous gagnons de la vitesse de calcul, espace mémoire et une réussite du 100 % pour la détection de changements de niveau sur la sortie du convertisseur à 1 bit (figure

5.20).

La figure 5.20 illustre le résultat de la normalisation de données avant et après avoir appliqué l'interpolation et raffiner le méthode de calcul. Les deux chronogrammes indiquent le signal de sortie de simulation avec un pas non uniforme ni aucune re-synchronisation comparé avec la sortie de l'oscillateur chaotique soumis au modèle du comparateur. Ainsi ,l'un prend en compte la sortie sans aucun traitement et un comparateur idéal ; tandis que l'autre prend la sortie normalisée et un comparateur avec un seuil pour comparer la validité de l'approche de la modélisation. Nous pouvons observer des bits qui sont perdus avec un comparateur idéal, alors qu'avec les signaux resynchronisés, les résultats du modèle et de la sortie normalisée sont les mêmes.

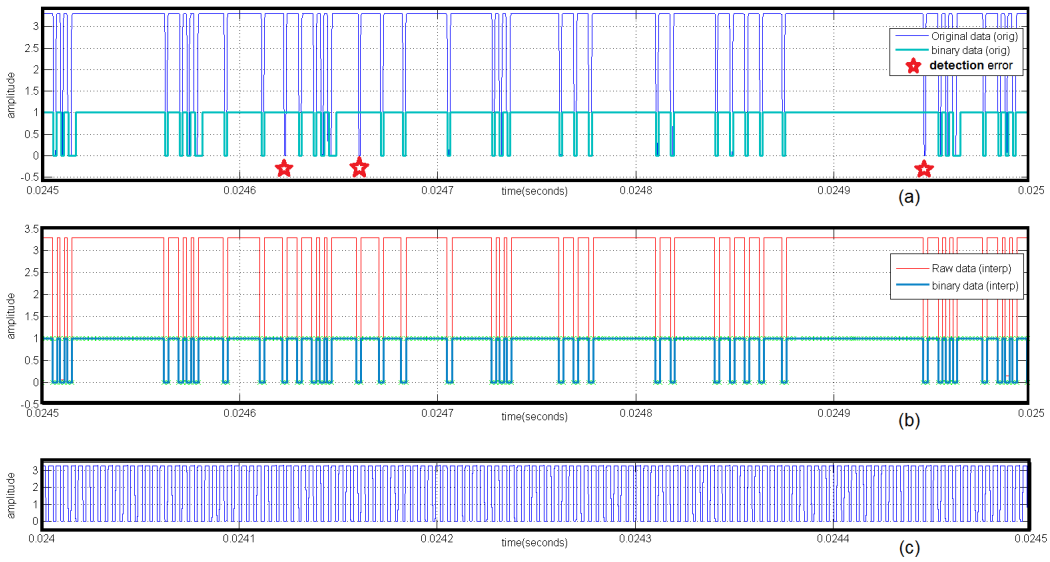

FIGURE 5.20 – Erreur de détection de données et correction par interpolation.

Chaque changement d'état correspond à une demi-période du signal d'horloge CLK, ce qui correspond a un débit de donnés brut de 500Kbps (approx), sur les conditions de test post Layout sur spectre.

Cependant l'implantation matérielle ajoute des capacitances parasites sur certains plots, notamment sur ceux qui etaient censés d'être une partie interne de la structure du systeme (sortie chaotique, sortie d'horloge de référence, biasing interne de base du circuit). Leurs caractéristiques se sont dégradées avec la présence des impédances (notamment du coté capacitif) des plots, en réduisant le débit de sortie.

La configuration des pins pour le test s'observe dans la figure 5.18. Cette dégradation a fait varier les points de biasing du circuit (n_{bias}, p_{bias}) et a eu une influence sur les points d'opération de la polarisation du système non linéaire par defaut. Deux options de récalibration possibles ont été choisis : un changement des tensions de

polarisation du circuit non linéaire, soit une variation sur la tension d'alimentation. Parmi les deux, la variation sur la tension d'alimentation de 3V à 2.7V a été la plus efficace, répétitive et avec moins d'influence par rapport à la charge des plots du test-chip parmi les chips testés du batch fabriqué. En revanche , la fréquence de sortie à été affectée (i.e. la fréquence de l'oscillateur changée de 250Khz à 140Khz) ainsi que les effets capacitifs en sortie sont plus observables.

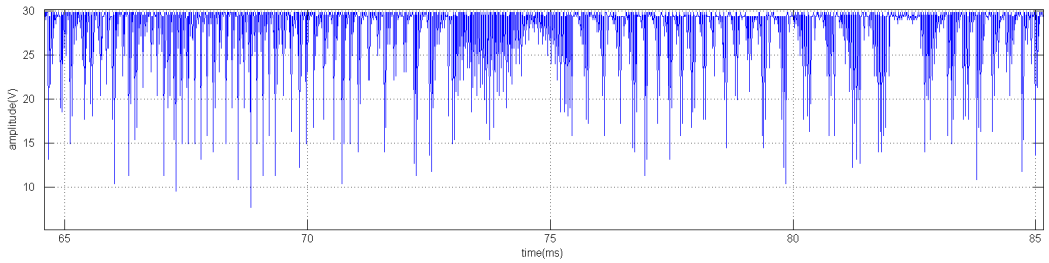

FIGURE 5.21 – Sortie Chaotique apres récalibration.

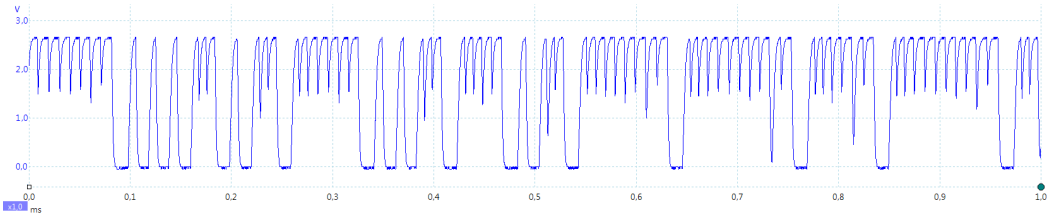

FIGURE 5.22 – détail de la sortie binaire sur le testchip.

Comme conditionnement sur les mesures de la sortie binaire aléatoire,(figures 5.21 et 5.22) nous avons utilisé une porte logique standard CMOS, afin de passer sur la plage exploitable pour la mesure avec le banc de test de la carte MyRio.

5.5 Conclusion.

Après avoir présenté l'oscillateur chaotique, bloc central du générateur d'aléa au sein du chapitre 4, celui-ci a été intégré au sein du TRNG afin de pouvoir en extraire les résultats exploitables pour le passage des tests d'aléa que nous étudierons dans le chapitre suivant.

Ainsi, après avoir explicité la structure du TRNG avec l'introduction de blocs de conversion entre une information analogique fournie par l'oscillateur chaotique et l'aspect de traitement numérique des informations, une présentation des résultats obtenus a été présentée que ce soit au niveau simulation pré, post Layout ou sur Silicium.

Malgré les effets des plots d'implantation sur certaines parties sensibles du circuit, le comportement aléatoire est maintenu, ainsi que le comportement chaotique du système non linéaire, avec une dégradation sur le débit du signal (qui passe à 70Kbps)

L'influence des variables sur l'oscillateur invalident les mesures de courant du circuit, étant donnée que pour une fréquence inférieure, les surtensions de commutation seront considerablement inférieures à celles obtenues sur le test d'extraction post-Layout sur Cadence.
Enfin, une présentation des résultats et l'impact de certains paramètres sur la structure a été mise en avant afin de justifier les conditions de test et les résultats associés. Partant de ces résultats, le passage aux différents tests d'aléa va être proposé dans le chapitre suivant afin de quantifier le caractère aléatoire du signal obtenu.

Chapitre 6

Évaluation de la qualité des RNGs

6.1 Introduction.

Prouver mathématiquement qu'un *Random Number Generator - RNG* est aléatoire est généralement difficile. En effet, malgré le fait d'avoir une source d'aléa, l'échantillonnage et la conversion vers des données binaires peuvent provoquer un biais sur la distribution souhaitée et rendre difficile d'exploitation des résultats. Toutefois, avec l'intervention d'un correcteur, nous pouvons arriver à compenser cela, mais ceci en réduisant le débit des valeurs.

Dans la pratique, le degré d'aléa se vérifie via l'utilisation de tests statistiques. En fonction de ces résultats, nous pourrions affirmer avec un degré de confiance que le générateur fournit des valeurs aléatoires non prédictives.

Dans ce chapitre, nous allons nous attacher à l'exploitation des résultats provenant des signaux sortants du comparateur qui sont ensuite traités par un correcteur, en utilisant des méthodes d'évaluation de robustesse des séquences aléatoires standardisées. Ainsi, ce chapitre débutera par une présentation générale des tests nécessaires à la vérification de l'aléatoirité des signaux pour ensuite se focaliser sur une présentation des tests NIST et enfin finir sur les tests en tant que tels développés.

6.2 Vérification de l'aléatoire sur les RNG.

Par définition, un test fournit un moyen de réaliser des décisions quantitatives, basées sur une conjecture de départ, laquelle peut être validée ou pas. Pour cela, nous allons appeler cette hypothèse de départ la condition indexée 0 (ou hypothèse H_0). Dans notre contexte des générateurs aléatoires, l'hypothèse au départ H_0 sera définie comme étant *"La suite analysée n'est pas aléatoire"*.

Une fois l'hypothèse initiale définie, nous allons maintenant quantifier les résultats des tests en utilisant comme indicateur les valeurs de probabilité aléatoire (*valeur-p*).

L'idée de l'analyse quantitative utilisée implique d'assurer une hypothèse initiale, qui ne peut jamais être acceptée directement, mais qui peut être rejetée seulement par le test statistique. La valeur p est considérée comme la mesure à partir de laquelle les données plaident contre l'hypothèse initiale. Les seuils suivants sont généralement pris en fonction d'un niveau de confiance, exprimé comme (1-α). Ainsi pour :

- $p < \alpha /10$: très forte présomption contre l'hypothèse,
- $\alpha /10 < p \leq \alpha /2$: forte présomption contre l'hypothèse,
- $\alpha /2 < p \leq \alpha$: faible présomption contre l'hypothèse,
- $p \geq \alpha$: pas de présomption contre l'hypothèse.

Si nous revenons à notre étude, l'hypothèse initiale indique que la séquence binaire à évaluer est aléatoire, et nous allons prendre un niveau de confiance du test à 99% (α=0.01), ce qu'implique que nous pouvons tolérer de rejeter une séquence sur 100 dans l'évaluation effectuée.

Un nombre de tests standardisés existe, basés sur l'analyse de certaines propriétés statistiques, telles que la distribution uniforme de valeurs et de non répétabilité, permettant de valoriser le caractère non prédictif du générateur sous étude. Nous pouvons donc citer à titre d'exemple :

1. **Les Tests de Knuth.**Créés par Donald Knuth [21], gouru des valeurs aléatoires au sein d'une époque où l'aléatoire n'avait pas l'importance actuelle pour les applications sécuritaires. Ces tests n'ayant pas été crées pour ces applications, ils n'offrent pas un niveau de confiance suffisant pour les applications cryptographiques.
2. **Les Tests de Diehard.** Crées par George Marsaglia [28], dans le but d'offrir des tests plus robustes que ceux de Knuth. Toutefois après la retraite de l'auteur, la suite de tests développée a été peu révisée et utilisée.
3. **Les Tests Crypt-X.** Crées par le centre de recherche en sécurité de l'information de l'Université de technologies de Queensland, et offerts comme une application commerciale pour la vérification sur systèmes de chiffrement de données et générateurs de clés.
4. **Les Tests ENT.** Développés par John Walker [1], et appliqués pour l'évaluation des générateurs pseudoaléatoires. Ces outils sont de type statistique disposant d'algorithmes de compression pour des applications à grande quantité d'informations sous observation.
5. **Les Tests NIST.** Développés par l'Institut National de Standards et Technologie (NIST), ils sont une succession de tests ciblés sur des applications cryptographiques. Ils sont basés sur une synthèse d'algorithmes existants auxquels s'ajoutent d'autres tests basés sur de nouveaux algorithmes créés par l'équipe de sécurité informatique et d'ingénierie statistique du NIST. Cette suite de tests est devenue comme un standard bien reconnu dans le monde des tests de l'aléatoire et du cryptographique.

Partant de ces constatations, nous avons donc mis en place les tests proposés par le NIST, basés sur le fait de leurs applications de type cryptographiques.

1. http ://www.fourmilab.ch/random/

6.3 La suite NIST.

Afin de répondre aux spécifications attendues, le NIST propose une batterie de tests de validation : FIPS 140-1 et FIPS 140-2. En effet, historiquement, ces deux successions de tests ont été développées comme outils de validation pour les systèmes de cryptage de type AES. Afin de prouver la puissance de ces tests, un DES a déjà été entièrement décrypté. Ainsi, nous allons dans cette partie mentionner de manière synthétique les paramètres de qualification et les tests réalisés, établis au sein d'un document détaillé [40].

Le NIST a développé des tests standardisés ayant comme but l'amélioration sur la protection de la communication de données numériques, de valeurs **binaires**. Ces tests sont groupés sur deux collections : les FIPS 140-1 et FIPS 140-2, appliquées pour l'évaluation des RNG, ayant comme objectif des données de grande taille.

6.3.1 Méthodologie utilisée.

La suite statistique NIST fait appel à 15 tests statistiques qui doivent être réalisés en suivant une démarche, qui consiste à trouver la variation $\chi^2_{(x)}$ d'un paramètre d'intérêt calculé à partir de la séquence binaire sous étude.
La suite contient 15 tests :

1. Le test de fréquence globale (monobit)
2. Le test de fréquence par blocs
3. Le test de répétition globale
4. Le test de répétition par blocs
5. Le test de rang de la matrice binaire
6. Le test spectral
7. Le test de non chevauchement de séquences
8. Le test de chevauchement de séquences
9. Le test universel de Maurer
10. Le test de la complexité linéaire
11. Le test Série
12. Le test de l'entropie approximée
13. Les sommes cumulatives
14. L'excursion aléatoire
15. Le variant sur l'excursion aléatoire.

Nous allons maintenant étudier le principe des fonctions statistiques en mettant en avant les deux méthodes possibles de validation du caractère aléatoire du signal.

6.3.2 Fonctions statistiques utilisées.

Les deux méthodes de validation du caractère aléatoire du signal, reposent sur la quantification des fonctions d'erreur $erf(x)$ et sa complémentaire $erfc(x)$ mais également sur l'analyse du paramètre χ^2. Ainsi,

- Lorsque l'analyse de la totalité de la séquence est sous étude, la densité de probabilité d'une suite de valeurs observées est caractérisée par une loi normale définie par :

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma}} \tag{6.1}$$

dont σ et μ, correspondent à la variance et moyenne, respectivement. Une distribution normale standard possède $\sigma = 1$ et $\mu = 0$. Ainsi, la fonction de distribution cumulative (cdf) d'une distribution normale, prend la forme :

$$\phi(x) = \frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^{x} \mathrm{e}^{-\frac{-t^2}{2}} \,\mathrm{d}t \tag{6.2}$$

La fonction d'erreur, *erf(x)*, indique la probabilité d'avoir une variable aléatoire avec une distribution normale, de moyenne nulle et variance $1/2$.

$$erf(x) = \frac{1}{\sqrt{\pi}} \int_{-x}^{x} \mathrm{e}^{-t^2} \,\mathrm{d}t \tag{6.3}$$

La fonction complémentaire est :

$$erfc(x) = \frac{2}{\sqrt{\pi}} \int_{-x}^{\infty} \mathrm{e}^{-t^2} \,\mathrm{d}t \tag{6.4}$$

Les deux fonctions, ϕ et erf peuvent alors être approximées en utilisant la relation :

$$\phi(x) = \frac{1}{2}[1 + erf(\frac{x}{\sqrt{2}})] \tag{6.5}$$

- La seconde méthode consiste en l'analyse du terme χ^2 en utilisant plusieurs degrés de liberté et en décomposant la séquence d'entrée en blocs. Si nous avons comme hypothèse que les données sont aléatoires, un sous bloc de données sera aussi aléatoire. Ainsi, l'analyse permet d'identifier la présence de patrons ou combinaisons particulières identifiables comme non aléatoires.

La fonction Γ est une extension de la fonction factoriel ($n!$), étendue sur des valeurs réelles positives :

$$\Gamma(z) = \int_{0}^{\infty} t^{z-1}\mathrm{e}^{-t} \,\mathrm{d}t = (z-1)! \tag{6.6}$$

Nous pouvons exprimer la fonction gamma en terme de deux fonctions complémentaires :

$$\Gamma(z) = \gamma(z,x) + \Gamma(z,x) \tag{6.7}$$

avec $\Gamma(z,x)$ la fonction gamma incomplète supérieure et $\gamma(a,x)$ l'inférieure.

$$\Gamma(z,x) = \int_x^\infty t^{z-1}\mathrm{e}^{-t}\,\mathrm{d}t \tag{6.8}$$

Nous allons utiliser deux expressions normalisés : la fonction gamma inférieure incomplète normalisée, $P_{(z,x)}$:

$$P(z,x) = \frac{\gamma(z,x)}{\Gamma(z,x)} \tag{6.9}$$

et la fonction gamma supérieur incomplète normalisée, $Q_{(z,x)}$:

$$Q(z,x) = 1 - P(z,x) = \frac{\Gamma(z,x)}{\Gamma(z)} = \frac{1}{\Gamma(z)}\int_0^\infty t^{z-1}\mathrm{e}^{z-1}\,\mathrm{d}t \tag{6.10}$$

6.3.3 Obtention des *valeurs-p.*

L'obtention des *valeurs-p* dépend du type de test, dans lequel s'observe une caractéristique en particulier de la distribution de données de la séquence. Nous allons donc dans la partie suivante étudier les différents tests et l'extraction de la *valeurs-p.*

6.4 Description des tests.

Nous allons maintenant étudier chaque test avec l'expression qui lui sera propre de la *valeurs-p.* Pour cela, nous nous mettons dans le cas où nous avons une séquence ε_i à analyser, possédant n éléments, sous laquelle nous allons initialement considérer l'hypothèse de l'aléatoire.

6.4.1 Test de fréquence globale.

L'objectif de ce test est de vérifier si la proportion de '1' et '0' dans la séquence de valeurs est égale ou proche de l'égalité. Le biais doit rester dans des limites acceptables. Nous allons alors pondérer les éléments de la séquence avec $X_i = 2\varepsilon_i - 1$.

La somme de cette nouvelle séquence, $\sum_{i=1}^{n} X_i$ représente le biais de la séquence.

L'analyse utilise le Théorème de Moivre-Laplace, indiquant que la distribution de la somme binomiale normalisée par $\sqrt{n}$ s'approche d'une distribution normale.
Le $valeur - p$ à calculer prend alors la forme suivante :

$$valeur - p = erfc(\frac{\sum_{i=1}^{n}(2\varepsilon_i - 1)}{\sqrt{2n}}) \tag{6.11}$$

6.4.2 Test de fréquence locale.

Ce test permet d'étudier le biais à l'intérieur de la séquence, en analysant N blocs individuels successifs de M bits (MN<n). La distribution de valeurs dans les blocs doit aussi respecter la distribution équiprobable, avec un marge d'erreur tolérable. Le biais local est

$$\pi_i|_1^N = \frac{\sum_{j=1}^{M}(\varepsilon_{(i-1)M+j})}{M} \tag{6.12}$$

χ^2 est défini par :

$$\chi^2 = 4M\sum_{j=1}^{N}(\pi_i - \frac{1}{2})^2 \tag{6.13}$$

Le $valeur - p$ à calculer est :

$$valeur - p = igamc(\frac{N}{2}, \frac{\chi^2}{2}) \tag{6.14}$$

6.4.3 Le test de répétition ("Long Run Test").

Un "run" est une séquence de k bits '1' ou '0' consécutifs. L'apparition d'un "run" de taille plus grande représente un changement d'état (monotonie) trop lent, indicatif d'une dépendance sur la valeur antérieure. L'analyse regarde si les occurrences des "runs" de différentes tailles obéissent à une caractéristique dans les limites de l'aléatoire.
Il existe une condition à satisfaire avant lancer ce test :

$$|\Pi - \frac{1}{2}| \geq \tau \tag{6.15}$$

dont $\Pi_i = \frac{\sum_{i=1}^{n}}{n}\varepsilon_i$ et $\tau = 2\sqrt{n}$
Le total de séquences de '1's et '0's se calcule en utilisant la formule

$$V_n(obs) = (\sum_{k=1}^{n} r_{(k)} + 1) \tag{6.16}$$

Où $r_{(k)} = 0$ si $\varepsilon_k = \varepsilon_{K+1}$, et $r_{(k)} = 1$ si $\varepsilon_k \neq \varepsilon_{K+1}$
Le valeur-p à calculer est :

$$valeur - p = erfc(\frac{|V_n(obs) - 2n\Pi(1-\Pi)|}{2\sqrt{2n}\Pi(1-\Pi)}) \tag{6.17}$$

6.4.4 Le test de répétition locale.

L'objectif de ce test est d'analyser la distribution des runs dans chacun des N blocs de M bits dans lequel la séquence originale est découpée.

La démarche de calcul consiste à trouver les fréquences V_i correspondants aux '1' consécutifs d'une taille i définie. ($V_2 \rightarrow$ "11" ,$V_4 \rightarrow$ "1111", etc)

Un tableau de probabilités π_i sont pré calculés pour différentes valeurs de M minimales ainsi qu'une valeur de classe K.
Le valeur χ^2 est donné par l'expression :

$$\chi^2 = \sum_{i=0}^{K} \frac{(V_i - N\pi_i)^2}{N\pi} \tag{6.18}$$

Le valeur-p à calculer est :

$$valeur - p = igamc(\frac{K}{2}, \frac{\chi^2}{2}) \tag{6.19}$$

6.4.5 Le test de rang de la matrice binaire.

L'objectif de ce test est d'estimer si la séquence de bits possède des patrons répétitifs à l'intérieur d'elle même.

Pour cela, nous allons analyser N blocs de taille fixe et évaluer s'il existe une dépendance linéaire entre chaque sous-blocs. L'analyse s'effectue en prenant de façon matricielle, en convertissant les sous-blocs vers des matrices de M x Q (usuellement on utilise M=Q=32).

Nous calculons le rang des sous-matrices. Si le déterminant d'une submatrice est différent de zéro, la matrice possède un rang de valeur M. Dans le cas contraire, le calcul se répète successivement pour toutes les sous-matrices d'ordre (M-k).

Théoriquement, il est suffisant d'analyser jusqu'au rang M-2. La probabilité pour une matrice d'avoir un rang M-k est négligeable (inférieure à 0.005) pour les ordres inférieurs à M-2 :

$$P_M = 0.2888 \tag{6.20}$$
$$P_{M-1} = 0.577 \tag{6.21}$$
$$P_{M-2} = 0.1284 \tag{6.22}$$

Pour que la somme de probabilités soit égale à 1, nous allons utiliser $P_{M-2} = 0.1336$.

Avec F_u, le paramètre d'analyse qui permet de réaliser l'estimation χ^2.
Soit :

- F_M le nombre de sub matrices de rang M
- F_{M-1} le nombre de sub matrices de rang M-1 parmi les N-F_M sub matrices
- F_{M-2}=N-F_M-F_{M-1} le nombre de matrices de rang inférieur ou égal à M-2.

La valeur χ^2 est donnée par l'expression

$$\chi^2 = \sum_{i=M-2}^{M} \frac{(F_i - NP_i)^2}{NP_i} \tag{6.23}$$

La valeur-p à calculer, est un indicatif de la déviation de la distribution par rapport au rang précalculé est :

$$valeur - p = igamc(\frac{K}{2}, \frac{\chi^2}{2}) \tag{6.24}$$

6.4.6 Test spectral.

Cette méthode basée sur la transformé discrète de Fourier (DFT) permet de détecter des caractéristiques périodiques dans la séquence sous étude (donc des patrons répétitifs).

Nous effectuons une pondération du type $x_k = 2\varepsilon - 1$ sur tous les éléments de la séquence.

Soit la composante de la DFT du i-éme bit :

$$F_{(u)} = \sum_{i=k-1}^{n} x_k e^{\frac{2\pi_i (k-1)u}{n} j} \tag{6.25}$$

Etant donné le caractère symétrique de la DFT, nous allons prendre que la partie positive du spectre (0 < i < n/2 -1).
Nous allons calculer un seuil maximal $T = \sqrt{n.log(\frac{1}{0.05})}$ (avec un niveau de confiance de 95%). Théoriquement, le nombre de valeurs inférieur au seuil T, dans une séquence aléatoire (N_0) vaut : $N_0 = 0.95n/2$

Soit N_1 le nombre réel de valeurs maximales inférieures au seuil T. La variable sous observation se calcule par la relation :

$$d = \frac{N_1 - N_0}{\sqrt{\frac{(n)(0.95)(0.05)}{4}}} \tag{6.26}$$

La valeur-p à calculer est :

$$valeur - p = erfc(\frac{|d|}{\sqrt{2}}) \tag{6.27}$$

6.4.7 Test de non chevauchement de séquences.

Ce test évalue l'apparition d'une séquence B, apériodique, de m bits à l'intérieur d'un des N blocs de taille M dans lesquels nous allons découper la séquence d'entrée. Le test s'évalue donc avec une fenêtre glissante de taille m.

La présence d'un patron spécifique dans une séquence dite aléatoire est un paramètre permettant d'attester de sa robustesse.

Soit W_j le nombre de fois que le patron B a été trouvé à l'intérieur de la j-ième sous-séquence. nous allons utiliser la moyenne (μ) et la variance d'une distribution normal pour l'obtention du χ^2.

$$\mu = \frac{M - m + 1}{2^m} \tag{6.28}$$

$$\delta^2 = M[\frac{1}{2^m} - \frac{2m - 1}{2^{2m}}] \tag{6.29}$$

La valeur χ^2 est donné par l'expression

$$\chi^2 = \sum_{j=1}^{N} \frac{(W_j - \mu)^2}{NP_i} \tag{6.30}$$

Le valeur-p à calculer est :

$$valeur - p = igamc(\frac{N}{2}, \frac{\chi^2}{2}) \tag{6.31}$$

6.4.8 Test de chevauchement de séquences.

Ce test évalue les apparitions d'une séquence B, apériodique, de m bits sur chacun des N blocs M bits que forme la séquence original. L'analyse s'effectue encore une fois avec une fenêtre glissante qui parcourt des blocs de m bits au long des M bits de chaque sub séquence.

L'intérêt de l'analyse se concentre dans l'évaluation, en terme des degrés de liberté (K) la quantité de fois que la séquence a été trouvée. V_k représente chacune des classes, avec k=[0 :K]. k=0 correspond au nombre de cas où le sous bloc analysé ne contient pas la séquence, et la dernière valeur correspond au cas ou le qouq bloc a été trouvé K ou bien plus de K fois dans un même sous bloc.

Associés à ces valeurs, nous avons les probabilités théoriques correspondantes (π_k), calculées en termes de λ et η :

$$\lambda = \frac{M - m + 1}{2^m} \tag{6.32}$$

$$\eta = \frac{\lambda}{2} \tag{6.33}$$

Ces deux paramètres interviennent sur la distribution de Polya-Aeppli [10], utilisés comme référence dans les études sur les critères de chevauchement.

La valeur χ^2 est donné par l'expression

$$\chi^2 = \sum_{i=0}^{K} \frac{(V_i - N\pi_i)^2}{N\pi_i} \tag{6.34}$$

La valeur-p à calculer est :

$$valeur - p = igamc(\frac{K}{2}, \frac{\chi^2}{2}) \tag{6.35}$$

L'algorithme d'analyse est explicité dans le document SP 800-22 [2].

6.4.9 Test de Maurer(Lempel-Ziv)

Ce test détermine le niveau de compressibilité d'une séquence, en analysant les bits autour d'un patron d'analyse. Si la séquence peut être compressée facilement, elle peut se considérer comme non aléatoire.

Nous allons diviser la séquence en Q segments de L bits , suivis par K segments de L bits. Les premiers Q segments permettent d'initialiser le test et les K segments sont soumis à l'algorithme de test ($K = \frac{n}{L} - Q$).

6.4.10 Test de Complexité Linéaire .

Ce test analyse la similitude entre une séquence de données et un générateur du type LFSR (donc d'une formule récursive).

La complexité linéaire est définie par la taille minimale du LFSR capable de générer les termes d'une séquence donnée. La taille calculée du générateur le plus proche indique le niveau de complexité et donc sa robustesse. Pour cela, il utilise l'algorithme de Berlekamp-Massey [3] pour déterminer la complexité linéaire de chacun des N blocs de M bits que l'on a transformé en séquences binaires et comparé avec les caractéristiques d'une séquence aléatoire.

6.4.11 Test série

Ce test évalue l'uniformité sur la distribution de séquences de données de taille fixe. Il utilise toutes les combinaisons possibles de valeurs à m bits, en cherchant le chevauchement de tels patrons. Il compare la distribution obtenue avec celle d'une séquence aléatoire, dans laquelle toutes les combinaisons possèdent la même probabilité d'apparaitre. [4]

6.4.12 Entropie Approximée

Il est très similaire au test série, mais est pris sur deux blocs adjacents de taille m et m+1. Une comparaison est effectuée avec leur équivalent pour une séquence aléatoire.

2. http ://csrc.nist.gov/publications/nistpubs/800-22-rev1a/SP800-22rev1a.pdf
3. http ://www.math.umn.edu/garrett/students/reu/MB-algorithm.pdf
4. http ://arxiv.org/pdf/1208.5740.pdf

6.4.13 Sommes Cumulatives.

Ce test analyse la valeur absolue des sommes partielles des éléments d'une séquence, pondérée sur '+1' et '-1' ($X_i = 2\varepsilon_i - 1$). Ceci permet d'estimer l'excès de '1' ou '0' existants pendant l'évolution de la séquence. Une valeur élevée d'une somme partielle correspond à une distribution non uniforme de valeurs dans l'évolution de la séquence.

Sous l'hypothèse d'une séquence aléatoire, nous allons analyser la distribution des sommes partielles.
Nous allons calculer z, la valeur maximale du module des sommes partielles S_k :

$$z = max_{1 \leq k \leq n} |S_k| \tag{6.36}$$

en fonction de la valeur de z calculée, nous allons trouver la valeur p :

$$p - value = \sum_{k=(-\frac{n}{z}+1)/4}^{(\frac{n}{z}-1)/4} [\Phi_{(\frac{(4k+1)z}{\sqrt{n}})} - \Phi_{(\frac{(4k-1)z}{\sqrt{n}})}] + \sum_{k=(-\frac{n}{z}-3)/4}^{(\frac{n}{z}-1)/4} [\Phi_{(\frac{(4k+3)z}{\sqrt{n}})} - \Phi_{(\frac{(4k+1)z}{\sqrt{n}})}] \tag{6.37}$$

dont $Phi(x)$ représente la fonction standard de probabilité normale cumulative.

6.4.14 Test Excursion Aléatoire.

Ce test compare la distribution des sommes *binaires* successives de bits qui composent une séquence avec celle d'une marche aléatoire ("random walk"), entre '1' et '0', sachant que cette dernière est une distribution normale. Nous allons définir un nombre d'états et compter le nombre de fois qu'un cycle se déplace entre tel état en aller-retour. Finalement une analyse statistique sera effectuée.

6.4.15 Test alternatif d'excursions aléatoires.

Il utilise la même observation sur le comportement de marche aléatoire mais avec l'analyse sur un autre type de distribution basé sur les événements, et donc sur la déviation du nombre de visites sur un état, comparée sur une distribution aléatoire. Le test utilise 18 états de passage possibles.

6.4.16 Considerations additionelles.

Parmi tous les tests présentés ci-dessus, suite à l'étude de leurs performances individuelles, nous pouvons avancer que :

- Le test de fréquence est le premier test à passer avant de continuer avec les autres tests. Si la distribution n'est pas proche de l'uniformité, c'est inutile de lancer le reste des tests.
- Le test d'entropie est très similaire au test série, mais considère une autre variable statistique pour l'analyse.

– Le test de complexité linéaire prend trop de ressources de calcul. Il sera alors nécessaire de trouver une solution pour s'en affranchir.

6.5 Mise en oeuvre des tests.

Après avoir étudié et présenté la référence officielle des tests NIST, comparée avec la liste d'erreurs-oublis indiqués par [19], nous avons été motivé à porter et vérifier les tests sur MatLab au lieu du code de base fourni sur C/C++, afin de réutiliser la même chaîne de modélisation et calcul au long de toutes les étapes. Les différents codes ont été proposés en annexe afin d'éviter d'alourdir le manuscrit.

Le principe du développement des codes sous Matlab, bien que fastidieuse, nous a permis d'avoir un référentiel de comparaison entre les différentes architectures proposées au sein de cette thèse mais également au sein des autres solutions précédemment développées. La seconde idée était de bien s'approprier les tests afin de ne pas avoir une suite de résultats sans savoir le pourquoi du comment. Ainsi, tous les écarts en terme de déviation aléatoire ont été expliqués et interprétés via un changement de la structure de l'oscillateur, via le code Matlab des tests NIST ou via une confirmation de résultats.

6.5.1 Principe de réalisation.

Afin de réaliser nos tests, nous disposons de trois sources possibles de données à utiliser :

– Les données provenants du modèle approximatif linéarisé. Ces données bien que très intéressantes, ne sont pas le reflet de la réalité. En effet, ces données ne tiennent pas en compte les effets parasites tels que les effets d'implantation et de transitions d'horloge qu'affectent la conversion vers des valeurs binaires.

– Les données de simulation post-Layout sur le chiptest. En prenant ces données de sortie du circuit, nous allons pouvoir appliquer un traitement des données simulées, exécuter le correcteur de biais (modelisé sur MatLab).
Cependant, un problème existe avec une taille de fichiers trop conséquentes lors de l'extraction via BSIM et ceci pour des intervalles très courts de simulation. Ensuite, avec cette information, nous devons effectuer une interpolation avec prise de données uniquement sur les fronts d'horloge concernés, afin d'alléger les tailles des fichiers.

  Ces contraintes ont limité la prise de données pour un signal de synchronisation de 250KHz, à un intervalle de 500msecs, générant des données brutes très conséquentes ($>$ 10Go de fichiers temporaires sur le serveur, triés vers des fichiers csv de plus de la 100 Mo). Le nombre de valeurs obtenues nous impose des limitations vis-à-vis de la réalisation du correcteur et l'application

de tests plus complexes.

– Les données extraites du testchip. A l'aide de la plateforme d'acquisition NI MyRio, nous avons récupéré des niveaux des séquences binaires de la sortie du circuit, synchronisés avec le signal d'horloge correspondant. Cela nous permet d'obtenir avec très peu d'utilisation mémoire/CPU, des données de l'ordre du million d'échantillons sur un fichier binaire facile à traiter sans aucun besoin d'interpolation, et avec un format de variable matrice requis par MatLab.

Concernant les conditions de validité des tests NIST (nombre d'échantillons, configuration requise du signal), il n'y a pas de grandes précisions fournies sur les documentations relatives. Toutefois, une préconisation est tout de même constante sur les différentes sources avec l'utilisation d'un échantillon 1 000 000 de points minimum sur la plupart des tests.

De plus, le test n'indique pas clairement les critères de décision sur les résultats des tests avec multiples *valeurs-p*, nous allons alors évaluer en fonction du nombre de cas avec résultats positifs et sur une moyenne du groupe de séquences utilisées. Nous allons donc soumettre aux tests NIST les signaux provenant des signaux post-Layout et sur test-chip afin de caractériser notre montage.

6.5.2 Test avec les données obtenues avec BSIM

Les contraintes posées par le moteur de simulation limitent le nombre de tests, en fonction du temps de calcul-espace mémoire requis. Ainsi, les plus complexes (dues au fait que NIST recommande une quantité de données minimale d'un million d'échantillons pour garantir des résultats significatifs dans l'analyse), ne seront pas exécutés. Cependant, nous avons lancé une partie des test sur cette séquence.

Le listing suivant, obtenu lors de l'exécution du script sur MatLab, indique la moyenne de résultats lors de l'application des tests sur les données obtenues sur BSIM, avec une taille de 148 550 points par séquence :

```
frequency test:
------------------------------------------------------------------------
sequence size |  p-value  |  Test Passed ?
------------------------------------------------------------------------
      8bit      |   0.288   |       YES
      9bit      |   0.290   |       YES
      10bit     |   0.293   |       YES
      11bit     |   0.287   |       YES
      12bit     |   0.289   |       YES
      13bit     |   0.292   |       YES
      14bit     |   0.318   |       YES
      15bit     |   0.315   |       YES
      16bit     |   0.307   |       YES
      17bit     |   0.063   |       YES
```

```
15 ____________________________________________________________________________
16 Blocktest test:
17 ----------------------------------------------------------------------------
18 block size    |  p-value  |  Test Passed ?
19 ----------------------------------------------------------------------------
20       8       |  0.655    |       YES
21      16       |  0.660    |       YES
22      32       |  0.653    |       YES
23      64       |  0.716    |       YES
24     128       |  0.457    |       YES
25     256       |  0.320    |       YES
26     512       |  0.429    |       YES
27    1024       |  0.432    |       YES
28    2048       |  0.684    |       YES
29    4096       |  0.314    |       YES
30    8192       |  0.382    |       YES
31   16384       |  0.152    |       YES
32   32768       |  0.053    |       YES
33   65536       |  0.076    |       YES
34  131072       |  0.057    |       YES
35 ____________________________________________________________________________
36 Longest Run test:
37 ----------------------------------------------------------------------------
38 Nr of segments  |  segment size    |  p-value  |  Test Passed ?
39 ----------------------------------------------------------------------------
40 23.000 segments | 128 x 49 bits.   |  0.240    |     YES
41 ____________________________________________________________________________
42 Binary matrix rank test:
43 ----------------------------------------------------------------------------
44    M     |   N    |  p-value  |  Test Passed ?
45 ----------------------------------------------------------------------------
46   32     |   32   |  0.415    |    YES
47 ____________________________________________________________________________
48 DFT test:
49 ----------------------------------------------------------------------------
50    p-value  |  Test Passed ?
51 ----------------------------------------------------------------------------
52   0.080     |   YES
53 ____________________________________________________________________________
54 Non overlapping template match test:
55 ----------------------------------------------------------------------------
56 Aperiodic     |  p-value  |  Times Passed ?
57 Template Size |   stdev   |
58 ----------------------------------------------------------------------------
59       2       |  0.0002   |   2  of  2
60       3       |  0.0390   |   4  of  4
61       4       |  0.1523   |   6  of  6
62       5       |  0.2559   |  12  of 12
63       6       |  0.2548   |  20  of 20
64       7       |  0.2453   |  40  of 40
65       8       |  0.2769   |  74  of 74
66      10       |  0.2799   | 142  of 284
67      11       |  0.2511   | 373  of 568
68      12       |  0.3205   ] 876  of 1116
```

Nous constatons à première vue que tous les tests sont valides et que le signal de sortie (en version post-Layout) est donc aléatoire.

6.5.3 Test avec les données mesurées sur le test-chip.

Dans le cas des données extraites du test-chip, nous avons pris 1 026 666 échantillons par séquence analysée, choisis aléatoirement parmi les puces avec les mêmes comportements. Nous avons alors lancé la même manipulation que celle réalisée avec les données post-layout.

```
frequency test:
-------------------------------------------------------------------------
sequence size |  p-value  |  Test Passed ?
-------------------------------------------------------------------------
8bit       |   0.288   |       YES
9bit       |   0.286   |       YES
10bit      |   0.284   |       YES
11bit      |   0.286   |       YES
12bit      |   0.285   |       YES
13bit      |   0.289   |       YES
14bit      |   0.288   |       YES
15bit      |   0.284   |       YES
16bit      |   0.269   |       YES
17bit      |   0.231   |       YES
18bit      |   0.227   |       YES

_____________________________________________________________________________
Blocktest test:
-------------------------------------------------------------------------
block size    |  p-value  |  Test Passed ?
-------------------------------------------------------------------------
8        |  0.774    |       YES
16        |  0.778    |       YES
32        |  0.675    |       YES
64        |  0.912    |       YES
128        |  0.778    |       YES
256        |  0.844    |       YES
512        |  0.974    |       YES
1024        |  0.969    |       YES
2048        |  0.815    |       YES
4096        |  0.723    |       YES
8192        |  0.611    |       YES
16384        |  0.420    |       YES
32768        |  0.979    |       YES
65536        |  0.897    |       YES
131072        |  0.842    |       YES
262144        |  0.986    |       YES
524288        |  0.862    |       YES

_____________________________________________________________________________
 Run test:
-------------------------------------------------------------------------
```

```
  p-value   |  Test Passed ?
-------------------------------------------------------------------
  0.443     |     YES
___________________________________________________________________
Longest Run Test
 ------------------------------------------------------------------
Nr of segments  |  segment size     |  p-value  |  Test Passed ?
-------------------------------------------------------------------
 128333.000     |  8 x 128333 bits |  0.639    | YES
 8020.000       | 128 x 8020 bits  |  0.867    | YES
 2005.000       | 512 x 2005 bits  |  0.867    | YES
 1026.000       |1000 x 1026 bits  |  0.867    | YES
 102.000        |10000 x  102 bits.|  0.938    | YES
___________________________________________________________________
Binary matrix rank test:
-------------------------------------------------------------------
M     |   N    |  p-value  |  Test Passed ?
-------------------------------------------------------------------
32     |   32   |  0.795    |    YES
___________________________________________________________________
DFT test:
-------------------------------------------------------------------
p-value  |  Test Passed ?
-------------------------------------------------------------------
0.094    |   YES
___________________________________________________________________
Non overlapping template match test:
-------------------------------------------------------------------
Aperiodic     |  p-value  |  Times Passed ?
Template Size |   stdev   |
-------------------------------------------------------------------
8       |  0.2899   |   73  of  74
11      |  0.1256   |  146  of 568
___________________________________________________________________
Maurer's Universal test:
-------------------------------------------------------------------
p-value  |  Test Passed ?
-------------------------------------------------------------------
0.9758   |   YES
___________________________________________________________________
Linear Complexity test:
-------------------------------------------------------------------
K     |  p-value  |  Test Passed ?
-------------------------------------------------------------------
6     |  0.2618   |   YES
___________________________________________________________________
Serial test:
-------------------------------------------------------------------
block size    | p-value1/2  |  Test Passed ?
-------------------------------------------------------------------
7             |0.5009/0.9024   |   YES
8             |0.4766/0.4891   |   YES
___________________________________________________________________
Approximate Entropy test:
```

```
95  ------------------------------------------------------------------------
96  block size    |  p-value  |  Test Passed ?
97  ------------------------------------------------------------------------
98  3bit       |   0.012   |       YES
99  4bit       |   0.106   |       YES
100 5bit      |   0.162   |       YES
101 6bit      |   0.450   |       YES
102 7bit      |   0.445   |       YES
103 8bit      |   0.000   |       NO
104 9bit      |   0.000   |       NO
105 ____________________________________________________________________________
106 Cumulative sums test:
107 ------------------------------------------------------------------------
108 p-value 1/2     |  Test Passed ?
109 ------------------------------------------------------------------------
110 0.9996/ 0.9867  |   YES
111 ____________________________________________________________________________
112 Random Excursion test:
113 ------------------------------------------------------------------------
114 p-value  |  Test Passed ?
115 ------------------------------------------------------------------------
116 0.0205   |   YES
117 ____________________________________________________________________________
118 Random Excursion Variant test:
119 ------------------------------------------------------------------------
120 p-value  |  Test Passed ?
121 ------------------------------------------------------------------------
122 0.0110   |   YES
```

De façon identique, les résultats des tests sur silicium confirment les résultats avancés sur l'état de post-Layout.

6.5.4 Discussion sur les resultats obtenus

Ayant effectué les tests statistiques, les séquences générées ont passé l'ensemble de la suite NIST. Dans le cas du test d' overlapping (test de chevauchement), il dépend de la séquence à analyser, et vu que la bibliographie disponible n'indique pas de critère de choix des patrons à vérifier sur ce test, nous allons l'isoler de l'analyse.

Du côté de la répétabilité des tests, la norme n'indique qu'une seule règle non justifiée qui est d'utiliser une quantité minimale de 1 000 000 bits par séquence analysée.

Un autre élément non indiqué clairement est le nombre de tests à effectuer pour avoir une validité de performance des résultats, étant au libre choix et contexte de l'utilisateur du test. En effet, il n'existe aucune référence justifiant les choix différents sur certains tests-paramètres réalisés par des entités qui mettent en place le test NIST.

Etant conscient du temps de calcul requis pour l'acquisition et traitement pour le test, nous allons compléter les résultats avec une quantité plus élevée de tests (100)

pour s'approximer à la définition statistique de la variable-p. Néanmoins, sur le groupe de tests réalisés, les résultats mettent déjà en évidence le caractère aléatoire et robuste des données générées.

6.6 Conclusion.

Ainsi, après avoir présenté les différents outils permettant la quantification d'un signal aléatoire, nous nous sommes attardés sur les tests NIST réputés pour leurs mises en pratique sur des signaux de type cryptographique. Partant de là, chacun des tests a été brièvement introduit avec la présentation de leur codage à titre individuel en code Matlab en annexe 1.

Une fois la base de travail réalisée, nous avons soumis notre structure au passage de ces tests avec tout d'abord l'utilisation des données post-Layout mais également de tests silicium. Ainsi, les résultats confirment dans les deux cas, que le signal obtenu est bien de type aléatoire sous réserve que les hypothèses non confirmées par la norme NIST (mais utilisées par la plupart des utilisateurs) soient bien valides.

Chapitre 7

Conclusion et Perspectives

Par ce travail de recherche présenté au sein de ce manuscrit de thèse, nous avons contribué à la conception d'un générateur aléatoire intégrable au sein d'un composant sécurisé. Ainsi, après avoir décrit plusieurs architectures de générateurs aléatoires, nous avons analysé séparément les différentes solutions et présenté leurs avantages et inconvénients tous développés en technologie standard sur silicium.

Une solution, basée sur l'échantillonnage d'une source d'entropie, a été retenue. Si l'on tient compte de l'utilisation impossible d'une source de bruit thermique directement amplifiée sur silicium liée à ses implications nocives sur les signaux sur le substrat, mais également aux demandes plus élevées en produit gain-bande nécessaires pour mener les variations intrinsèques dues au mouvement moléculaire de la source de bruit (une résistance ou une diode, i.e.), nous nous sommes alors orientés vers l'utilisation d'un générateur artificiel de bruit, basé sur un élément qui simule l'effet du bruit amplifié, avec une dynamique moins large.

Afin de choisir un tel générateur, nous avons étudié différentes solutions de systèmes chaotiques. Une réalisation en temps discret a été implémentée. En effet, la possibilité du pilotage du circuit de façon analogique et numérique, permet aussi bien la calibration que la gestion de puissance du système. Un autre point favorable à cette solution est le nombre et la simplicité des blocs nécessaires dans la réalisation d'un tel système. Par ailleurs, la dépendance d'un élément non linéaire, laisse le champ ouvert à une gamme infinie de comportements avec la possibilité d'un passage d'un signal périodique vers un signal chaotique. Par ailleurs, l'approche itérative de la topologie permet une modélisation mathématique de base facile à réaliser. Bien évidement, tout ceci est possible en partant d'une itération idéale numérique qui n'inclue pas d'information instantanée sur le chemin du signal.

Dans le but de simplifier la modélisation du système, nous sommes partis d'une fonction disposant d'un gabarit proche de la fonction "tente", avec deux régions linéaires de discontinuité. Partant de cette fonction, nous avons fait une co-simulation entre le modèle simplifié et leur contrepartie du kit AMS, afin de mettre en évidence les différences liées au comportement non linéaire intrinsèque des transistors intégrant les blocs modélisés. Le résultat de l'étude est un modèle proche des deux zones prévues mais présentant des effets rajoutés liés à la dynamique du système. Les résultats

montrent alors que le comportement obtenu montre une déviation vers trois régions et non deux zones comme précédemment attendues avec une fonction de type tente. Toutefois, nous avons constaté que la troisième région de la dynamique disparaît lors de l'ajout du bloc sur le système réalimenté d'itérations, du à l'offset mené par le suiveur. Finalement, un gabarit de deux régions traversant la ligne de symétrie est obtenu basé sur le comportement dynamique d'itérations du circuit complet.

Par ailleurs, l'orbite obtenue lors des itérations du système présente des différences liées aux effets non linéaires explicités avant, mais conserve la caractéristique attendue vers le chaos. Les éléments de l'oscillateur ont été dimensionnés avec l'utilisation d'un élément de calibration permettant le passage entre un signal périodique et un signal chaotique. En terme qualitatif, l'analyse discrète montre que le résultat obtenu est bien de type chaotique avec l'étude de la valeur de l'exposant de Lyapunov (LLE *largest Lyapunov Exponent*).

Avec une source artificielle d'aléa chaotique, nous pouvons réaliser l'échantillonnage du signal obtenu. Ceci a été effectué au travers d'un comparateur qui permet une conversion à 1bit. Le comparateur est sensible aux changements rapides du signal et comporte un niveau d'hystérésis sur la détection. Les valeurs obtenues en brut sont ensuite cadencées avec une horloge de référence, également utilisée pour le pilotage du cadencement des interrupteurs du chemin de retour du signal de l'oscillateur. Une fois les signaux convertis vers des variations binaires, elles présentent un niveau de biais inhérent. Afin d'avoir une distribution équiprobable, nous devons utiliser un circuit correcteur binaire, avec l'utilisation de générateurs pseudo-aléatoires. Leurs propriétés permettant de masquer les défaillances de la source de bruit échantillonnée, il est alors possible d'avoir une sortie aléatoire avec une distribution uniforme.

Afin de valider les caractéristiques aléatoires des séquences récupérées, nous avons mis en place le test de validation NIST. Ainsi, à partir des données récupérées en sortie de l'oscillateur, nous constatons qu'ils passent les 15 tests à l'exception du test de non-chevauchement, (basé sur une fenêtre de plus de 8 bits pour l'étude de chevauchements). Les résultats indiquent que la source de séquences aléatoires possède des caractéristiques d'imprédictibilité.

Concernant les résultats sur silicium via la réalisation d'un testchip, les résultats sont très satisfaisants sachant les critères précis attendus. Toutefois, nous pouvons également mettre en avant quelques caractéristiques susceptibles d'amélioration. En effet, vu que cette structure chaotique est censée s'intégrer au sein d'un système plus complexe sur Silicium, le circuit de base a été conçu pour s'autogérer. Or, lors de la réalisation du test-chip, nous avons dû intégrer des plots de test et donc les éléments parasites associés. Ainsi, les effets des plots de sortie sur ces bornes ont mené des pertes. Les capacités et éléments ajoutés ont changé la dynamique interne, en faisant varier, par exemple, les capacités de l'oscillateur en anneau, modifiant la fréquence du signal et aussi les points de polarisation. Heureusement, ces variations

ont pu être compensées par l'application de tensions de polarisation externes mais également en faisant varier la tension d'alimentation (la première option étant à conseiller sur une implantation finale). Les signaux conservent malgré tout la détérioration des patrons non périodiques une fois le point d'entrée sur le chaos atteint. Nous avons ainsi obtenu un signal aléatoire, mais au détriment des caractéristiques prévues sur le banc de test d'extraction post-Layout. Cela impliquera un travail des interfaces d'accès en sortie des testpoints pour garantir une meilleure robustesse contre les capacitances des plots....

Nous avons également constaté, dans la partie test et validation, que les références existantes sur les protocoles de test ne sont pas clairement énoncées. Au delà des conseils du NIST par rapport à la quantité de données par séquence de test, l'utilisation des patrons de test et des conditions spécifiques qui permettent de diminuer les temps de calcul restent comme un savoir faire 'personnel' des entités qui mettent en place ces algorithmes. Dans notre cas, nous avons donc essayé de trouver un équilibre entre les consignes du NIST, et des concepteurs qui appliquent ces consignes ou font des addendums sur l'information officielle, sur laquelle restent encore des lacunes et erreurs typographiques importantes et imprécisions sur certains énoncés.

Afin de s'assurer l'efficacité des tests, nous avons exécuté aussi de nouveaux tests en comparaison avec d'autres générateurs déjà validés existants. Parallèlement, nous avons lancé une quantité de tests autour de la centaine de séquences de 2Mbits, afin d'avoir une granularité plus élevé sur les comparatifs statistiques sur les valeurs statistiques obtenues dans l'ensemble de tests effectués sur chaque séquence prise aléatoirement.

Enfin, ce travail de recherche a ouvert bien plus de perspectives via une première réalisation sur silicium ayant montré la possibilité de réaliser une source d'aléa très basse consommation. Ce travail a été encouragé par des présentations à des conférences internationales ou de nouvelles perspectives d'applications se sont présentées. C'est donc avec joie que même si ce travail de recherche n'est qu'un début et devrait se poursuivre via d'autres projets.

Annexe 1 : Portage du test NIST sur Matlab

Cet annexe regroupe l'intégralité des codes Matlab relatifs au test NIST développés dans le but de valider le caractère aléatoire de notre signal de sortie.

1. Le test de fréquence global (monobit)

```
% TEST NUMBER 01/  The Frequency (monobit) Test
function [passed S_obs P_valueavg] = frequencytest(input_sample,m)
length_sample=2^m;
%number of loops to perform before an std dev
dΔ = length(input_sample)-length_sample;
P_valueaccum=[];
for j=0:dΔ
    input_sample1=input_sample(1+j:length_sample+j);
% step1 let's ponderate our sample before analyse
    ponderate_sample=(input_sample1==1)-(input_sample1==0);
% calculate sum then mean..
    S=sum(ponderate_sample);
    S_obs=abs(S)/sqrt(length_sample);
    P_value=erfc(S_obs/sqrt(2));
    P_valueaccum=[P_valueaccum P_value];
end
P_valueavg= std(P_valueaccum)passed=0;
if P_valueavg≥0.01 % Decision Rule (at 1% level)
    passed=1;
end
```

2. Le test de fréquence par blocs

```
% Test number 002:  Frequency test within a block
function [passed chisquare_obs P_value] = blocktest(input_sample,blocksize)
length_sample=length(input_sample);% ≥ 100 elements
offset=mod(length_sample,blocksize);
nslices=fix(length_sample/blocksize);
slices=[];slicesum=[];
for jj=1:nslices
```

```
    slices(:,jj)=(input_sample((blocksize*(jj-1)+1):jj*
blocksize))';
    slicesum=[slicesum sum(slices(:,jj))];
end
%proportion pi vectors:
pi_slices=slicesum/blocksize;
chisquare_obs=4*blocksize*(sum((pi_slices-0.5).^2));%chisquare obs
Pax=gammainc(chisquare_obs/2,nslices/2);%=p(a,x)  % igamc function:
P_value=1-Pax;%Q(a,x)=1-p(a,x)
passed=0;
if P_value≥0.01 % Decision Rule (at 1% level)
    passed=1;
end
```

3. Le test de répétition global

```
% Test Number 003:  The Runstest!!
function [passed Vn_obs P_value] = runtest(input_sample)
% input_sample=[1 1 0 0 1 0 0 1 0 0 0 0 1 1 1 1 1 1 0 1 1 0 1 0 1 0
%1 0 0 0 1 0 0 0 1 0 0 0 0 1 0 1 1 0 1 0 0 0 1 1 0 0 0 0 0 1 0 0 0 1 1 0
% 1 0 0 1 1 0 0 0 1 0 0 1 1 0 0 0 1 1 0 0 1 1 0 0 0 1 0 1 0 0 0 1 0 1
%1 1 0 0 0 ];
% runs (repeated bits) Test
% frequency test must be passed before this
% first, i count the numbers of 1 and 0's
%input_sample=[1 0 0 1 1 0 1 0 1 1]%debug
length_sample=length(input_sample);% ≥ 100 elements

pi_sum=sum(input_sample)/length_sample
tau=2/sqrt(length_sample);

if abs(pi_sum-0.5) ≥ tau
    passed=1;
    P_value=0;
    Vn_obs=0;
    return;%break;
end


r=[];
for jj=1:length_sample-1
    if (input_sample(jj+1)==input_sample(jj))
    r=[r 0];
    else
    r=[r 1];
    end;

end
Vn_obs=1+sum(r);

```

```
P_value=erfc(abs(Vn_obs-2*length_sample*pi_sum*(1-pi_sum))/
(2*sqrt(2*length_sample)*pi_sum*(1-pi_sum)));
% decision
passed=0;
if P_value≥0.01
    passed=1;
end
%passed;
```

4. Le test de répétition par blocs

```
% Test Number 004:  The Longest Run of ones !!
function [passed P_value  N n_groups] = longestruntest(input_sample,
M,K)
if (M==8)
    pival=[0.21484375 0.3671875 0.23046875 0.1875];V0=1;
end
if (K==5)
    if (M==128)
        pival=[0.1174 0.2430 0.2493 0.1752 0.1027 0.1124];V0=4;
    end
    if (M==512)
        pival=[0.1170 0.2460 0.2523 0.1755 0.1027 0.1124];V0=6;
    end
    if (M==1000)
        pival=[0.1307 0.2437 0.2452 0.1714 0.1002 0.1088];V0=7;
    end
end
if (M==10000)
    pival=[0.0882 0.2902 0.2483 0.1933 0.1208 0.0675 0.0727];V0=10;
end
n=length(input_sample);% ≥ 100 elements
N=floor(n/M);
n_groups=N;
accum_Pvalues=[];tmp=[];group=[];runz=[];
for ii=1:N
    tmp=input_sample(M*(ii-1)+1:M*ii);
    group=[group tmp'];%;%separated groups are created
end;
% now to find the longest runs of patterns:detect pattern 1  , 11,
111, 1111, etc
    acc1=[];acc2=[];acc=0;
    [aa,bb]=size(group);
    vvalue=[];
    for jjj=1:bb
        acc=0;
        tmp=group(:,jjj);% vector size (Mx1)
        acc1=[];acc2=[];
        for iii=1:M
            acc= (acc + tmp(iii)) * (tmp(iii)==1);
```

```
            acc1=[acc1 acc];
        end
        acc2=max(acc1);
        vvalue=[vvalue acc2];
    end;
% now recollect the values and sum
    vvector=[];
    for p=V0:(V0+K)
        if (p==V0)
            temp=sum(vvalue≤p); %V0
            vvector=[vvector temp];
        elseif (p==V0+K)
            temp=sum(vvalue≥p); %V0+K
            vvector=[vvector temp];
        else
            temp=sum(vvalue==p); %others
            vvector=[vvector temp];
        end
    end
    vvector;
pival;
% step3 now compute chi2 using M;K;N
chi2=((vvector- N*pival).^2)./(N*pival);
    chi_observed=sum(chi2);
% igamc function:::
    Pax=gammainc(chi_observed/2,K/2)/(2^(K/2)*gamma(K/2));%=p(a,x)
    P_value=1-Pax; %Q(a,x)=1-p(a,x)
passed=0;
if P_value≥0.01
    passed=1;
end
```

5. Le test de rang de la matrice binaire

```
%Test number 005: Binary matrix Test!!
function [passed M Q P_value] = binarymatrix(input_sample)
n=length(input_sample);%length of the bit string
M=32; %3;
Q=32; %3; souhaitable 32 * 32  approx 1000
N=floor(n/(M*Q));%  number of blocks under test
% we will use blocks of  MQ sized bit matrices.
% the rest of matrices can go away!!!
storage=[];reloc=[];
for i=1:N
    reloc=input_sample(1+(i-1)*M*Q:i*M*Q);
    storage=[storage reloc'];
end
% so you can recover each i-th  M*Q group by storage(:,i)
% now let's transform a column into a single MxN temporary matrix
R=[];
```

```
for i=1:N
    temp=reshape(storage(:,i),M,[])';
    R=[R rank(gf(temp))];
% gf performs an special data conditionning allowing binary or
%operation
% instead of normal calculation allowing the right results
end
fm=find(R==M);
f_m=length(fm);
fm1=find(R== (M-1));
f_m1=length(fm1);
remaining=N-f_m1-f_m;
chi_obs=(f_m-0.2888*N)^2/(0.2888*N)+(f_m1-0.5776*N)^2/(0.5776*N)+..
        (remaining-0.1336*N)^2/(0.1336*N);
P_value=exp(-chi_obs/2);%1-gammainc(chi_obs/2,1)
passed=0;
if P_value≥0.01
    passed=1;
end
```

6. Le test spectral

```
%Test number 006: DFT Test!!
function [passed P_value] = DFTest(input_sample)
Xk=2*input_sample-1; %ponderation
nn=length(Xk);
fk=fft(Xk,nn);
S=fk(1:fix(nn/2));
M=abs(S);
plot(M,'b');hold on;plot(M,'b-');grid ;
T=sqrt(2.995732274*nn);%Threshold fixed
No=0.95*nn/2;
n1=find(M<T); %<
N1=length(n1);
d=(N1-No)/sqrt(nn*.05*.95/4);% Corrections of the NIST Statistical
% Test Suite for Randomness Song-Ju Kim, Ken Umeno, and Akio Hasegawa
P_value=erfc(abs(d)/sqrt(2))
passed=0;
if P_value≥0.01
    passed=1;
end
```

7. Le test de non chevauchement de séquences

```
%Test number 007: Non overlapping template matching test!!
function [passed P_value] = novtempmatch(input_sample,B)
```

```
n=length(input_sample);% ≥ 100 elements
N=8;%
M=fix(n/N);%slice it on Ngroups of Mpoints
R=[];
for i=1:N%relocate each group as columns.
    R=[R input_sample(1+(i-1)*M:i*M)'];
end
% Set a template pattern to be matched : B (has m bits)
m=length(B);
B=B';
W=[];
for i=1:N %analysis for N blocks of M elements
  temp=R(:,i);
  tst=[];
  for k=0:1:M-m
    A=temp(1+k:m+k);
    tst=[tst isequal(A, B)];
  end
  W=[W sum(tst)];
end
mu=(M-m+1)/(2^m);
chi2=M*((1/2^m)-((2*m-1)/2^(2*m)));%ok
chiobs=(W-mu).^2;
chi_observed=sum(chiobs)/chi2;
% igamc function:::
 Pax=gammainc(chi_observed/2,N/2);%=p(a,x)
P_value=1-Pax;%Q(a,x)=1-p(a,x)
passed=0;
if P_value≥0.01
    passed=1;
end
```

8. Le test de chevauchement de séquences

```
%Test number 008: Non overlapping template matching test!!
function [passed P_value] = overlaptmatch(input_sample,B)
K= 5;%2 set up the degrees of freedom  (hardcoded)
M= 1032;%M: Length of substring  (hardcoded)  1032
%N:number of substrings
%B: m bit template to match with
n=length(input_sample);% ≥ 100 elements
m=length(B);%length of the testchain
N=968;%fix(n/M)%;% partition on N independent blocks
%slice it on Ngroups of Mpoints
%relocate each group as N columns.
R=[];
for i=1:N
    R=[R input_sample(1+(i-1)*M:i*M)'];
end
% Set a template pattern to be matched :  B (B has m bits)
```

```
B=B';
W=[];
for i=1:N %analysis for N blocks of M elements
  temp=R(:,i);
  tst=0;
  k=0;
  while k<M-m
     A=temp(1+k:m+k);% test a m bit seq
     if isequal(A,B)==1
         tst=tst+1;
     end
     k=k+1;
  end

W=[W tst];
end
% now take W and generate the V vector
V=[0 0 0 0 0 0];
V(1)=sum(W==0);
V(2)=sum(W==1);
V(3)=sum(W==2);
V(4)=sum(W==3);
V(5)=sum(W==4);
V(6)=sum(W≥5);
%****************************************
lambda=(M-m+1)/(2^m);
eta=lambda/2;
%calculating pivals...
pivals(1)=exp(-eta);
pivals(2)=eta/2*pivals(1);
pivals(3)=(eta+2)*pivals(2)/4;
pivals(4)=((eta^2)/6+eta+1)*pivals(2)/4;
pivals(5)=((eta^3)/24+(eta^2)/2+3*eta/2+1)*pivals(2)/8;
pivals(6)=0.139865;

mu=(M-m+1)/(2^m);
chi2=M*((1/2^m)-((2*m-1)/2^(2*m)));%ok
chiobs=((V-N*pivals).^2)./(N*pivals);
chi_observed=sum(chiobs)

P_value=1-chi2cdf(chi_observed,K);%1-Pax;%Q(a,x)=1-p(a,x)
passed=0;
if P_value≥0.01
    passed=1;
end
```

9. Le test universel de Maurer

```
%Test number 009: Maurer universal statistical test!!
function [passed P_value] = maurertest(input_sample)
```

```
%L=length of each block
%Q=number of blocks in the inititialization sequence
L=0;
n=length(input_sample);
if n<387840
  sprintf('%s','sample size must be > 387840 points needed. quitting
test')
  return;
end
if n≥387840
  L=6;
end
if n≥904960
  L=7;
end
if n≥2068480
  L=8;
end
if n≥4654080
  L=9;
end
if n≥10342400
  L=10;
end
if n≥22753280
  L=11;
end
if n≥49643520
  L=12;
end
if n≥107560960
  L=13;
end
if n≥231669760
  L=14;
end
if n≥496435200
  L=15;
end
if n≥1059061760
  L=16;
end

if L>0
Q=10*2^L;
a=mod(n,L);
K=(n-a)/L - Q;
% So the input sample is splitted in:
% Init segment:  Q*L bits  % Test:  K*L bits
% the rest: discarded!
Init=[];
for i=1:Q
  Init=[Init input_sample(L*(i-1)+1:L*i)'];
end
```

```
57
58  Test=[];
59  for i=1:K
60    Test=[Test input_sample(Q*L+L*(i-1)+1:Q*L+L*i)'];
61  end
62
63  Binary_T=[];%bintodec for Q%
64  [u v] = size(Init);%u is used to binarize
65  for j=1:v
66     T_bin=0;
67     tmp=fliplr(Init(:,j)');
68     for i=1:u
69       T_bin=T_bin+tmp(i)*2^(i-1);
70     end
71     Binary_T=[Binary_T T_bin];
72  end
73  Binary_T=sort(Binary_T);
74  %now do separate associative groupings
75  accQ=[];
76  for i=0:2^u-1
77     va=find(Binary_T==i);
78     accQ=[accQ length(va)];
79  end
80  %acc contains de pondered position of each ocurrence
81  % index1= T0(number of "0s")  index2=T1 (number of "1s")....
82  %now for K
83  Binary_K=[];
84  [u v] = size(Test);
85  %u is used to binarize
86  for j=1:v
87  T_bin=0;
88  tmp=fliplr(Test(:,j)');
89  for i=1:u
90  T_bin=T_bin+tmp(i)*2^(i-1);
91  end
92  Binary_K=[Binary_K T_bin];
93  end
94  accQ;
95  Binary_K;
96  accK=accQ;
97  sum=0;
98  for i=1:K
99  sss=accK(Binary_K(i)+1);
100 accK(Binary_K(i)+1)=Q+i;
101 sum=sum+log2(Q+i-sss);
102 end
103 fn=sum/K; % the test statistic
104 % selection of expectedvalue from a list
105 if L==6
106     expected_value=5.2177052;variance=2.954;
107 elseif L==7
108     expected_value=6.1962507;variance=3.125;
109 elseif L==8
110     expected_value=7.1836656;variance=3.238;
```

```
elseif L==9
    expected_value=8.1764248;variance=3.311;
elseif L==10
    expected_value=9.1723243;variance=3.356;
elseif L==11
    expected_value=10.170032;variance=3.384;
elseif L==12
    expected_value=11.168765;variance=3.401;
elseif L==13
    expected_value=12.168070;variance=3.410;
elseif L==14
    expected_value=13.167693;variance=3.416;
elseif L==15
    expected_value=14.167488;variance=3.419;
elseif L==16
    expected_value=15.167379;variance=3.421;
end
c=0.7- (0.8/L)+(4+32/L)*(K^(-3*L)/15);
sigmaa=c*sqrt(variance/K)
Pax=erfc(abs((fn-expected_value)/(sqrt(2*sigmaa))));
P_value=Pax%Q(a,x)=1-p(a,x) % igamc function:::
passed=0;
if P_value≥0.01
    passed=1;
end
passed
end;
```

10. Le test de la complexité linéaire

```
function [passed P_value] = complin(input_sample)
n=length(input_sample);M=500;
N=floor(n/M);%M=fix(n/N);
K=6;
%relocate each group as columns.
R=[];
for i=1:N
    R=[R input_sample(1+(i-1)*M:i*M)'];
end
Lvector=[];
for i=1:N %analysis for N blocks of M elements
  temp=R(:,i);
  [f,LCP]=Berlekamp_Massey2(temp');
  L=max(LCP);
  Lvector=[Lvector L];
end
mu=M/2 + ( 9 + (-1)^(M+1))/36 - (M/3+2/9)/(2^M);
Tvector=(-1)^M*(Lvector-mu)+2/9;
%create and calculate V vector from Tvector
V=[0 0 0 0 0 0 0];
```

```
V(1)=sum(Tvector<-2.5);
V(2)=sum(and(Tvector≥-2.5,Tvector≤-1.5));
V(3)=sum(and(Tvector>-1.5,Tvector≤-0.5));
V(4)=sum(and(Tvector>-0.5,Tvector≤0.5));
V(5)=sum(and(Tvector>0.5,Tvector≤1.5));
V(6)=sum(and(Tvector>1.5,Tvector≤2.5));
V(7)=sum(Tvector>2.5);
pivals=[0.01047 0.03125 0.125 0.5 0.25 0.0625 0.02078];
chi=(V-N*pivals).^2 ./ (N*pivals);
chiobs=sum(chi);
%*******************************
P_value=1-chi2cdf(chiobs,K);
passed=0;
if P_value≥0.01
    passed=1;
end
```

11. Le test Série

```
% Test Number 011:  The Runstest!!
function [passed P_val1 P_val2] = serialtest(input_sample,m)
n=length(input_sample);
%augmented sequence now count the frequencies of overlapping
% for m , m-1 and m-2 bit combinations
matrice=[];psi=[];
for (j=m-2:1:m)
matrice=zeros(1,2^j);
e1=[input_sample input_sample(1:j-1)]; % add the first m-k elements
n1=length(e1);
    for i=0:2^j-1  %de2bi(number,padding);
         pattern= de2bi(i,j,'left-msb')  ;
         bitplace=0;
         for k=1:1:n1-m+1
             tmp=e1(k:k+j-1);
             if isequal(tmp,pattern)==1
                bitplace=bitplace+1;
             end
         end
         matrice=[matrice bitplace];
    end;
         ppsi2=((2^j)/n)*sum(matrice.^2)-n;
         psi=[psi ppsi2];
end;
rotm2=psi(3)-psi(2);
rot2m2=psi(3)-2*psi(2)+psi(1);
%*******************************
P_val1=1-chi2cdf(rotm2,2^(m-1));% igamc function:::
passed=0;
P_val2=1-chi2cdf(rot2m2,2^(m-2));% igamc function:::
if and(P_val1≥0.01,P_val2≥0.01)==1
```

```
    passed=1;
end
```

12. Le test de l'entropie approximée

```
% Test Number 012:  The Runstest!!
function [passed P_val] = approxentropy(input_sample,m)
%input_sample=[1 1 0 0 1 0 0 1 0 0 0 0 1 1 1 1 1 1 0 1 1 0 1 0 1
%0 1 0 0 0 1 0 0 0 1 0 0 0 0 1 0 1 1 0 1 0 0 0 1 1 0 0 0 0 1 0 0
%0 1 1 0 1 0 0 1 1 0 0 0 1 0 0 1 1 0 0 0 1 1 0 0 1 1 0 0 0 1 0 1
%0 0 0 1 0 1 1 1 0 0 0 ] ;
%m=2;  m< log2(length(outstream))-5
%uu=log2(length(input_sample))-5
%if m ≥ uu
%sprintf('%s','m too big. quitting')
%return;
%end
%for j=m:m+1% 2 block lengths on analyse
 %**********************************************
 %append m-1 bits
 n=length(input_sample);
 %augment the sequence by appending the first m-1 bits
 e1=[input_sample input_sample(1:m-1)];
 n1=length(e1);
 e2=e1;
 % splice in blocks of m bit size, overlapped
 R=[];
 ppp=n1-m+1;

 %for pp=1:ppp
 %R=[R e2(pp:m-1+pp)'];
%     R=[R (e2(1:m))'];
%    e2(1)=[];
 %end
% check overlapping blocks
%**********************************************
 matrice=zeros(1,2^m);
 for i=0:2^m-1  %de2bi(number,padding);
          pattern= de2bi(i,m);% m bit pattern to match
          flag=0;
          for(k=1:ppp)
              %tmp=(R(:,k))';
              %if (isequal(pattern,e2(k:m-1+k))==1)
              flag=flag+isequal(pattern,e2(k:m-1+k));%1;
              %end;
          end;
          matrice(i+1)=flag;
end;
Cii=matrice/n;Ci=[];
nCi=length(Cii);
```

```
for w=1:nCi
    if Cii(w)==0
    else
        Ci=[Ci Cii(w)];
    end
end

psim=Ci.*log(Ci);
psi1=sum(psim);

%*************
m=m+1;
%augment the sequence by appending the first m bits
 e1=[input_sample input_sample(1:m-1)];
 n1=length(e1);
 e2=e1;
 % splice in blocks of m bit size, overlapped
 R=[];
 ppp=n1-m+1;

% for pp=1:ppp
% R=[R e2(pp:m-1+pp)'];
%     R=[R (e2(1:m))'];
%    e2(1)=[];
 %end
 % check overlapping blocks
%*********************************************
 matrice=zeros(1,2^m);
 %matrice=[];
 for i=0:2^m-1  %de2bi(number,padding);
          pattern= de2bi(i,m);% m bit pattern to match
          flag=0;
          for(k=1:ppp)
%               if (isequal(pattern,e2(k:m-1+k))==1)
%             tmp=R(:,k);
%              if (isequal(pattern,tmp')==1)
              flag=flag+isequal(pattern,e2(k:m-1+k));
              %end;
          end;
          matrice(i+1)=flag;
end;

Cii=matrice/n;Ci=[];
nCi=length(Cii);
for w=1:nCi
    if Cii(w)==0
    else
        Ci=[Ci Cii(w)];
    end
end

psim=Ci.*log(Ci);


```

```
psi2=sum(psim);
m=m-1;%restore value of m

apen=psi1-psi2;
chi2=2*n*(log(2)-apen);%2*n* ??
%
% % igamc function:::
 P_val=1-gammainc(chi2/2,2^(m-1));
passed=0;
 if P_val≥0.01
     passed=1;
 end
```

13. Les sommes cumulatives

```
% Test Number 013:  The cumulative sums tests!!
function [passed P_val1] = cusum(input_sample)
n=length(input_sample);
% 0 forward - 1 backward.
e=input_sample;
esign=2*e-1;
sum0=[];sum1=[];
for i=1:1:n
    tmp=sum(esign(1:i));
    sum0=[sum0 tmp];
end
for i=1:1:n
    tmp=sum(esign(n-i+1:n));
    sum1=[sum1 tmp];
end
z0=max(sum0);%forward
z1=max(sum1);%backwards
% forward
s1 = 0;
s2=0;
    for k=fix((-n/z0+1)/4):fix((n/z0-1)/4)
        s1 = s1  + normcdf(((4*k+1)*z0)/sqrt(n))
- normcdf(((4*k-1)*z0)/sqrt(n));
    end
    for k=fix((-n/z0-3)/4):fix((n/z0-1)/4)
        s2 = s2 + normcdf(((4*k+3)*z0)/sqrt(n))
- normcdf(((4*k+1)*z0)/
sqrt(n));
    end
P_val1=1-s1+s2
% backward
s1 = 0;s2=0;
    for k=fix((-n/z1+1)/4):fix((n/z1-1)/4)
        s1 = s1  + normcdf(((4*k+1)*z1)/sqrt(n))  - normcdf(((4*k-1)*z1)/sqrt(n));
    end
```

```
    for k=fix((-n/z1-3)/4):fix((n/z1-1)/4)
        s2 = s2 + normcdf(((4*k+3)*z1)/sqrt(n)) - normcdf(((4*k+1)*z1)/
sqrt(n));
    end
P_val2=1-s1+s2
    passed=0;
 if P_val1≥0.01
     passed=1;
 end
```

14. L'excursion aléatoire

```
% Test Number 015:  The random excursion tests!!
function [passed P_val] = randomwalk(input_sample)
pi1=[0.5000 0.2500 0.1250 0.0625 0.0312 0.0312];
pi2=[0.7500 0.0625 0.0469 0.0352 0.0264 0.0791];
pi3=[0.8333 0.0278 0.0231 0.0193 0.0161 0.0804];
pi4=[0.8750 0.0156 0.0137 0.0120 0.0105 0.0733];
pi5=[0.9000 0.0100 0.0090 0.0081 0.0073 0.0656];
pi6=[0.9167 0.0069 0.0064 0.0058 0.0053 0.0588];
pi7=[0.9286 0.0051 0.0047 0.0044 0.0041 0.0531];
pivals=[pi1;pi2;pi3;pi4;pi5;pi6;pi7];
n=length(input_sample);
esign=2*input_sample-1;
walk=zeros(1,n+2);
for i=1:1:n
    walk(i+1) = sum(esign(1:i));
end
onlyzeros=find(walk==0);%
%now detect each 0xxx0 pattern from the obtained sequence
lastz=length(onlyzeros);
%filter the last possible 00 combination
if onlyzeros(lastz)==onlyzeros(lastz-1)
    walk(max(onlyzeros))=[];
    onlyzeros(max(lastz))=[];
end
J=length(onlyzeros)-1;
%  now perform on the fly the tests, the variable size of the samples
%is a
%  pain to prelocate or stack columns for any 0xx0
% possible pattern.
%  possible values that occur:
% -4 -3 -2 -1 1 2 3 4
possibles= [-4 -3 -2 -1  1  2  3 4];
Vtest=[];
for i=1:J
%dummy=walk(
blank    = [ 0  0  0  0  0  0  0 0]; %initial state for check
a=onlyzeros(i:i+1); %pick sequence to test position
u=walk(a(1)+1:a(2)-1);% extract the cycle to be tested
```

```
%and now the  iterative test
    for j=1:8%length of blank/possibles
        blank(j)=sum(u==possibles(j));
    end
Vtest=[Vtest ; blank]  ;
end
%now do a retest to perform the final count...
Vvector=[];
for k=1:8
    pit=[0 0 0 0 0 0];
    uu=Vtest(:,k);
    pit(1)=  sum(uu==0);
    pit(2)=  sum(uu==1);
    pit(3)=  sum(uu==2);
    pit(4)= sum(uu==3);
    pit(5)= sum(uu==4);
    pit(6)= sum(uu≥5);
    Vvector=[Vvector ;pit];
end
%now perform a calculation using the chi2
Chi2=zeros(1,8);
for k=1:8
    x=possibles(k);
    y=abs(x);
    Chi2(k)=sum(((Vvector(k,:)-J*pivals(y,:)).^2)./(J*pivals(y,:)))/2;
end
 Px_val=1-gammainc(Chi2,2.5)  ;%5/2
 P_val=min(Px_val)
passed=0;
 if P_val≥0.01
     passed=1;
 end
```

15. Le variant sur l'excursion aléatoire

```
% Test Number 015:  The random excursion variant tests!!
function [passed P_val] = randomwalkvar(input_sample)
n=length(input_sample);
esign=2*input_sample-1;
walk=zeros(1,n+2);
for i=1:1:n
    walk(i+1) = sum(esign(1:i));
end
onlyzeros=find(walk==0);%
%now detect each 0xxx0 pattern from the obtained sequence
lastz=length(onlyzeros);
%filter the last possible 00 combination
if onlyzeros(lastz)==onlyzeros(lastz-1)
    walk(max(onlyzeros))=[];
    onlyzeros(max(lastz))=[];
```

```
end
J=length(onlyzeros)-1;
peak=9;
% construct a matrix of all the possible states
possibles= [ -peak:-1  1:peak];
Vtest=[];
for i=1:J
%dummy=walk(
blank    = zeros(1,18); %initial state for check
a=onlyzeros(i:i+1); %pick sequence to test position
u=walk(a(1)+1:a(2)-1);% extract the cycle to be tested
%and now the  iterative test
    for j=1:18%length of blank/possibles
        blank(j)=sum(u==possibles(j));
    end
Vtest=[Vtest ; blank]  ;
end
%now do a retest to perform the final count...
Vvector=zeros(1,18);
for jj=1:J
    Vvector=Vvector+Vtest(jj,:);
end
P_val=zeros(1,18);
for kk=1:18
%now calculate the p value using the erfc function¨
    P_val(kk)=erfc(abs(Vvector(kk)-J)/
sqrt((2*J*(4*abs(possibles(kk))-2))));
end
passed=0;
 if min(P_val)≥0.01
     passed=1;
 end
 P_val=min(P_val)
```

Example d'utilisation des scripts

```
%input sequence: outstream1
clc;
%*******************
fprintf('    Data size %10.0f',length(outstream1));
fprintf('************************************************\n' );
fprintf('performing frequency test:\n' );
mmm=floor(log2(length(outstream1)));
for i=8:mmm
[passed S_obs P_value]=frequencytest(outstream1,i);%tst1
fprintf(' -- size sequence= %10.0f',i );
fprintf(' -- P_Value= %6.3f',P_value );
fprintf(' -- Test Passed: ' );
    if (passed==1)
        fprintf('YES\n' );
```

```
    else
        fprintf('NO\n' );
    end
end;
%******************
fprintf('**************************************************\n' );
%tst2
fprintf('performing blocktest:\n' );
for i=3:mmm %26 max in demo official
[passed chisquare_obs P_value]=blocktest(outstream1,2^i);
fprintf('    Block Size %4.0f',2^i);
fprintf(' -- P_Value= %6.3f',P_value );
fprintf(' -- Test Passed: ' );
    if (passed==1)
        fprintf('YES\n' );
    else
        fprintf('NO\n' );
    end
end;
fprintf('**************************************************\n' );
%tst03
fprintf('performing runtest:\n' );
[passed Vn_obs P_value] =runtest(outstream1); % ok
fprintf(' -- P_Value= %6.3f',P_value );
fprintf(' -- Test Passed: ' );
    if (passed==1)
        fprintf('YES\n' );
    else
        fprintf('NO\n' );
    end
%******************
fprintf('**************************************************\n' );
%tst04
fprintf('performing longest run test:\n' );
%rewritten function:
% M can be:  8   128    512    1000    10000
% K can be:  3     5      5       5        6
tM=[8   128    512    1000    10000];
tK=[3     5      5       5        6];
u=length(tM);
for i=1:u

    [passed P_value  N n_groups] = longestruntest(outstream1,
tM(i),tK(i));
    fprintf(' Analysis: %6.3f',n_groups );
    fprintf(' segments of: %4.0f',tM(i));fprintf(' x %4.0f',N);
    fprintf(' bits. P_Value= %6.3f',P_value );
    fprintf(' -- Test Passed: ' );
        if (passed==1)
            fprintf('YES\n' );
        else
            fprintf('NO\n' );
        end
end
```

```
%********************
fprintf('**************************************************\n' );
%tst05
fprintf('performing binary matrix rank test:\n' );
[passed M Q P_value]=binarymatrix(outstream1);% fixed!
fprintf(' Parameters: M= %4.0f',M);fprintf('Q= %4.0f',Q);
fprintf(' -- P_Value= %6.3f',P_value );
fprintf(' -- Test Passed: ' );
    if (passed==1)
        fprintf('YES\n' );
    else
        fprintf('NO\n' );
    end
%********************
fprintf('**************************************************\n' );
%tst06
fprintf('performing DFT test:\n' );
[passed P_value] = DFTest(outstream1);%ok
fprintf(' -- P_Value= %6.3f',P_value );
fprintf(' -- Test Passed: ' );
    if (passed==1)
        fprintf('YES\n' );
    else
        fprintf('NO\n' );
    end
%********************
%tst07 - sizes to test: 2 to  8
fprintf('**************************************************\n' );
fprintf('performing non overlapping template match test:\n' );
for i=5:9
 fprintf('aperiodic template size: %2.0f',i);fprintf('\n' );
templateset=aperiodic_templates(i);
pass_counter=0;
tmp_pvalues=[];
[mm nn]=size(templateset);
    for j=1:mm
        B=templateset(j,:);
        [passed P_value]=novtempmatchA(outstream1,B);
        fprintf(' template=');fprintf('%1.0f',B );
        fprintf(' -- P_Value= %6.4f',P_value );
        fprintf(' -- Test Passed: ' );
        tmp_pvalues=[tmp_pvalues P_value];
        if (passed==1)
            fprintf('YES\n' );
             pass_counter=pass_counter+1;
        else
            fprintf('NO\n' );
        end
    end
figure();hist(tmp_pvalues);
ul=    std(tmp_pvalues);
fprintf('P_value std dev: %2.4f', ul);fprintf('\n' );
fprintf(' times passed =');fprintf('%10.0f',pass_counter );
fprintf(' of ');fprintf('%10.0f',mm ); fprintf(' tests \n');
```

```
123 end
124 %************************************************************
125 %test8 clc
126 fprintf('*************************************************\n' );
127 fprintf('performing overlapping template match test:\n' );
128 for i=4
129     fprintf('pattern length: %2.0f', i);fprintf('\n' );
130     for j=2^i-2:(2^i-1)
131        B= de2bi(j,i,'left-msb');  %B=de2bi(170,i,'left-msb');
132     [passed P_value] = overlaptmatch(outstream1,B);
133     fprintf('  -- B: '); fprintf('%1.0f', B);
134     fprintf('  -- Pvalue: %3.3f', P_value); fprintf('  Passed: ');
135         if (passed==1)
136             fprintf('YES\n' );
137         else
138             fprintf('NO\n' );
139         end
140     end
141 end
142 %************************************************************
143 %test9
144 fprintf('*************************************************\n' );
145 fprintf('performing maurer test:\n' );
146     [passed P_value] = maurertest(outstream1)
147     fprintf('  -- Pvalue: %3.3f', P_value); fprintf('  Passed: ');
148         if (passed==1)
149             fprintf('YES\n' );
150         else
151             fprintf('NO\n' );
152         end
153
154 fprintf('*************************************************\n' );
155 fprintf('performing serial test:\n' );
156 for i=5:9
157  fprintf('template size: %2.0f',i);fprintf('\n' );
158  [passed P_value1 P_value2]=serialtest(outstream1,i);
159     fprintf('  -- Pvalue: %3.3f', P_value1); fprintf('  Passed: ');
160         if (passed==1)
161             fprintf('YES\n' );
162         else
163             fprintf('NO\n' );
164         end
165 end
166 fprintf('*************************************************\n' );
167 fprintf('performing approximative entropy:\n' );
168 for i=9:9
169  fprintf('seq size: %2.0f',i);fprintf('\n' );
170  tic
171  [passed P_value]=approxentropy(outstream1(1:1000000),i);%
172   toc
173  fprintf('  -- Pvalue: %3.3f', P_value); fprintf('  Passed: ');
174         if (passed==1)
175             fprintf('YES\n' );
176         else
```

```
177             fprintf('NO\n' );
178         end
179 end
```

Autres fonctions

Algorithme de Berlekamp-Massey

```
1  % Berlekamp-Massey algorithm for finding the shortest
2  % linear feed-back shift register, aka the linear complexity,
3  % of a binary sequence.
4  % Input s is a vector whose components are either 0 or 1.
5  % s represents a binary sequence of length length(s).
6  % Output f represents the shortest feedback polynomial p(x) of
7  %sequence s
8  % The i-th component in f is the coefficient of x^(i-1) in p(x).
9  % The number of components in f equals to the linear complexity
10 %plus one.
11 % The first entry in f must be equal to 1,
12 % but the last entry in f may equal 0.
13 % The returned value f satisfies the property that
14 % the entries in mod(conv(f,s),2) are all zeros except the first
15 %and last
16 % length(f)-1 entries.
17 % Output LCP is the linear complexity profile
18 % The i-th entry of LCP is the linear complexity of the first
19 % i bits of the binary sequence s.
20 function [f, LCP] = Berlekamp_Massey2(s)
21 LCP = zeros(1,length(s)); % Linear complexity profile
22 f = [1]; % feedback polynomial initialized to the constant polynomial 1
23 g = [1]; % initialized to the constant polynomial 1
24 m = 0;
25 for n = 1:length(s)
26  if mod(f * s(n:-1:(n-length(f)+1))',2) == 1 % check if discrepancy
27 % is not zero
28  % Compute the shortest feedback polynomial for the first n bits of s
29    L = max(length(f), n-length(f)+2);
30    new = mod([f zeros(1,L-length(f))] + [zeros(1,n-m) g zeros
31 (1,L-n+m-length(g))],2);
32    if L > length(f) % if the degree strictly increase
33      g = f; % record the feedback polynomial
34      m = n; % and when the increase occurs
35    end
36    f = new;
37  end
38  LCP(n) = length(f)-1;
39 end
```

Fonction de Distribution de Poisson

```
% Generate the 'phi(k)' Poisson distribution probabilites
function ppi=poissons(eta,u)
% ex:phi_0= poissions(1,0); phi_1=poissons(1,1) ...
% by now u limited to 4 !!!
% i need a formal definition of phi
    ppi=exp(-1*eta);
if u==0
    ppi=ppi*1;
    ppi=0.324652;

elseif u==1
    ppi=eta*ppi/2;
    ppi=0.182617;
elseif u==2
    ppi=eta*ppi*(eta+2)/8;
    ppi=0.142670;

elseif u==3
    ppi=eta*ppi*(eta^2/6+eta+1)/8;
    ppi=0.106645;
elseif u==4
    ppi=eta*ppi*(eta^3/24+eta^2/2+3*eta/2+1)/8;
    ppi=0.077147;
else
    ppi=0.166269;
end
% if necessary, fix just to:
% pi0=0.324652;pi1 = 0.182617;pi2 = 0.142670;
% pi3 = 0.106645; pi4 = 0.077147;pi5 = 0.166269.
```

Modele algorithmique du correcteur LFSR

```
function[c seq]=LFSR_corrector(s,t,raw)
%s=initial state of LFSR, you can choose any lenght of LFSR
%example: s=[1 1 0 0 1] for 5 bit LFSR
%t=tap positions, e.g.t=[5 2]
% c will be matrix containing the states of LFSR raw wise
% seq= generated sequence
%-------------------------------------------------------------
n=length(s);
c(1,:)=s;
m=length(t); % size of the tap mapping vector
L=length(raw);
mu=mod(L,2);
for k=1:(L-mu)/2% so this way i use the positive edges?
    %2^n-2; % do a complete cycle (to modify to perform an indep loop
```

```
% first of all, calculate the actual xor'ed value from each affected
% shift register output.
    b(1)=xor(s(t(1)), s(t(2)));
    if m>2;
        for i=1:m-2;
        b(i+1)=xor(s(t(i+2)), b(i));
        end
    end
% now xor it with the actual incoming raw bit:
in=xor(b(m-1), raw(2*k-1)); % -1 or +0 check this because of
%the edge independence of this

j=1:n-1;
s(n+1-j)=s(n-j);
s(1)=in;%b(m-1);
c(k+1,:)=s;
end
% finally seq rescues the remaining stream
seq=c(:,n)';
```

Annexe 2 : Valorisation

---

1. J. A. Aguilar Angulo, E. Kussener, H. Barthélemy, - *TRNG based on a Discrete time Chaotic Oscillator* - Workshop Phisic, Cyber-Physical Security (from smart-grids to industrial infrastructures), May 2015

2. J. A. Aguilar Angulo, E. Kussener, H. Barthélemy, B. Duval, - *Discrete Time Chaos Based Random Number Generator*- FTFC Monaco, 2014

3. E. Kussener, J.A. Aguilar Angulo, O. Mainard, D. Goguenheim, G. Oudinet, P. Salin, C. Forni, - *Implantable electrostimulation system in freely moving rodent for DBS treatment* - FTFC Monaco, 2014

4. J. A. Aguilar Angulo, E. Kussener, H. Barthélemy, B. Duval - *A New Oscillator-Based Random Number Generator* - Proc. of IEEE NEWCAS, 2012

5. J. A. Aguilar Angulo, E. Kussener, H. Barthélemy, B. Duval - *A new oscillator-based Random Number Generator* - Proc. of Faible Tension Faible Consommation (FTFC), 2012

Bibliographie

[1] T. Amaki, M. Hashimoto, and T. Onoye. An oscillator-based true random number generator with jitter amplifier. In *Circuits and Systems (ISCAS), 2011 IEEE International Symposium on*, pages 725–728, May 2011. 19, 26

[2] Takehiko Amaki, Masanori Hashimoto, and Takao Onoye. Jitter amplifier for oscillator-based true random number generator. In *ASP-DAC*, pages 81–82. IEEE, 2011. 20

[3] R. Jacob Baker. *CMOS Circuit Design, Layout, and Simulation*. Wiley-IEEE Press, 3rd edition, 2010. 68

[4] G.K. Balachandran and R.E. Barnett. A 440 na true random number generator for passive rfid tags. *Circuits and Systems I : Regular Papers, IEEE Transactions on*, 55(11) :3723–3732, Dec 2008. 19, 26

[5] M Barú, O de Oliveira, and F Silveira. A 2v rail-to-rail micropower cmos comparator. 1996. 90

[6] A. Beirami. Zigzag map : a variability-aware discrete-time chaotic-map truly random number generator. *Electronics Letters*, 48 :1537–1538(1), November 2012. 26, 83

[7] Bernhard E. Boser. Analog design usign gm/id and ft metrics. university of berkeley, 2011. 61

[8] Wei Chen, Wenyi Che, Zhongyu Bi, Jing Wang, Na Yan, Xi Tan, Junyu Wang, Hao Min, and Jie Tan. A 1.04 uw truly random number generator for gen2 rfid tag. In *Solid-State Circuits Conference, 2009. A-SSCC 2009. IEEE Asian*, pages 117–120, Nov 2009. 19, 26

[9] J. Chengxin Liu, McNeill. A digital-pll-based true random number generator, 2005. 19

[10] O. Chrysaphinou and S. Papastavridis. A limit theorem on the number of overlapping appearances of a pattern in a sequence of independent trials. *Probability Theory and Related Fields*, 79(1) :129–143, 1988. 111

[11] H.Oguey ; D.Aebischer. Cmos current reference without resistance. *IEEE Journal of Solid-State Circuits*, 32(7) :1132–5, july 1997. 71

[12] Klaus Finkenzeller. *RFID Handbook*. John Wiley and Sons, 2003.

[13] Daniel Genkin, Adi Shamir, and Eran Tromer. Rsa key extraction via low-bandwidth acoustic cryptanalysis. Cryptology ePrint Archive, Report 2013/857, 2013. `http://eprint.iacr.org/`. 12

[14] Clemens Helfmeier, Christian Boit, Dmitry Nedospasov, and Jean-Pierre Seifert. Cloning physically unclonable functions. In *HOST*, pages 1–6. IEEE, 2013. 23

[15] Morris W. Hirsch, Stephen Smale, and Robert Devaney. *Differential Equations, Dynamical Systems, and an Introduction to Chaos*. Academic Press, 2nd edition, 2004.

[16] J. Holleman, B. Otis, S. Bridges, A. Mitros, and C. Diorio. A 2.92 uw hardware random number generator. In *Solid-State Circuits Conference, 2006. ESSCIRC 2006. Proceedings of the 32nd European*, pages 134–137, Sept 2006. 26

[17] Paul Jespers. *The Gm/ID Methodology, a Sizing Tool for Low-voltage Analog CMOS Circuits : The Semi-empirical and Compact Model Approaches*. Springer Publishing Company, Incorporated, 1st edition, 2009. 61

[18] M. Jessa and L. Matuszewski. Enhancing the randomness of a combined true random number generator based on the ring oscillator sampling method. In *Reconfigurable Computing and FPGAs (ReConFig), 2011 International Conference on*, pages 274–279, Nov 2011. 21

[19] Song ju Kim, Ken Umeno, and Akio Hasegawa. Corrections of the nist statistical test suite for randomness ?, cryptology eprint archive, report 2004/018. 114

[20] V. D. Juncu, M. Rafiei-Naeini, and P. Dudek. Integrated circuit implementation of a compact discrete-time chaos generator. *Analog Integr. Circuits Signal Process.*, 46(3) :275–280, March 2006. 83

[21] Donald E. Knuth. *The Art of Computer Programming, Volume 2 (3rd Ed.) : Seminumerical Algorithms*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1997. 104

[22] Sandeep S. Kumar, Jorge Guajardo, Roel Maes, Geert Jan Schrijen, and Pim Tuyls. The butterfly puf : Protecting ip on every fpga. In Mohammad Tehranipoor and Jim Plusquellic, editors, *HOST*, pages 67–70. IEEE Computer Society, 2008. 22

[23] Siew-Hwee Kwok, Yen-Ling Ee, Guanhan Chew, Kanghong Zheng, Khoongming Khoo, and Chik How Tan. A comparison of post-processing techniques for biased random number generators. In Claudio Agostino Ardagna and Jianying Zhou, editors, *WISTP*, volume 6633 of *Lecture Notes in Computer Science*, pages 175–190. Springer, 2011.

[24] Patrick Lacharme. Fast software encryption. chapter Post-Processing Functions for a Biased Physical Random Number Generator, pages 334–342. Springer-Verlag, Berlin, Heidelberg, 2008.

[25] Edward Norton Lorenz. Deterministic nonperiodic flow. *Journal of the Atmospheric Sciences*, 20 :130 – 141, 1963 1963. 99

[26] Soumyajit Mandal, Scott K. Arfin, and Rahul Sarpeshkar. Fast startup cmos current references. In *ISCAS*. IEEE, 2006.

[27] Samuel Manen. Le design analogique en technologie cmos. Ecole de Microélectronique Fréjus, 2011. 61

[28] George Marsaglia. The Marsaglia Random Number CDROM including the Diehard Battery of Tests of Randomness. http ://www.stat.fsu.edu/pub/diehard/, 1995. 104

[29] Shapour Mohammadi. LYAPROSEN : MATLAB function to calculate Lyapunov exponent. Statistical Software Components, Boston College Department of Economics, July 2009. 80

[30] Shapour Mohammadi. LYAPROSEN : MATLAB function to calculate Lyapunov exponent. Statistical Software Components, Boston College Department of Economics, July 2009.

[31] Sergey Morozov, Abhranil Maiti, and Patrick Schaumont. An analysis of delay based puf implementations on fpga. In *Proceedings of the 6th International Conference on Reconfigurable Computing : Architectures, Tools and Applications*, ARC'10, pages 382–387, Berlin, Heidelberg, 2010. Springer-Verlag.

[32] Boris Murmann. ft and intrinsic gain gm/id design methodology, howpublished = university of stanford. EE214 Course notes. Lecture 5, year = 2004,. 61

[33] Zhou S. ; Zhang W. ; Wu N. An ultra-low power cmos random number generator. *Solid-State Electronics.*, 52(2) :223–238, May 2007. 19, 26

[34] T. Nakura, M. Ikeda, and K. Asada. Ring oscillator based random number generator utilizing wake-up time uncertainty. In *Solid-State Circuits Conference, 2009. A-SSCC 2009. IEEE Asian*, pages 121–124, Nov 2009. 26

[35] B. A. M. Owens, M. T. Stahl, N. J. Corron, J. N. Blakely, and L. Illing. Exactly solvable chaos in an electromechanical oscillator. *Chaos*, 23(3) :033109, September 2013. 83

[36] C.S. Petrie and J. Alvin Connelly. Modeling and simulation of oscillator-based random number generators. In *Circuits and Systems, 1996. ISCAS '96., Connecting the World., 1996 IEEE International Symposium on*, volume 4, pages 324–327 vol.4, May 1996. 19

[37] C.S. Petrie and J.A. Connelly. A noise-based ic random number generator for applications in cryptography. *Circuits and Systems I : Fundamental Theory and Applications, IEEE Transactions on*, 47(5) :615–621, May 2000. 26

[38] Michael T. Rosenstein, James J. Collins, and Carlo J. De Luca. A practical method for calculating largest lyapunov exponents from small data sets. *Phys. D*, 65(1-2) :117–134, May 1993. 42

[39] Ulrich Rührmair, Frank Sehnke, Jan Sölter, Gideon Dror, Srinivas Devadas, and Jürgen Schmidhuber. Modeling attacks on physical unclonable functions. In *Proceedings of the 17th ACM Conference on Computer and Communications Security*, CCS '10, pages 237–249, New York, NY, USA, 2010. ACM. 23

[40] Andrew Rukhin, Juan Soto, James Nechvatal, Elaine Barker, Stefan Leigh, Mark Levenson, David Banks, Alan Heckert, James Dray, San Vo, Andrew Rukhin, Juan Soto, Miles Smid, Stefan Leigh, Mark Vangel, Alan Heckert, James Dray, and Lawrence E Bassham Iii. A statistical test suite for random and pseudorandom number generators for cryptographic applications, 2001. 105

[41] Ali Sadr and Mostafa Zolfaghari-Nejad. Physical unclonable function (puf) based random number generator. *CoRR*, abs/1204.2516, 2012. 20

[42] Jens Spars and Steve Furber. *Principles of Asynchronous Circuit Design : A Systems Perspective*. Springer Publishing Company, Incorporated, 1st edition, 2010.

[43] N. Stefanou and S.R. Sonkusale. High speed array of oscillator-based truly binary random number generators. In *Circuits and Systems, 2004. ISCAS '04. Proceedings of the 2004 International Symposium on*, volume 1, pages I–505–8 Vol.1, May 2004. 19

[44] G. S. Jovanovic ;M. K. Stojcev. A method for improvement stability of a cmos voltage controlled ring oscillator. *ICEST.*, 2 :715–718, 2007. 56

[45] J. Jakovenko V. Kote, V. Molata. A noise-based ic random number generator for applications in cryptography. *Electroscope. Západoceská univerzita v Plzni, Fakulta elektrotechnická*, 6, 2012. 16

[46] S. Ozoguz S. Kilinc A. Toker A. Zeki V. Tavas, A. S. Demirkol. An ic random number generator based on chaos. *Electroscope. Západoceská univerzita v Plzni, Fakulta elektrotechnická*, 2, 2010. 24, 26

[47] Ihor Vasyltsov, Eduard Hambardzumyan, Young-Sik Kim, and Bohdan Karpinskyy. Fast digital trng based on metastable ring oscillator. In Elisabeth Oswald and Pankaj Rohatgi, editors, *CHES*, volume 5154 of *Lecture Notes in Computer Science*, pages 164–180. Springer, 2008. 22

[48] Sang-Kyung Yoo, Deniz Karakoyunlu, Berk Birand, and Berk Sunar. Improving the robustness of ring oscillator trngs. *ACM Trans. Reconfigurable Technol. Syst.*, 3(2) :9 :1–9 :30, May 2010. 20

Index

Julio Alexander AGUILAR ANGULO
Laboratoire IM2NP, Equipe CCI
Conception d'un Générateur de Valeurs aléatoires sur la Technologie CMOS AMS 0.35μm

Résumé en Français :
Les générateurs de suites binaires aléatoires constituent la partie primordiale d'un système cryptographique. La vitesse , la qualité des suites générées , la sécurité et la consommation jouent un rôle essentiel dans le choix d'un générateur. La sécurité du système cryptographique augmente si un tel système peut être réalisé dans un seul circuit.
Le travail de recherche développé consiste donc en la réalisation d'un générateur de nombres aléatoires fonctionnant en basse consommation, basse vitesse. Le circuit proposé est de type analogique et valide l'ensemble des tests NIST assurant le caractère du signal.
Une réalisation sur Silicium en technologie 0,35μm a été implémentée et validée via les tests NIST développés sous Matlab. De ce travail de thèse, un certain nombre de publications ont montré la plue value recherche des résultats .
Mots clés : CMOS, TRNG, PTAT, Basse consommation, Basse fréquence.

Résumé en Anglais :
Random binary sequences generators constitute the essential part of a system Cryptographic. The speed, quality of generated suites, safety and consumption play an essential role in the selection of a generator. The security of the cryptographic system increases if such a system can be realized in a single circuit.
The developed research work consists in the realization of a random number generator running in low power, low speed. The proposed circuit is analog and Valid all NIST tests ensuring the randomness of a signal.
A realization on silicon in 0,35μm technology has been implemented and validated through NIST developed tests Matlab. In this thesis, a number of publications have demonstrated the added value Plue search results.
Keywords : CMOS, TRNG, Low consumption, Low voltage