RÉPUBLIQUE DU CAMEROUN
Paix-Travail-Patrie
- - - - - - - - -
UNIVERSITÉ DE YAOUNDÉ 1
- - - - - - - - -
FACULTÉ DES SCIENCES
CENTRE DE RECHERCHE ET DE
EN SCIENCES, TECHNOLOGIES ET
GEOSCIENCES
- - - - - - - - -
UNITÉ DE RECHERCHE ET
FORMATION DOCTORALE SCIENCE
PHYSIQUES ET APPLICATIONS

REPUBLIC OF CAMEROON
Peace-Work-Fatherland
- - - - - - - - -
UNIVERSITY OF YAOUNDE 1
- - - - - - - - -
FACULTY OF SCIENCES
POSTGRADUATE SCHOOL OF
SCIENCES, TECHNOLOGYS AND
GEOSCIENCES
- - - - - - - - -
RESEARCH AND POSTGRADUATE
TRAINING UNIT IN PHYSICS AND
APPLICATIONS

DÉPARTEMENT DE PHYSIQUE
DEPARTEMENT OF PHYSICS
LABORATOIRE D'ÉNERGIE ET SYSTÈMES ÉLECTRIQUES ET ÉLECTRONIQUES
LABORATORY OF ENERGY AND ELECTRICALS AND ELECTRONICS SYSTEMS

THÈME

IMPLÉMENTATION D'UNE RÉCURRENCE DU CHAT D'ARNOLD MONOBIT À GRANDE PÉRIODE EN DIMENSION 8

THÈSE

présentée en vue de l'obtention du diplôme de **DOCTORAT/Ph.D de physique**

Spécialité : **Énergie et systèmes électriques et électroniques**

Par :

DJEUGOUE NZEUGA Hermann
Matricule : **12U0332**
Master en physique

Devant le jury composé de :

Président :	ESSIMBI ZOBO Bernard ,	Pr ;	Université de Yaoundé I ;
Rapporteur :	EYEBE FOUDA Jean Sire Armand,	MC ;	Université de Yaoundé I ;
Membres :	MBINACK CLément,	MC ;	Université de Yaoundé I ;
	MELINGUI Achille,	MC ;	Université de Yaoundé I ;
	TSAFACK Pierre,	MC ;	Université de Buea ;
	BODO Bertrand,	MC ;	Université de Yaoundé I.

***Année* : 6, Octobre 2023**

UNIVERSITE DE YAOUNDE I
UNIVERSITY OF YAOUNDE I

FACULTE DES SCIENCES
FACULTY OF SCIENCE

DEPARTEMENT DE PHYSIQUE
DEPARTMENT OF PHYSICS

ATTESTATION DE CORRECTION DE LA THESE DE DOCTORAT/Ph.D

Nous, Professeurs **MBINACK Clément, BODO Bertrand, MELINGUI Achille, TSAFACK Pierre** et Professeur **ESSIMBI ZOBO Bernard**, respectivement Examinateurs et Président du jury de la thèse de Doctorat/Ph.D de Monsieur **DJEUGOUE NZEUGA Hermann** Matricule **12U0332**, préparée sous la direction du Professeur **EYEBE FOUDA Jean Sire Armand**, intitulée : « **Implémentation d'une récurrence du chat d'Arnold monobit à grande période en dimension 8** », soutenue le vendredi **, 6 octobre 2023**, en vue de l'obtention du grade de Docteur/Ph.D en Physique, Spécialité **Energie et Systèmes Electriques et Electroniques**, attestons que toutes les corrections demandées par le Jury de soutenance ont été effectuées.

En foi de quoi, la présente attestation lui est délivrée pour servir et valoir ce que de droit.

Fait à Yaoundé le .....02 NOV 2023.....

Le Président du Jury

Examinateur

MBINACK Clément, MC, Uy1

ESSIMBI ZOBO Bernard,
Pr, Uy1

UNIVERSITE DE YAOUNDE I * FACULTE DES SCIENCES * DEPARTEMENT DE PHYSIQUE
Le Chef de Département

Le Chef de Département de Physique

Jean-Marie Bienvenu
Professeur

Table des matières

Table des matières iv

Liste des tableaux v

Liste des figures viii

Dédicace ix

REMERCIEMENTS x

LISTE DES ABRÉVIATIONS xii

RÉSUMÉ xv

ABSTRACT xvi

INTRODUCTION GÉNÉRALE 1

Chapitre 1: REVUE DE LA LITTÉRATURE 6

1.1 Introduction 6

1.2 Généralités sur les générateurs de nombres pseudo-aléatoires 6

1.3 Les générateurs de nombres pseudo-aléatoires linéaires 10

1.3.1 Les générateurs congruentiels linéaires (GCL) 11

1.3.2 Registre à décalage de rétroaction linéaire 13

1.3.3 Les générateurs optimisés à table de correspondance 14

1.3.4 GFSR, TGFSR, Mersenne Twister 16

1.4 Les générateurs de nombres pseudo-aléatoires non linéaires 18

1.4.1 Les GNPAs chaotiques différentiels 18

1.4.2 Les GNPAs chaotiques récurrentiels 20

1.4.3 Les GNPAs basés sur des itérations chaotiques 21

1.5 Systèmes dynamiques 22

1.5.1 Système dynamique chaotique continue 23

1.5.2 Système dynamique chaotique discret 26

1.5.3 Critères d'évaluation d'une fonction Chaotique . . . 27
1.6 Conclusion . . . 28
Chapitre 2: MATÉRIELS ET MÉTHODE **29**
2.1 Introduction . . . 29
2.2 Matériels : Les outils de caractérisation d'aléa . . . 29
2.2.1 L'exposant de Lyapunov . . . 30
2.2.2 La sensibilité aux conditions initiales . . . 31
2.2.3 L'entropie . . . 32
2.2.4 La corrélation . . . 32
2.2.5 Le test statistique de NIST SP800-22 . . . 34
2.3 Méthodes . . . 37
2.3.1 Les généralités sur la récurrence du chat d'Arnold (RCA) . . . 37
2.3.2 La récurrence du chat d'Arnold linéaire par morceau 2D sur 4 bits . . . 42
2.3.3 La récurrence du chat d'Arnold linéaire par morceau monobit en 8D . . . 55
2.4 Conclusion . . . 63
Chapitre 3: RÉSULTATS ET DISCUSSION **64**
3.1 Introduction . . . 64
3.2 L'implémentation des différents PWLCM sur FPGA . . . 64
3.3 La mise en réseau de PWLCM monobit en dimension 8 . . . 68
3.4 L'implémentation FPGA du GNPA-PWLCM . . . 69
3.5 L'analyse de l'aléa du générateur de nombres pseudo-aléatoires proposé . . . 72
3.5.1 Le test statistique de NIST . . . 72
3.5.2 L'espace des clés de chiffrement . . . 72
3.5.3 Le portrait de phase . . . 73
3.5.4 La sensibilité à la clé de chiffrement . . . 74
3.5.5 L'entropie des informations . . . 76
3.5.6 L'histogramme . . . 77
3.5.7 La corrélation . . . 77
3.6 Le procédé de validation du générateur . . . 79
3.6.1 La validation des résultats théoriques . . . 79

3.6.2 La validation des résultats expérimentaux 80
3.7 La comparaison du GNPA proposé avec la littérature 83
3.8 Conclusion 84
CONCLUSION GÉNÉRALE **86**

Liste des tableaux

TABLE 1.1 Exemple de systèmes chaotiques à temps discret. 26

TABLE 2.1 Matrice du plus grand exposant de Lyapunov individuel des différentes conditions initiales du PWLCM pour $n = 2$, $\alpha = \beta = 1$, $M = N = 2$, $c_1 = 0, c_2 = 3$, $d_1 = 3, d_2 = 5$, $a_1 = 1$, $a_2 = 1$, $b_1 = 0, b_2 = 2$, $\lambda_0(3,3) = 2,3895$ . . 48

TABLE 2.2 Matrice du plus grand exposant de Lyapunov individuel des différentes conditions initiales du PWLCM pour $n = 2$, $\alpha = \beta = 1$, $M = N = 2$, $c_1 = c_2 = 0$, $d_1 = d_2 = 0$, $a_1 = 0$, $a_2 = 1$, $b_1 = b_2 = 0$, $\lambda_0(3,3) = 2,3895$ 48

TABLE 2.3 Matrice du plus grand exposant de Lyapunov individuel des différentes conditions initiales du PWLCM pour $n = 2$, $\alpha = \beta = 1$, $M = N = 2$, $c_1 = 0$, $c_2 = 3$, $d_1 = d_2 = 0$, $a_1 = 0$, $a_2 = 1$, $b_1 = b_2 = 0$, $\lambda_0(3,3) = 2,3895$ 48

TABLE 2.4 Sensibilité de la période de la Π_{PWLCM} en fonction de a_i et b_i pour $n = 3$, $M = N = 1$, $\alpha = \beta = 1$, $c_1 = 3$, $d_1 = 5$ et différentes valeurs de (a_1, b_1) 50

TABLE 2.5 Sensibilité de la période de la Π_{PWLCM} en fonction de a_i et b_i pour $n = 3$, $M = N = 1$, $\alpha = \beta = 1$, $c_1 = 3$, $d_1 = 5$ et différentes valeurs de (c_1, d_1) 50

TABLE 2.6 Dépendance de la période à la précision pour $a_1 = a_2 = 0, c_1 = 0, c_2 = 11$ et $\alpha' = 3$, $\beta' = 1$ 52

TABLE 2.7 Les résultats du test de NIST 800-22 : 55

TABLE 3.1 Comparaison des ressources utilisées lors de la mise en œuvre de PWLCM $2D$ sur $4-$bits, PWLCM $4D$ sur $2-$bits et de PWLCM $8D$ sur $1-$bit, chaque système produit une sortie 8 bits. 68

TABLE 3.2 Description des signaux intervenant au module GNPA-PWLCM. 71

TABLE 3.3 Les Caractéristiques du GNPA-PWLCM sur FPGA. Avec $N = 64$-bits interne, $w = 32$ et $w = 64$ bits en sortie. 71

TABLE 3.4 Résultat des tests de NIST des séquences produites par le GNPA-PWLCM 32 et 64 bits. 73

TABLE 3.5 Comparaison de la conception proposée avec d'autres dans la littérature. . 83

TABLE 3.6 Les caractéristiques FPGA des générateurs [78] et GNPA-PWLCM 84

Table des figures

FIGURE 1.1 Les différentes classes des générateurs de nombres aléatoires. 7
FIGURE 1.2 Architecture générale d'un vrai générateur de nombres pseudo-aléatoires 7
FIGURE 1.3 Architecture générale d'un générateur de nombres pseudo-aléatoires . . . 8
FIGURE 1.4 Architecture générale d'un FPGA 9
FIGURE 1.5 Un générateur de registre à décalage à rétroaction linéaire 4 bits avec un polynôme de rétroaction $a_0X_4 + a_1X_3 + a_2X_2 + a_3X_1 + a_4$, avec $(a_0 = a_4 = 1)$. 14
FIGURE 1.6 Registre à décalage basé sur LUT et GNPA optimisé FIFO FPGA : (a) La carte de chaque ligne de la matrice de récurrence comme une porte XOR en utilisant LUT-FF, (b) utilise la mémoire de bloc RAM comme $k \cdot k$ FIFO pour stocker les séquences récursives, (c) charge l'état dans le registre à décalage SR basé sur FIFO au lieu de BRAM, (d) cascade de n'importe quel nombre de Xilinx SRL32 pour créer un k-bit SR 15
FIGURE 1.7 Architecture de registre à décalage torsadé à rétroaction généralisée : à chaque opération de récurrence t, il calcule $x(t + N)$ grâce aux trois mots $x(t)$, $x(t + 1)$, et $x(t + m)$ et génère la sortie avec la fonction de temporisation. 16
FIGURE 1.8 Implémentation de la récurrence linéaire (L.R) pour "Mersenne Twister" GNPA en utilisant le tampon circulaire des registres (L.R est la récurrence linéaire de la fonction de transfert de MT) 18
FIGURE 1.9 L'attracteur du système de Loretz pour $r = 24,75$ 24
FIGURE 1.10 Circuit de Chua 25
FIGURE 1.11 L'attracteur du circuit de Chua $c_1 = 9$ 25

FIGURE 2.1 Récurrence de la carte du chat d'Arnold 37
FIGURE 2.2 Portrait de phase de la récurrence du chat D'Arnold pour $\alpha = \beta = 1$, $x(0) = 0$, $y(0) = 0$ et $m = 1$ 38
FIGURE 2.3 Portrait de phase de la récurrence du chat D'Arnold discret pour $\alpha = \beta = 1$, $x(0) = 1$, $y(0) = 5$ et $m = 1$ 41
FIGURE 2.4 L'évolution de dynamique de la récurrence du chat d'Arnold conventionnelle (a) et de la PWLCM (b) dans un espace de phase de précision 4-bits 45
FIGURE 2.5 Le portrait de phase de la PWLCM dans un espace de phase codé sur $n = 4$-bits 45

FIGURE 2.6 Dynamique des exposants de Lyapunov pour les conditions initiales respective $z_0 = 2^n x_0 + y_0$, pour $n = 4$ et a_i, b_i, c_i et d_i suivant celle du tableau 2.3 (réglage1),du tableau 2.1 (réglage3) et du tableau 2.2 (réglage2). Les périodes correspondent respectivement à $T_1 = 9320$, $T_3 = 2520$, $T_2 = 12$. Les exposants de Lyapunov sont évalues après 1000 itérations de PWLCM 48

FIGURE 2.7 Dynamique des exposants de Lyapunov pour les conditions initiales respective $z_0 = 2^n x_0 + y_0$, pour $n = 4$ et a_i, b_i, c_i et d_i suivant celle du tableau 2.3 (réglage1),du tableau 2.1 (réglage3) et du tableau 2.2 (réglage2). les périodes correspondent respectivement à $T_1 = 1,51 \cdot 10^{15}$, $T_3 = 1,02 \cdot 10^{10}$, $T_2 = 24$. Les exposants de Lyapunov sont évalues après 1000 itérations de PWLCM 49

FIGURE 2.8 Représentation générale de l'orbite d'un système chaotique numérique . . 51

FIGURE 2.9 Distribution des périodes des orbites de la PWLCM $T(x_0, y_0)$ pour chaque condition initiale avec $a_1 = 0, a_2 = 0, c_1 = 0, c_2 = 11$ et $\alpha' = 3$, $\beta' = 1$. 53

FIGURE 2.10 Distribution des probabilités de la PWLCM $T(x_0, y_0)$ pour chaque conditions initiales avec $a_1 = 0, a_2 = 0, c_1 = 0, c_2 = 11$ et $\alpha' = 3$, $\beta' = 1$ 53

FIGURE 2.11 Mixage des images à l'aide des transformations RCA discret et PWLCM. La première ligne montre les résultats du RCA discret, la deuxième ligne représente les résultats du PWLCM, tandis que la troisième ligne montre l'image inversée obtenue à partir du PWLCM inversé. 54

FIGURE 2.12 Niveau de Complexité d'une opération d'addition en fonction de la précision. (a) est une addition monobit , elle contient 1 porte logique XOR, (b) est une addition 2 bits, elle contient 3 portes logiques XOR, 3 portes logiques AND et une porte logiques OR. 57

FIGURE 2.13 Evaluation périodique pour 100 combinaison différents des paramètres de contrôle des systèmes dynamiques 8D et 2D, proposé avec $n = 1$ 60

FIGURE 2.14 Représentation graphique des 8 Exposants de Lyapunov de la PWLCM 8D ($\Lambda_{1,2,3,4,5,6,7,8}(z0)$) en fonction des conditions initiales $z0 = x_1(0) + x_2(0) \cdot 2 + x_3(0) \cdot 2^2 + x_4(0) \cdot 2^3 + x_5(0) \cdot 2^4 + x_6(0) \cdot 2^5 + x_7(0) \cdot 2^6 + x_8(0) \cdot 2^7$, pour $n = 1$ et un vecteur de paramètres de contrôles aléatoires $\mathbf{a}_{i,k}$ et $\mathbf{c}_{i,k}$ 63

FIGURE 3.1 Architecture des termes d'addition modulaire (P_k). 65

FIGURE 3.2 Architecture générale du module d'évaluation de chaque variable d'état x_i. 66

FIGURE 3.3 Architecture générale de la PWLCM. 66

FIGURE 3.4 Implémentation FPGA généralisée de PWLCM n-bit. La commande "LOAD" permet de fixer $x(0)$ comme condition initiale, a et c comme paramètres de contrôles dans le registre à $kd(2n + 1) + nd$-bits ; "RESET" permet d'effacer le registre de $kd(2n + 1) + nd$-bits ; et la commande "SW" permet de charger les conditions initiales dans les registres à nd-bits **FF1**, donc de démarrer le système. 67

FIGURE 3.5 Le réseau de PWLCM monobit en dimension 8 : La structure interne du PRNG-PWLCM pour $N = 64$ 70

FIGURE 3.6 La structure Général du GNPA-PWLCM N bits sur le FPGA 71
FIGURE 3.7 Les portraits de phase entre (x_a, x_b) ;(x_c, x_d) ;(x_e, x_f) ;(x_g, x_h) ; 74
FIGURE 3.8 Les formes d'onde traduisant la sensibilité du générateur : (a) aux conditions initiales , (b) à la clé du générateur 75
FIGURE 3.9 Le Taux de changement séquence 76
FIGURE 3.10 Entropie de l'information de 50 séquences. 77
FIGURE 3.11 Distribution des valeurs dans les séquences aléatoires x_a, x_b, x_c, x_d, x_e, x_f, x_g. 78
FIGURE 3.12 Analyse de corrélation des séquences pour GNPA-PWLCM :(**a**) Auto-corrélation, (**b**) Cross-corrélation. 78
FIGURE 3.13 Résultats de la simulation GNPA-PWLCM sur Vivado. 79
FIGURE 3.14 Résultats de la simulation GNPA-PWLCM sur MATLAB. 79
FIGURE 3.15 Diagramme du Pmod Header. 80
FIGURE 3.16 Carte de prototypage FPGA ZYBO 7020. 81
FIGURE 3.17 Dispositif expérimentale. 81
FIGURE 3.18 L'évolution temporelle des sorties $x_a(t)$ et $x_b(t)$ capturée sur l'oxilloscope numérique. 82
FIGURE 3.19 Portrait de phase formé par les séquences x_a et x_b. 82
FIGURE 3.20 Dispositif de prototypage FPGA : Zybo Z7 90
FIGURE 3.21 Logiciel vivado 2018.3 91
FIGURE 3.22 Interface du logiciel vivado 2018.3 91
FIGURE 3.23 Interface du logiciel MATLAB 93

Dédicace

Á

la famille NZEUGA

Remerciements

Une thèse de doctorat est le fruit d'un travail collectif. Pour cela, je tiens à remercier une personne pour son encadrement, sa disponibilité et son amitié : Professeur **EYEBE FOUDA Jean Sire Armand**, mon directeur de thèse. Il a pu, malgré son emploi de temps bien chargé, être toujours présent à mes côtés et me faire profiter de son expérience, son intelligence et ses connaissances des outils en ce qui concerne ma recherche. Les travaux que j'ai pu mener et ce document ne seraient pas ce qu'ils sont sans sa motivation et ses encouragements, sa patience, son recul, son regard critique, son écoute, et la pertinence de ses conseils. Ce fut un grand plaisir de travailler avec lui, et j'espère pouvoir continuer à le faire.

Je tiens également à remercier les membres de mon jury de thèse : Monsieur ESSIMBI ZOBO Bernard, Professeur à l'université de Yaoundé 1, Monsieur MBINACK Clément, Maître de conférences à l'université de Yaoundé 1, Monsieur MELINGUI Achille , Maître de conférences à l'université de Yaoundé 1, Monsieur TSAFACK Pierre, Maître de conférences à l'université de Buea et Monsieur BODO Bernard, Maître de conférences à l'université de Yaoundé 1,ainsi que les membres de mon jury de présoutenance, d'audition et doctorial pour avoir accepté d'examiner ce travail. Merci pour leurs suggestions et leurs précieux conseils, qui ont permis de clarifier et d'améliorer cette thèse.

J'exprime également mes plus profonds remerciements au Recteur de l'Université de Yaoundé 1, à travers le centre de recherche et de formation doctoral en science technologies et géosciences de m'avoir sélectionné en thèse.

Je souhaite exprimer ma gratitude à l'université de Yaoundé 1. Spécialement aux personnels du département de physique et en particulier au Professeur NDJAKA Jean Marie Bienvenu le chef du département de physique, Professeur ESSIMBI ZOBO Bernard le chef du laboratoire d'électronique, Professeur KOFANE Timoléon Crépin, Professeur WOAFO Paul, Professeur TCHAWOUA Clément, Professeur BEN-BOLIE Germain Hubert, Professeur BIYA MOTTO Fédéric , Professeur MBINACK Clément, Professeur VONDOU DEBERTINI Appolinaire, Professeur ZEKENG Serge Sylvain, professeur DJUIDJE KENMOE Germaine, qui m'ont grandement aider à façonner mon esprit scientifique tout au long de ce parcours.

Mes remerciements particuliers au Docteur AYISSI EYEBE Guy pour les discussions fructueuses et les encouragements pendant les séminaires à l'unité de prototypage du laboratoire d'électronique de l'université de Yaoundé 1.

Je suis également reconnaissant envers le Laboratoire d'énergie, systèmes électriques et électroniques pour les discussions utiles, les séminaires et l'ambiance amicale que nous avons eu tout au long de ce travail. Un merci Spéciale au Docteur TAGNE Samuel, Docteur GNYAMSI Gaëtan, M. PANCHA Hertz, M. DJEUFA DAGOUMEGEI Guy Morgand, M. EMAKOUA Hermann, Mm. CHOUAMENI Claire, Mm. WANDJA Guillène Martiale, mes camarades de promotions et tous les cadets académiques pour la sincère collaboration, la solidarité, l'esprit d'équipe et les débats passionnants qui nous ont mutuellement enrichis.

Je ne remercierai jamais assez mes parents, mes frères et sœurs (NGAMENI NZEUGA Roséline, KAMENI NZEUGA Raoul, NAMENI NZEUGA Idriss, TCHEUMENI NZEUGA Fabiola, NTCHUISSE NZEUGA Ange Lucresse), pour avoir toujours été présents, m'avoir toujours aidé et soutenu, et pas seulement durant mes années d'études. Sans leur gentillesse, encouragement et dévouement, je n'en serais pas là.

Je ne saurai finir sans remercier, la famille TCHEUTCHOUA et l'ensemble du corps enseignant du collège polyvalent Saint Augustin et aussi de toutes les autres structures oú j'ai été encadreur, pour leur soutien durant cette période de recherche.

Finalement, je voudrais exprimer mes remerciements à tous ceux qui n'ont pas été mentionnés et qui, par leur dur labeur et leur gentillesse, ont rendu cette thèse possible.

LISTE DES ABRÉVIATIONS

ADC : Analog-Digital Converter

ASIC : Application-Specific Integrated Circuit

ASG : Alternating Step Generator

AXI : Advanced eXtensible Interface

BRAM : Block Random Access Memory

CAN : convertisseur Analogique-Numérique

CPRNG : Chaotique Pseudo Random Number Generator

CLK : Clock

CLCG : Couple Linear Congruential Generator

CLB : Configurable Logic Block

CIPRNG : Chaotic iteration pseudo random number generator

DAC : Digital-Analog Converter

DFT : Discrete Fourier Transform

DSP : Digital Signal Processor

DACM : Discret Arnold Cat Map

DAM : Digital Asset Management

DRNG : Deterministe Random Number Generator

FF : Flip-Flop

FPGA : Field Programmable Gate Arrays

GCL : Générateur Congruentiel Linéaire

GNA : Générateur de nombres aléatoires

GNPA : Générateur de nombres pseudo-aléatoires

GNPD : Générateur de nombres pseudo-aléatoires deterministe

GNPN : Générateur de nombres pseudo-aléatoires non deterministe

GNPA-PWLCM : Générateur de nombres pseudo-aléatoires-Piece Wise Linear Cat Map

HFWMS : hyperchaotic four-wing memristive system

I2C : Inter Integrated Circuit Bus

IoTs : Internet des objets

I/O : Input/Output

JTAG : Joint Test Action Group

LCA : Logic Cell Array

LFSR : linear feedback shift register

LUT : Look Up Table

LPRNG : Linear Pseudo Random Number Generator

MCG : Multiplicative Linear Gongruential

MWC Multiply with Carry

MATLAB : Matrix Laboratry

NRNG : Non-linear Random Number Generator

NIST : National Institute of Standards and Technology

NG : negation

PWLCM : Piece-Wise Linear Cat Map

QACM : Quantum Arnold Cat Map

RAM : Random Access Memory

RCA : Récurrence du chat d'Arnold

SG : Shrinking Generator

SD : Secure Digital

SR : Shift Register

SDIO : Secure Digital Input Output

SPI : Serial Peripheral Interface

TRNG : True Random Number Generator

USB : Universal Serial Bus

UART : Universal Asynchronous Receiver Transmitter

VHDL : VHSIC Hardware Description Language

WNS : Worst Negative Slack

XOR : eXclusive-OR

RÉSUMÉ

Dans cette thèse, nous avons mis en œuvre une approche simple pour générer des entiers pseudo-aléatoires sur un octet, utile pour l'implémentation des simulations numériques, des jeux de hasard et surtout pour les applications de cryptographie. Pour cela, nous avons considéré un ensemble de N particules dont la dynamique individuelle est modélisée par des équations linéaires discrètes dans un espace d'état fini. Ces équations ont été obtenues en introduisant des termes d'addition modulaire dans la récurrence du chat d'Arnold (RCA) conventionnelle. De plus, en considérant une interaction entre les N particules, nous avons obtenu un comportement de l'ensemble du réseau sensible aux conditions initiales, et dont la période peut être infiniment augmentée. Le système obtenu possède une grande période et peut être utilisé comme un générateur de nombres pseudo-aléatoires (GNPA) numérique. Cette approche de conception des entiers pseudo-aléatoires permet de résoudre le problème de la dégradation dynamique observée lors de la conception numérique des GNPA basée sur les systèmes chaotiques. Ce travail c'est terminé par une implémentation expérimentale FPGA du réseau dynamique proposé dans espace fini de précision mono-bit en utilisant le langage Vérilog sur la plateforme de programmation Vivado 2018.3. La fréquence maximum des opérations obtenue est de $177,809$ MHz avec un débit de $11,37$ Gbit/s utilisant seulement 0.17% (90) de table de correspondance (LUT, Look up table en anglais), 0.06% (65) bascules (FF). Les résultats des analyses statistiques des séquences aléatoires génèrent montrent que le système proposé peut être utiliser pour diverses applications numériques. Plus précisément pour l'implémentation des systèmes de cryptographie sécurisées en temps réel dans les dispositifs aux ressources matérielles limité tels que les systèmes IoTs.

Mots clés : Chaos discret, récurrence du chat d'Arnold, arithmétique modulaire, espace fini, nombres pseudo-aléatoires ,FPGA, cryptographie, IoTs.

ABSTRACT

In this thesis, we have implemented a simple approach to generate pseudo-random integers on a byte, useful for implementing numerical simulations, gambling games, and especially for cryptographic applications. For this purpose, we considered a set of N particles whose individual dynamics are modeled by discrete linear equations in a finite state space. These equations were obtained by introducing modular addition terms into of the conventional Arnold cat map (ACM). In addition, by considering an interaction between the N particles, we obtained behavior of the entire network that is sensitive to initial conditions and whose period can be infinitely increased. The resulting system has a large period and can be used as a digital pseudo-random number generator (PRNG). This approach to designing pseudo-random integers solves the problem of dynamic degradation observed in numerical design of PRNG based on chaotic systems. The work concludes with an experimental FPGA implementation of the proposed dynamic network in a finite space of monobit precision, using the Verilog language on the Vivado 2018.3 programming platform. The maximum operating frequency achieved was 177.809 MHz with a throughput of 11.37 Gbit/s, using only 0.17% (90) lookup tables (LUTs) and 0.06% (65) flip-flops (FFs). The results of the statistical analysis of the generated random sequences demonstrate that the proposed system can be used for various digital applications. Specifically, it can be employed for implementing secure cryptography systems in real-time scenarios with limited hardware resources, such as IoT systems.

Keywords : Discret chaos, Arnold cat map, modular arithmetic, finite space, pseudo-randoms numbers, FPGA, cryptography, IoTs.

INTRODUCTION GÉNÉRALE

Le développement de la technologie des appareils mobiles intelligents connectés via internet favorise l'échange rapide des informations. En effet, des établissements tels que les hôpitaux, les centres de recherche et les lieux d'habitation sont de plus en plus connectés au réseau internet via des dispositifs basés sur l'internet des objets(IoT) et génèrent de grandes quantités de données [1]. Face à l'importance des données échangées, les utilisateurs font face à des menaces de sécurité potentielles liées à la confidentialité, l'intégrité, l'authenticité et la non-répudiation des données échangées. Cela conduit à un besoin de cryptographie pour la protection de ces données transmises via le réseau internet. Diverses techniques de cryptographie telles que sont répertoriées dans la littérature. Le niveau de sécurité que propose ces différentes techniques dépend de la qualité du générateur de nombre pseudo-aléatoire(GNPA) qu'elles intègrent [2]. Conçu à partir des systèmes dynamiques, celui-ci apparaît alors comme l'élément clé dans la sécurité des dispositifs de télécommunication. Cependant, la plupart des GNPAs proposés dans la littérature sont difficiles à intégrer dans les dispositifs digitaux à faible complexité comme l'IoT en raison du compromis entre la sécurité et les performances matérielles [3]. Cela est dû au fait qu'ils sont confrontés à un plus large éventail de limitations tels qu'une faible surface, une faible latence, une vitesse élevée, une faible puissance, une longue période, un caractère aléatoire élevé et une génération de séquences de clés sécurisées [4, 5]. La cryptographie à faible complexité est un domaine relativement nouveau qui devrait permettre de développer la mise en œuvre des GNPAs plus efficaces susceptibles d'être implémentés dans des dispositifs ayant une puissance de calcul, une batterie et une mémoire limitée [6]. Elle est adaptée à des dispositifs à faible puissance, et permet d'équilibrer le compromis entre les exigences de faibles coûts des ressources, les performances (en termes de débit) et la résistance aux attaques cryptographiques. Mis à part la cryptographie, la génération de nombres pseudo aléatoire est aussi un problème clé dans de nombreuses autres applications tels que : les simulations stochastiques, les tests de circuits numériques, les jeux de hasard, la simulation des phénomènes aléatoires dans les jeux vidéo ou des dessins animés etc.

Les GNPAs linéaires utilisent comme source d'entropie les systèmes dynamiques linéaires. Parmi ces générateurs on distingue : le générateur congruentiel linéaire [7] et le registre à décalage

à rétroaction linéaire [8] qui sont les deux méthodes GNPAs les plus simples et les moins complexes sur le plan matériel. De même, les GNPAs linéaires, tels que le Mersenne Twister [9] et les générateurs Fibonacci décalés [9, 10], sont également des méthodes GNPA couramment utilisées. Cependant, ces méthodes ne sont pas sécurisées en raison de leurs structures linéaires. De plus, elles génèrent généralement des séquences dont le caractère aléatoire est affecté par certaines régularités indésirables, et ne conviennent donc pas à une large classe d'applications (par exemple, les applications cryptographiques) [12–14].

Les systèmes dynamiques chaotiques quant à eux, ont des propriétés d'ergodicités et d'imprédictibilités qui leurs permettent de générer des séquences aléatoires totalement différentes en utilisant différents paramètres ou valeurs initiales. Grâce à ces propriétés, plusieurs auteurs ont proposé d'étudier ces systèmes dynamiques chaotiques pour générer des séquences pseudo-aléatoires [15–17]. Dans ces GNPAs, les nombres aléatoires sont obtenus par la numérisation du système chaotique continue. Cette opération est réalisée en discrétisant l'espace d'état sur un nombre de n bit à partir des méthodes de calcul numérique approximatif du système. Cependant, lors de la mise en œuvre dans des dispositifs à faible complexité ayant des valeurs de précisions arithmétiques n très faible, elles présentent les inconvénients de courtes périodes et de faibles propriétés statistiques. Ces défauts sont causés par la dégradation que subit les dynamiques chaotiques des systèmes [1], les rendant prédictibles et inadaptés à certaines applications. Pour éviter ces inconvénients, il est conseillé de réaliser une numérisation en simple ($n = 32$) ou en double ($n = 64$) précision, d'éffectuer la mise en cascade de plusieurs systèmes dynamiques identiques (ou différents) ou d'augmenter la dimension du système. Bien que ces solutions améliorent les propriétés statistiques des séquences générées, elles entraînent néanmoins comme inconvénient, une plus grande complexité de calcul que les GNPAs linéaires, nécessitant plus de ressources matérielles, un temps de calcul élevé et un processeur de haute gamme [17].

Récemment, plusieurs systèmes chaotiques ont été développé [18]. Ils peuvent être classées en deux catégories : les systèmes chaotiques à une dimension et les systèmes chaotiques multidimensionnels (MD). Les systèmes chaotiques unidimensionnels sont des systèmes mathématiques qui simulent l'évolution d'une seule variable d'état pendant un temps discret. Comme exemples nous avons la récurrence logistique, la récurrence de Tent et la récurrence chaotique linéaire par morceau. Ces systèmes ont généralement des structures simples et sont faciles à mettre en œuvre. Ils ont d'excellentes propriétés chaotiques et ont été utilisé pour différentes applications de sécurité [19]. Cependant, ils présentent plusieurs faiblesses en matière de sécurité : leurs plages chaotiques sont limitées [20], un nombre de paramètres de contrôle très réduit , et leurs sorties sont faciles à prédire

avec de faibles coûts de calcul [21, 22]. Par contre, les récurrences chaotiques MD modélisent les évolutions d'au moins deux variables. Nous avons comme exemples : la récurrence d'Hénon, le système de Lorenz, le système de Chen et Lee et les systèmes hyper chaotiques. Comparés aux systèmes chaotiques à une dimension, les systèmes chaotiques MD ont généralement de meilleures performances chaotiques, un grand nombre de paramètres de contrôle et leurs orbites chaotiques sont plus difficiles à prédire [23]. Cependant, les récurrences chaotiques MD ont des coûts de calcul élevés et sont difficiles à mettre en œuvre dans les dispositifs d'implémentation numérique tels que les DSP, FPGA, ASIC ou les microcontrôleurs. Ces faiblesses limitent leurs performances dans certaines applications basées sur le chaos, en particulier dans les applications en temps réel.

Le développement rapide des technologies de télécommunications numériques nécessite l'attention des chercheurs pour développer des systèmes de cryptographie en adéquation avec ces technologies et par conséquence de mettre en œuvre des GNPAs de faible complexité, rapides et efficaces. La récurrence du chat d'Arnold [24] (ou Arnold cat map, en anglais) est l'une des récurrences chaotiques $2D$ les plus étudiées. En raison de ses caractéristiques adaptées à la cryptographie de faible complexité, elle a été largement utilisée dans un certain nombre d'applications cryptographiques [25–28]. Au fil des années, un certain nombre de généralisations de la RCA $2D$ sont apparus dans la littérature [25, 29–31].

Pour équilibrer les compromis entre les exigences de faibles coûts en ressources, les performances (en débit) et la résistance aux attaques cryptographiques nous avons proposé dans cette Thèse, une nouvelle famille de récurrence du chat d'Arnold 8D monobit ($n = 1$) qui est une extension de la généralisation qui a été décrite dans [32] nommé "Pice-wise Linear Arnold Cat Map (PWLCM)" pour la génération rapide des séquences pseudo-aléatoires. L'objectif de cette proposition [32] vise à augmenter le nombre de paramètres de contrôle et la période de récurrence du Chat d'Arnold classique jusqu'à ce qu'il devienne plus grand que le nombre de points non triviaux de l'espace de phase. L'extension en 8D permet d'augmenter à son tour la taille de l'espace de phase et la clé de tout schéma cryptographique adoptant cette généralisation. Cela permet de résoudre les problèmes de sécurité connus dans les systèmes de cryptographie chaotiques. Cependant, la lenteur engendrée dans la génération des nombres pseudo-aléatoires est compensée par la réduction de la précision à 1-bit. En effet, une telle précision permet d'éliminer les opérations de multiplication contenues dans les additionneurs par de simples opérations logiques XOR. Il s'ensuit alors une réduction de la complexité du système et une augmentation du débit de la génération des nombres pseudo-aléatoires. Par la suite nous avons proposé de mettre en réseau plusieurs PWLCM monobit en dimension 8 ainsi conçu pour réaliser un générateur de nombres pseudo-aléatoires rapide, et

dont l'architecture nécessite très peu de ressources matérielles. Le générateur de nombres pseudo-aléatoires (GNPA) que nous proposons a été implémenté sur FPGA et ne nécessite que 90 tables de correspondances (LUT) et 65 bascules (FF), avec un débit de 11.37 Gbit/s. La mise en réseau permet d'étendre l'espace des clés jusqu'à 328 bits. Notre intérêt dans ces travaux se concentre sur l'utilisation de nos compétences en conception matérielle/logicielle avec les installations FPGA de l'unité de prototypage du laboratoire d'énergie et systèmes électriques et électroniques de l'université de Yaoundé 1, pour intégrer et mettre en œuvre de nouveaux processus d'itérations chaotiques sécurisés.

Dans ce projet de recherche, nous avons proposé un nouveau système dynamique assimilable à un générateur de nombres pseudo-aléatoires répondant aux exigences suivantes :

— Il est essentiellement basé sur la théorie des récurrences chaotiques numérisées, où seules les nombres entiers positifs sont considérés.

— L'implémentation matérielle réalisée est indépendante de la technologie (pas de DSP ou de blocs mémoires) et est facile à intégrer sur le système à puce pour les applications FPGA.

— Le débit élevé, la surface réduite et une faible consommation d'énergie sont des caractéristique que nous avons pris en compte.

— Être capable de traiter une large plage de données, possède un grand espace de clé et une période infinie.

— Enfin, il a un taux élevé de réussite aux tests statistiques.

A- CONTRIBUTION

Ce manuscrit présente la conception et l'évaluation d'un générateur de nombre pseudo-aléatoire basé sur la récurrence chaotique du chat d'Arnold monobit en dimension 8. Tout en mettant en évidence ses avantages par rapport à d'autres GNPA linéaires et chaotiques. Nos contributions dans cette thèse se résument comme suit :

— Il présente une étude d'un grand nombre d'implémentations matérielles sélectionnées de générateurs de nombres aléatoires sur FPGA. Les générateurs linéaires et non-linéaires sont discutés dans le cas des GNPAs. Chaque approche est expliquée en détail, une discussion des résultats sur les implémentations et les tests statistiques est systématiquement effectuée.

— Un nouveau système dynamique est proposé à la communauté scientifique. Les résultats de l'analyse dynamique de ce système ont montré qu'il présente un niveau d'aléa plus complexe

que celui de la récurrence du chat d'Arnold classique.

— Nous avons résolu dans cette thèse, le problème de dégradation dynamique des systèmes chaotiques lorsqu'ils sont implémentés sur des dispositifs aux ressources de calcul limitées. Nous avons proposé un générateur de nombres pseudo-aléatoires à faible complexité et sécurisé, effectuant des opérations sur 1 bit. Cette précision de calcul étant la plus petite possible qui puisse exister en numérique. Ce faisant, le générateur obtenu utilise très peu de ressources matérielles, présente un bon profil statistique, tout en fonctionnant à un débit de **11,37** Gbps.

B- ORGANISATION DE LA THÈSE

Le reste de ce manuscrit est divisé en quatre parties, comme détaillé ci-dessous :

Le chapitre 1, présente un état de l'art des générateurs de nombres pseudo-aléatoires. Les GNPAs linéaires et non-linéaires sont présentés, et les notions de système dynamique sont abordées à la fin de ce chapitre.

Le chapitre 2, débute, par une description des outils mathématiques utilisés pour caractériser l'aléa produit par les générateurs de nombres pseudo-aléatoires. Ensuite, un aperçu des fondements mathématiques concernant les GNPAs basés sur des itérations chaotiques, qui sont les principaux objets considérés au cours de notre thèse, est fourni. En particulier, nous présentons la récurrence du chat d'Arnold en dimension 2, la proposition que nous avons faite sur la généralisation de la récurrence du chat d'Arnold en dimension 2, et une extension en dimension 8 est ensuite déduite du système de dimension 2 que nous avons proposé.

Le chapitre 3, présente les résultats de l'implémentation des différents systèmes que nous avons proposé pour la génération des nombres pseudo-aléatoires. Par la suite, nous présentons notre méthodologie de conception de GNPA en mettant en réseau 8 PWLCM monobit en dimension 8 suivant une architecture série-parallèle. Le générateur de nombre pseudo-aléatoire ainsi obtenu a été mis en œuvre sur le dispositif FPGA Zybo (xc7z020clg400-1) pour des systèmes embarqués.

Et enfin, ce manuscrit se termine par une conclusion générale, dans laquelle la contribution est résumée et les travaux futurs prévus sont décrits.

CHAPITRE 1

REVUE DE LA LITTÉRATURE

1.1 Introduction

L'utilisation des générateurs de nombres pseudo-aléatoires a pris une dimension importante ces dernières décennies. Un grand nombre d'application dans le domaine des télécommunications, de la cryptographie, des simulations numériques ou encore des jeux de hasard, ont contribué au développement de plusieurs techniques permettant de produire ces nombres. Les méthodes utilisées pour la génération des nombres aléatoires dépendent de la source d'entropie utilisée. Elle peut provenir d'un phénomène physique pour les vrais générateurs de nombres aléatoires ou d'un processus algorithmique pour les générateurs de nombres pseudo-aléatoires.

1.2 Généralités sur les générateurs de nombres pseudo-aléatoires

Un générateur de nombres aléatoires (GNA) est un dispositif largement utilisé dans de nombreuses applications [33] telles que les simulations, l'analyse numérique, la programmation informatique, la cryptographie [34], la prise de décision, l'échantillonnage, etc. Il est capable de produire une séquence de nombres pour lesquels il est impossible de trouver un lien entre un nombre et ses prédécesseurs. Ces nombres sont produits à partir d'une source spécifique d'entropie. La **Fig.1.1** montre la classification des générateur de nombres aléatoires.

En fonction de la nature de la source d'entropie utilisée pour l'implémentation d'un GNA, on distingue : Les générateurs de nombres aléatoires déterministes (GNAD), et les générateurs de nombres aléatoires non déterministes (GNAN). Les GNAN, aussi appelés vrais GNA ("True Random Number Generator(TRNG)", en anglais) [35–37] utilisent comme source d'entropie, des processus physiques soumis au bruit thermique [38], au bruit atmosphérique [39], à la décroissance radioactive, etc., qui sont imprédictibles et incontrôlables. Ce type de générateur répondre aux exigences des applications cryptographiques. La **Fig.1.2** illustre l'architecture de conception générale d'un vrai générateur de nombres pseudo-aléatoires.

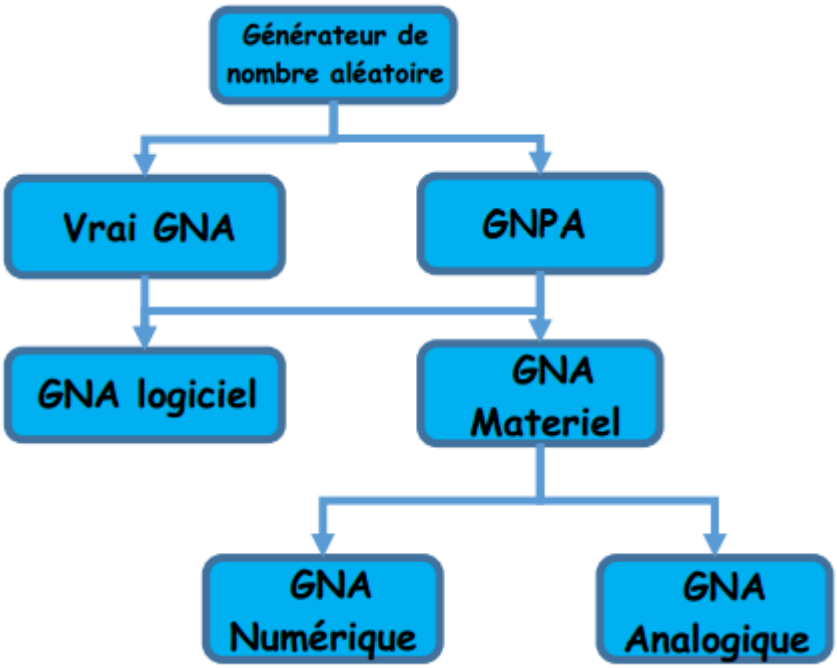


FIGURE 1.1 – Les différentes classes des générateurs de nombres aléatoires.

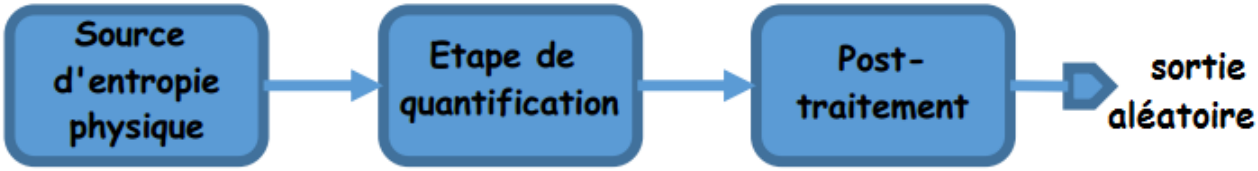


FIGURE 1.2 – Architecture générale d'un vrai générateur de nombres pseudo-aléatoires

D'autre part, les GNADs sont considérés comme des GNAs basés sur les processus déterministes. Ils sont définis par [40] un espace d'état S, une fonction de transition $f : S \longrightarrow S$, une fonction d'extraction des sorties $g : S \longrightarrow U$ à partir d'un état donné $x(t)$, ou U représente l'espace des sorties aléatoires. Ils génèrent une séquence de nombres à partir d'une graine initiale $x(0)$, ce qui les rendent moins coûteux, indépendant du matériel et rapide du point de vue de la mise en œuvre [41]. En raison de la nature déterministe du processus, ils produisent une série de nombres pseudo-aléatoires au lieu des nombres aléatoires. D'où le nom de générateur de nombres pseudo-aléatoires (GNPA) **Fig.1.3**. Les qualités statistiques de ces nombres produits doivent être proches d'un vrai aléatoire pour être considérées comme crypto-graphiquement sécurisées.

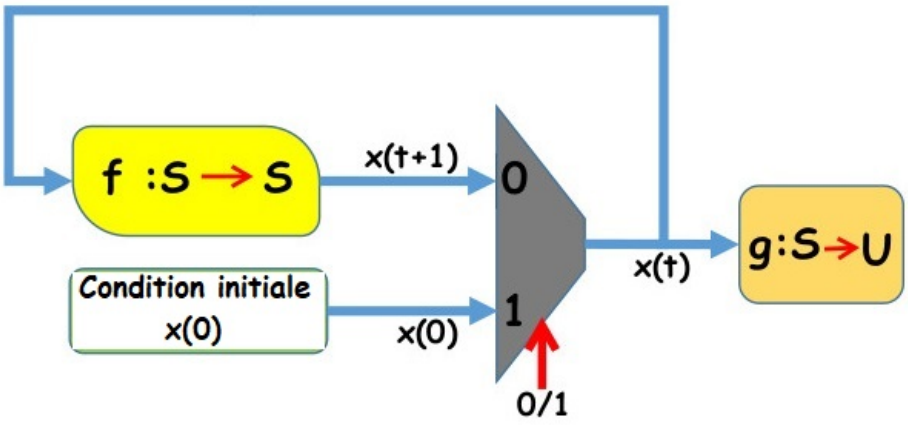


FIGURE 1.3 – Architecture générale d'un générateur de nombres pseudo-aléatoires

Les nombres pseudo-aléatoires étant produits par un processus déterministe, ils finissent toujours par entrer dans un cycle. La période du cycle représente le nombre minimum d'itérations nécessaire pour obtenir son état initial. Comme nous l'avons mentionné précédemment, la méthode traditionnelle de construction de GNA consistait à utiliser une machine mécanique ou un phénomène physique [42]. Cependant, malgré la qualité du caractère aléatoire généré, la plupart de ces techniques sont des processus lents (extraction du bruit d'un composant) ou coûteux (l'extraction ou la mesure du bruit peut nécessiter un équipement spécifique tel qu'un oscilloscope). Tous ces inconvénients poussent les chercheurs à développer des générateurs de nombres pseudo-aléatoires électronique basés sur la conception par logiciels. Ceux-ci consistent en la mise en œuvre des algorithmes déterministes en ciblant une carte électronique spécifique, tel qu'un réseau de porte logique programmable ("Field Programmable Gate Array (FPGA)"), les microprocesseurs, des traitements de signaux numériques ("Digital Signal Processor (DSP)") ou les microcontrôleurs.

Les dispositifs de prototypage FPGA sont de plus en plus utilisés, car ce sont des systèmes matériels reconfigurables. Ils permettent un prototypage rapide, c'est-à-dire explorer un certain nombre de solutions matérielles et sélectionne la meilleure dans un délai plus court [43]. La méthodologie de conception FPGA repose sur l'utilisation d'un langage de haut niveau tel que Verilog HDL, VHDL et d'un outil de synthèse. Pour cette raison, les FPGA sont devenus des plates-formes populaires pour la mise en œuvre de générateurs de nombres pseudo-aléatoires ou des schémas cryptographiques complets. Les cartes FPGA permettent d'obtenir une génération aléatoire à grande vitesse et de haute qualité. L'architecture générale d'un FPGA présentée à la **Fig.1.4** est basée sur le LCA (Logic Cell Array), qui est composé de trois parties, à savoir : "Configurable

Logic Block (CLB)" [44], Input Output Block (IOB) et les commutateurs d'interconnexion. Le FPGA pourrait en outre inclure des composants plus complexes comme un traitement numérique du signal, une mémoire vive (RAM), un gestionnaire d'horloge numérique ou un convertisseur analogique-numérique (ADC/DAC). La nomination des blocs internes dépend des éditeurs de FPGA (Xilinx, Altera, Actel...) même s'ils ont une fonctionnalité similaire. La structure CLB est principalement basée sur des tables de correspondances, "Look Up Table, (LUT)" [45], en plus des bascules, Flip-Flop (FF) et de quelques multiplexeurs. Une LUT d'entrée K est une matrice mémoire $2^K - 1$ bit basée sur une table de vérité d'entrées K bits. Ces dernières peuvent exécuter toutes les fonctions logiques comme XOR/ADD/SHIFT.

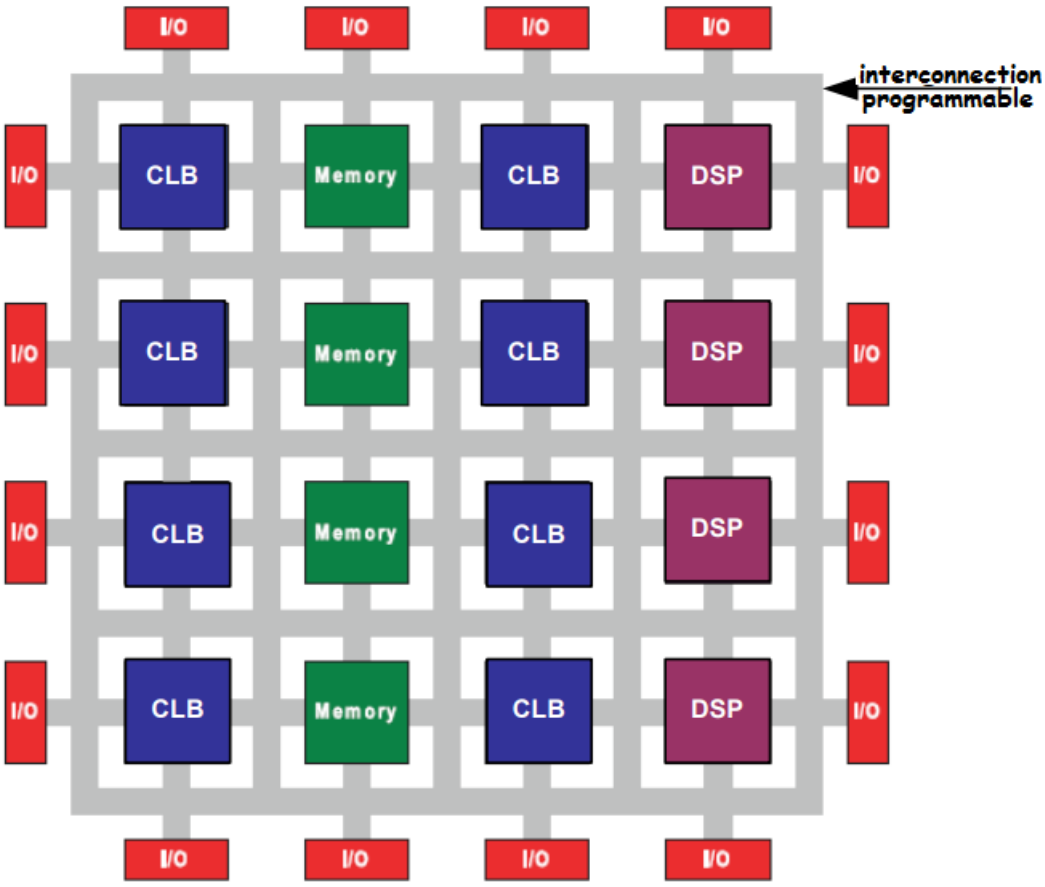


FIGURE 1.4 – Architecture générale d'un FPGA

Les qualités des différents modèle de GNPA implémentés sur FPGA sont évalués selon de nombreux critères. D'un point de vue statistique, **la sortie doit être vérifiée par rapport à une suite de tests bien connue comme ceux du test NIST ou du TestU01** [41]. Du point de vue

matériel, **un objectif est de fournir la fréquence la plus élevée par bit généré aléatoirement avec moins de ressources matérielles FPGA (CLB, IOB, ...)**.

De manière générale, les GNPA sont définies par la relation de récurrence $\mathbf{x}(t+1) = f(\mathbf{x}(t))$. f représente une fonction linéaire pour les GNPA linéaires (GNPAL) ou une application non-linéaire pour les GNPA chaotique (GNPAC). Étant donné que notre étude est centrée sur la mise en œuvre d'un système dynamique dont la dont l'évolution est comparable à celle d'un GNPA, nous allons dans la suite de ce chapitre passer en revue un large éventail de générateurs de nombres aléatoires mis en œuvre sur un FPGA. Les générateurs de nombres pseudo-aléatoires linéaires et non-linéaires seront discutés. Chaque approche sera expliquée. Les performances en termes de fréquence, de taille de zone, de faiblesses et d'évaluations statistiques seront enfin présentées, lorsqu'elles seront disponibles.

1.3 Les générateurs de nombres pseudo-aléatoires linéaires

Rappelons tout d'abord qu'une façon courante de définir un générateur de nombres pseudo-aléatoires linéaires est de considérer deux fonctions f et g, comme étant des récurrences linéaires définie par [46] :

$$f : \mathbb{F}_2^N \longrightarrow \mathbb{F}_2^N, g : \mathbb{F}_2^N \longrightarrow \mathbb{F}_2^M$$

$$\mathbf{x}(t+1) = f(\mathbf{x}(t)) \qquad \text{et} \qquad \mathbf{y}(t) = g(\mathbf{x}(t))$$

où généralement $N > M$, g est à sens unique, $x(0)$ est une graine fournie par l'utilisateur et $\mathbf{y}(t)$ est retourné à l'utilisateur. Par définition, $\mathbb{F}_2$ est un corps fini de cardinalité 2. Par conséquent, un GNPA linéaire de w bits peut être défini par les équations suivantes [47] :

$$\mathbf{x}(t+1) = A \cdot \mathbf{x}(t) \tag{1.1}$$

$$\mathbf{y}(t) = B \cdot \mathbf{x}(t) \tag{1.2}$$

$$r(t) = \sum_{l=1}^{w} \mathbf{y}_{l-1}(t) \cdot 2^{-l} \tag{1.3}$$

A l'étape t, $x(t) = (x_0(t), ..., x_{k-1}(t))^T \in \mathbb{F}_2^k$ est le vecteur d'état de k bits, $y(t) = (y_0(t), ..., y_{w-1}(t))^T \in \mathbb{F}_2^w$ est le vecteur de sortie à w bit ou k et w sont des entiers positifs, A est une matrice de transition $k \cdot k$ avec des éléments dans $\mathbb{F}_2$, B est une matrice de transformation de sortie $w \cdot k$ avec

des éléments dans $\mathbb{F}_2$, et $r(t) \in [0, 1)$ représente la sortie. Toutes les opérations en 1.1 et 1.2 sont effectuées en $\mathbb{F}_2$, c'est-à-dire modulo 2.

Suivant les valeurs de w et k utilisées, plusieurs cas peuvent être envisagés : le cas le plus simple est celui ou nous avons $w = k$, dans ce cas B est une matrice identité. Dans ce cas, les bits d'état sont directement utilisés comme bits de sortie aléatoires. Dans le cas où $w < k$, la sortie de l'**Eq.1.1** se propage dans un autre circuit pour subir un ensemble d'opération avant de produire les bit de sortie, comme dans le cas de Mersenne Twister [9]. Parmi les PRNGs qui existent dans la littérature.

1.3.1 Les générateurs congruentiels linéaires (GCL)

Cette famille de générateurs repose sur une simple formule de récurrence linéaire de la forme :

$$x(t+1) = (a \cdot x(t) + b) \bmod m \tag{1.4}$$

Où a représente le multiplicateur, b l'incrément et m le modulo tel que $0 \leq a, b \leq m - 1$. Lorsque $b = 0$ le GCL est appelé générateur congruentiel multiplicatif (GCM) [48].

Ici, $x(0)$ représente la graine du générateur et son choix est généralement effectué par des opérations dites d'accumulation d'entropie. Elles consistent à exploiter des données physiques variables (temps entre deux accès disques, clics de souris, saisie clavier, tailles de fichiers, bruit d'une résistance, température d'un composant, etc) pour modifier régulièrement la graine du générateur. Les termes de la suite sont compris entre 0 et $m - 1$. Comme le nombre de valeurs est fini, la suite est amenée à se répéter au bout d'un certain temps, on dit qu'elle est ultimement périodique de période maximale égale à $m - 1$. Pour que le générateur soit de période égale $m - 1$, il est important de respecter les règles de Knuth. Mais même en le faisant cela, rien ne garantit que les nombres générés auront de bonnes propriétés statistiques. C'est pour cette raison qu'elle a subi plusieurs optimisations au cours du temps parmi lesquelles nous avons les **Ranq1** et **Ran** [49].

Ranq1 est un MCG fonctionnant avec un modulo 2^{64}, tandis que sa graine est produite par un registre à décalage ou-exclusif (XOR) droit de 64 bits [50]. Rappelons d'abord que le XORshift prend une entrée et exécute itérativement un ou-exclusif du nombre binaire avec une traduction

légèrement décalée (gauche et droite) de lui-même. Le second, le générateur **Ran**, combine un générateur GCL avec deux registres à décalage XOR, et les résultats sont XORed par un générateur multiplicateur avec retenue ou "Multiply with Carry (MWC)" en anglais [51]. Dans MWC, l'**Eq.1.4** est modifiée comme suit : la constante b est remplacée par la retenue $b(t)$ qui est définie par $b(0)$, la retenue initiale qui est inférieure à a et $b(t+1) = \lfloor \frac{a \cdot x(t) + b(t)}{2^{32}} \rfloor$.

Dans [52] les auteurs présentent un couplage de deux "Coupling Linear Congruential Generators (CLCG)", désignés par **CLCG-1** et **CLCG-2**. Chaque système génère une sortie séparée avec différents paramètres décrits comme suit :

$$\begin{aligned} x_1(t+1) &= (a_1 x_1(t) + b_1) \bmod 2^k \\ x_2(t+1) &= (a_2 x_1(t) + b_2) \bmod 2^k \\ C_1(t+1) &= \begin{cases} 1 & \text{if} \quad x_1(t+1) \geq x_2(t+1) \\ 0 & \text{si non} \end{cases} \end{aligned} \tag{1.5}$$

$$\begin{aligned} x_3(t+1) &= (a_3 x_3(t) + b_3) \bmod 2^k \\ x_4(t+1) &= (a_4 x_4(t) + b_4) \bmod 2^k \\ C_2(t+1) &= \begin{cases} 1 & \text{if} \quad x_3(t+1) \geq x_4(t+1) \\ 0 & \text{si non} \end{cases} \end{aligned} \tag{1.6}$$

Le premier CLCG-1 1.5 est caractérisé par $\{x_1(t+1); x_2(t+1); C_1(t+1)\}$ tandis que le deuxième CLCG-2 1.6 est défini avec $\{x_3(t+1); x_4(t+1); C_2(t+1)\}$ ($C_1(t+1)$ et $C_2(t+1)$ sont des séquences de bits). CLCG-2 à pour rôle de sélectionner quel bit doit être extrait de CLCG-1 comme sortie finale $y(t)$: $y(t) = C_1(t+1)$ si $C_2(t+1) = 0$, sinon le bit $C_1(t+1)$ est ignoré. Par exemple, les auteurs supposent un format simple des multiplicateurs : $a_1 = a_3 = 2^{\delta_1} + 1$ et $a_2 = a_4 = 2^{\delta_2} + 1$, où $1 < \delta_1; \delta_2 < k$. En effet, le nouveau format de $x_1(t+1)$ pour CLCG-1 (et de même pour x_2, x_3 et x_4) est le suivant :

$$x_1(t+1) = ((2^{\delta_1} x_1(t) \bmod 2^k) + x_1(t) + b_1) \bmod 2^k \tag{1.7}$$

où $2^{\delta_1} x_1(t)$ est le résultat du décalage de $x(t)$ exactement une fois vers la gauche, et la modulation correspond au k bits les moins significatifs de $(2^{\delta_1} x_1(t) \bmod 2^k)$. Cependant, une grande valeur de k conduit à une latence élevée. Pour résoudre ce problème, une implémentation de P étapes d'addition et de comparaison pour les deux CLCG a été proposée par la suite. Il divise les nombres de k-bits repartis en P-bloc, traite chaque partie de k/P-bits dans une étape de pipeline, et génère finalement 1-bit de C_1 et C_2 simultanément. De plus, il prend les résultats de

chaque étape et les envoie à la fois aux étapes précédentes et suivantes, afin de produire les sorties actuelles et futures.

1.3.2 Registre à décalage de rétroaction linéaire

Les générateurs de registre à décalage à rétroaction linéaire ("Linear Feedback Shift Register (LFSR)") ou Tausworthe [53] sont des générateurs à récurrence linéaires. Un LFSR utilise une séquence de bascules (FF) comme registres à décalage pour générer un bit par itération. Chaque registre est connecté à ses voisinages, la valeur binaire dans chaque registre est décalée à chaque itération, tandis que le dernier registre produit la sortie (**Fig.1.5**). Un XOR est utilisé sur certains registres conçus pour construire une entrée de rétroaction vers le premier registre, qui est exprimée par un polynôme caractéristique. Comme le montre la **Fig.1.5**, deux configurations sont généralement considérées, à savoir les configurations Galois et Fibonacci. Ces deux implémentations susmentionnées sont synchronisées avec une horloge principale (CLK), dans laquelle à chaque front les données maintenues (1 bit) dans FF sont libérées et une nouvelle entrée est stockée. La matrice A de l'**Eq. 1.8** est dans ce cas :

$$x(t+1) = A \cdot x(t) \bmod m \tag{1.8}$$

$$A = \begin{pmatrix} 0 & I_{k-1} \\ a_k & a_{k-1}, ..., a_1 \end{pmatrix} \tag{1.9}$$

Le polynôme caractéristique de la matrice A est $x(t+1) = a_1 x(t) + \cdots + a_k x(t+1-k)$. Dans les équations ci-dessus, $a_1; ...; a_k$ représentent les coefficients LFSR, chacun dans $\mathbb{F}_2$. Ainsi, si l'un de ces coefficients existe, il déploie un opérande XOR sur la sortie. Même si de nombreuses implémentations FPGA de tels LFSR peuvent être trouvées dans la littérature, seules quelques-unes d'entre elles sont réellement optimisées pour cette architecture. Dans [54], les auteurs présentent deux types de LFSRs. Le premier, appelé "Shrinking Generator (SG)" utilise deux LFSRs de 67 bits (LFSR-1 et LFSR-2). A chaque cycle d'horloge, la SG prend directement la valeur du bit de sortie qui est généré par le second LFSR-2 si le bit de sortie du premier LFSR-1 est égal à 1. Dans le cas contraire, les deux sorties sont rejetées. La deuxième version, nommée "Alternating Step Generator (ASG)" considère un troisième LFSR-3 de 141 bits en plus des deux précédents. Ce dernier est utilisé pour contrôler quel bit de sortie sera extrait des deux premiers LFSR de 131 bits.

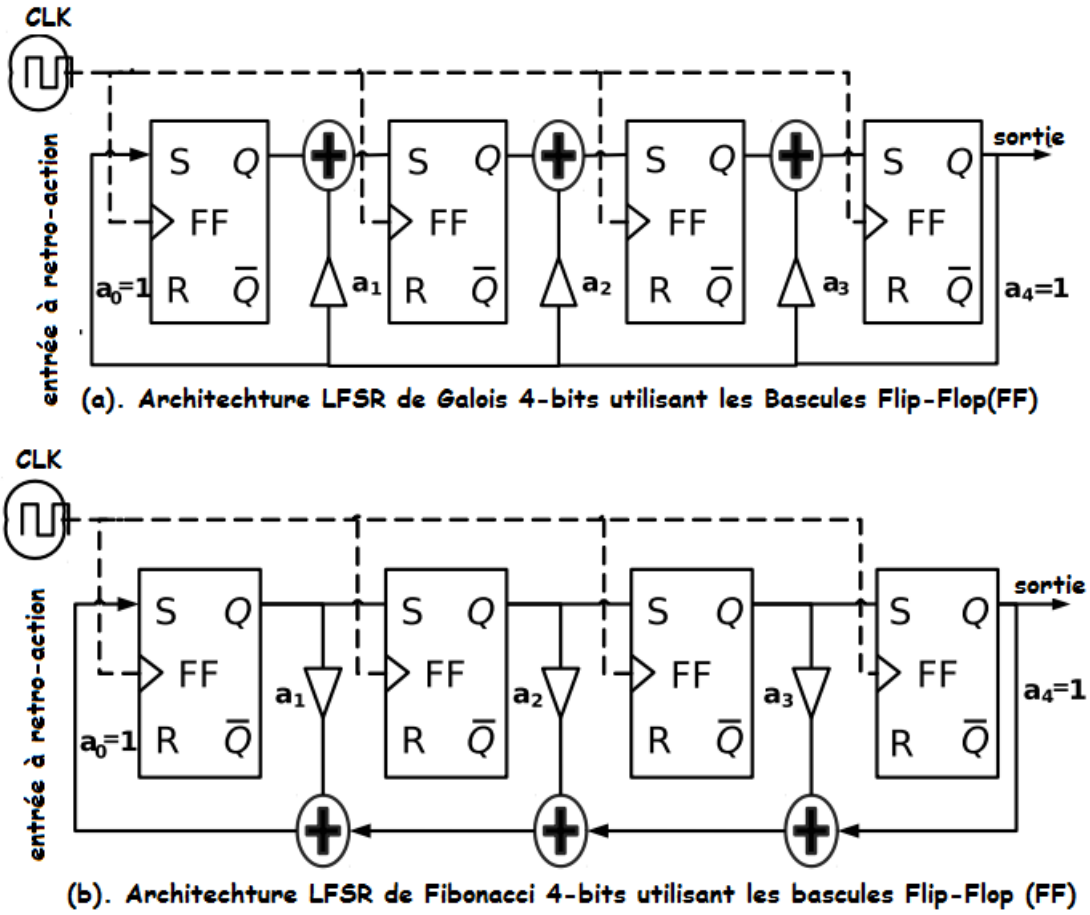


FIGURE 1.5 – Un générateur de registre à décalage à rétroaction linéaire 4 bits avec un polynôme de rétroaction $a_0X_4 + a_1X_3 + a_2X_2 + a_3X_1 + a_4$, avec $(a_0 = a_4 = 1)$.

À des fins de comparaison, si T_1, T_2 et T_3 sont respectivement les périodes de LFSR-1, LFSR-2 et LFSR-3, notons que le SG a une période totale de $T_{SG} = (2^{T_1} - 1)(2^{T_2} - 1)$ (longueur de 64 bits), alors que c'est $T_{ASG} = 2^{T_1}(2^{T_1} - 1)(2^{T_3} - 1)$ pour ASG (longueur de 128 bits).

1.3.3 Les générateurs optimisés à table de correspondance

Les générateurs optimisés à table de correspondance ou "Look Up Table(LUT)" utilisent un bloc logique comme composant numérique défini dans un "Configurable Logic Block (CLB)" fourni par les fournisseurs de FPGA. Il est utilisé pour mettre en œuvre de nombreux générateurs de fonctions et d'opérations pour réduire la quantité de ressource matérielle utilisée lors de l'implémentation. Une LUT est constituée d'un bloc de RAM (Random Access Memory) implémenté sous forme de table de vérité indexée par les entrées de la LUT.

Dans [55–57], les auteurs présentent une série de GNPA LUT basés sur des algorithmes récursifs à matrice linéaire $\mathbb{F}_2$ (**Fig.1.6** [57]). L'idée principale est de produire une efficacité maxi-

male au niveau de la zone. Les auteurs associent soit des bascules (FF), soit des registres à décalage (SR), soit un bloc de RAM à la LUT pour effectuer des opérations de décalage/multiplication dans le FPGA. Cependant, la création de séquences à longue période $T = 2^w$ avec cette méthode est une tâche difficile. Pour résoudre ce problème, de grandes paires FF, SR, RAM optimisées basées sur LUT sont étudiées.

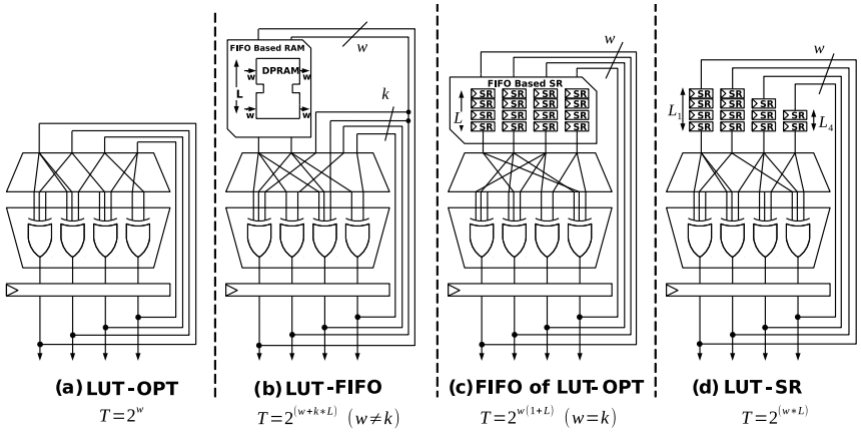


FIGURE 1.6 – Registre à décalage basé sur LUT et GNPA optimisé FIFO FPGA : (a) La carte de chaque ligne de la matrice de récurrence comme une porte XOR en utilisant LUT-FF, (b) utilise la mémoire de bloc RAM comme $k \cdot k$ FIFO pour stocker les séquences récursives, (c) charge l'état dans le registre à décalage SR basé sur FIFO au lieu de BRAM, (d) cascade de n'importe quel nombre de Xilinx SRL32 pour créer un k-bit SR

Le premier GNPA proposé sur la **Fig.1.6**(a) est appelé LUT-OPT (LUT optimisé, il relie chaque ligne de la matrice de récurrence A en tant que porte XOR en utilisant uniquement LUT et FF. Pour générer w bits par cycle, il faut w LUT-FF dans une seule LUT de k bits pendant une période de $T = 2^w$ où $w = k$. Leurs estimations des ressources FPGA concluent que même si une application nécessite 64 bits pour chaque cycle, leurs implémentations utilisent nécessairement 512 LUT-FF pour produire une période de $2^{512} - 1$. La seconde, la LUT-FIFO (b), permet d'augmenter la période jusqu'à $T = 2^{(w+k \cdot L)}$ sans utiliser le couple LUT-FF, mais qui utilise la mémoire bloc RAM (RAM double port) de FPGA sous forme de FIFO $L \cdot k$ pour stocker les séquences récursives. Dans ce cas, chaque nouveau bit de sortie dépend d'un bit de la dernière itération. Ils proposent ensuite un registre à décalage SR (c) basé sur FIFO avec une longueur fixe L de 1 bit, pour charger l'état k-bit en parallèle au lieu d'utiliser une RAM à double port. Ils proposent également un LUT-SR GNPA (d) qui transforme l'utilisation de LUT en registre à décalage de k bits en utilisant "Xilinx SRL32", la longueur de chaque "SR" variant comme suit : $1 < L_i < L$. "Xilinx SRL32",

permet la mise en cascade de n'importe quel nombre de registres à décalage jusqu'à 32 bits pour créer un registre à décalage avec n'importe quelle taille nécessaire.

1.3.4 GFSR, TGFSR, Mersenne Twister

Le "Twisted Generalized Feedback Shift Register (TGFSR)" est une extension du "Generalized Feedback Shift Register (GFSR)" [14] qui utilise un tableau de registres à décalage pour générer plusieurs bits. Par conséquent, le TGFSR est basé sur la répétition de la séquence de N valeur $x_0 ...., x_{N-1}$, contenant chacun k bits et deux paramètres qui sont : un masque binaire de taille c telle que $c \leq k-1$ et un point médian initial m avec $1 \leq m \leq N$, comme indiquer sur la **Fig.1.7**

TGFSR calcule la $t + Nime$ valeur $(t = 0, 1, ...)$ à partir des deux premières valeurs $x(t)$ et $x(t+1)$ avec une valeur médiane $x(t+m)$ suivant les étapes ci-dessous :

1. Il calcule les c bits de poids faible (LSB) de $x(t+1)$ et les $k-c$ bits de poids fort (MSB) de $x(t)$. Ces deux vecteurs sont obtenus grâce aux deux vecteurs de masques de bits suivants : $\bar{S}_c$ pour (0 ; ... ; 0 ; 1 ; ... ; 1) et $\underline{S_{k-c}}$ pour (1 ; ... ; 1 ; 0 ; ... ; 0).
2. Ces deux vecteurs sont ensuite concaténés par $(x(t)\&\underline{S_{k-c}})|(x(t+1)\&\bar{S}_c)$.
3. Le résultat x_0 est alors "multiplié" par une matrice A, caractérisée par des valeurs $(a_0; a_1; ...; a_{w-1})$ telles que définies dans l'**Eq.1.11**.
4. Les résultats finaux des calculs précédents sont alors XORed avec le mot médian $x(t+m)$.

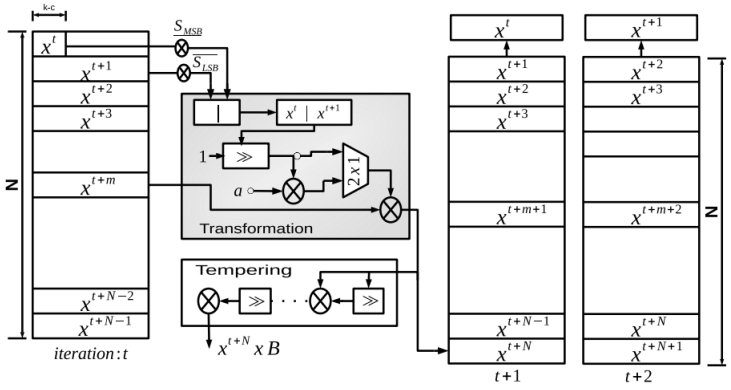


FIGURE 1.7 – Architecture de registre à décalage torsadé à rétroaction généralisée : à chaque opération de récurrence t, il calcule $x(t+N)$ grâce aux trois mots $x(t)$, $x(t+1)$, et $x(t+m)$ et génère la sortie avec la fonction de temporisation.

En mettant $c = 0$, alors l'**Eq.1.10** représente le TGFSR PRNG [58], inversement c'est Mersenne Twister [9].

$$x(t+N) = x(t+m) \oplus (((x(t)\&\underline{S_{k-c}})|(x(t+1)\&\bar{S}_c)) \times A) \tag{1.10}$$

ou :

$$x^{'} \times A = \begin{cases} x^{'} \gg 1 & \text{if} \quad x_0^{'} = 0 \\ (x^{'} \gg 1) \oplus (a_0, a_1, ..., a_{w-1}) & \text{si non} \end{cases} \tag{1.11}$$

Considérez c_1 et c_2 comme des masques de bits donnés et b_1, b_2, b_3 et b_4 sont des paramètres entiers constants. A l'itération t, TGFSR utilise un module de temporisation pour améliorer l'équidistribution. Cette étape, qui est décrite comme une séquence de calcul bit à bit/décalage est équivalente à un produit matriciel. Celui-ci est défini dans l'**Eq.1.12** où c_1, c_2 (resp, b_1, b_2) sont des masques de bits tempérés (resp, décalages de bits).

$$\begin{aligned} z &= x(t+N) \oplus (x(t+N) \gg b_1) \\ z &= z \oplus (x(t+N) \ll b_2)\&c_1) \\ z &= z \oplus (x(t+N) \ll b_3)\&c_2) \\ y(t) &= z \oplus (z \gg b_4) \end{aligned} \tag{1.12}$$

Mersenne Twister (MT) est proposé comme un cas spécial de TGFSR qui a une longue période de $2^{wN-c} - 1$. Pour ce faire, les auteurs de [58] proposent deux configurations de MT : les "Mersenne Twisters" MT19937 et MT11213. Ces deux configurations ont pour périodes respectives $2^{19937} - 1$ et $2^{11213} - 1$. Ils sont Le plus souvent utilisés dans la méthode de monte Carlos pour des application de finance, plusieurs implémentations de ces deux générateurs ont été proposé au fil des années [59–62]. La plus récente a été proposé dans [63]. Les auteurs proposent une solution beaucoup plus optimale à l'utilisation des RAM, qui est nommée "Circular Buffer (CB)" **Fig.1.8**. La solution est basée sur la relation fixe entre les indices des mots. Les valeurs x_j^t, x_{j+1}^t et x_{j+m}^t écrits dans le tampon passent à l'unité de transformation au fur et à mesure qu'ils sont produits. A chaque itération, le premier nombre x_j^t est cadencé hors de la mémoire tampon tandis que de nouvelles données x_j^{t+1} sont écrites à l'emplacement libre. De cette façon, la récurrence linéaire et la mémoire tampon des registres peuvent être remplacé par une mémoire tampon circulaire. La récurrence linéaire est effectuée par une logique combinatoire entre l'entrée et la sortie de la mémoire tampon. Par conséquent, l'architecture est simplifiée puisqu'aucune opération logique

pour les indices de table des valeurs n'est nécessaire.

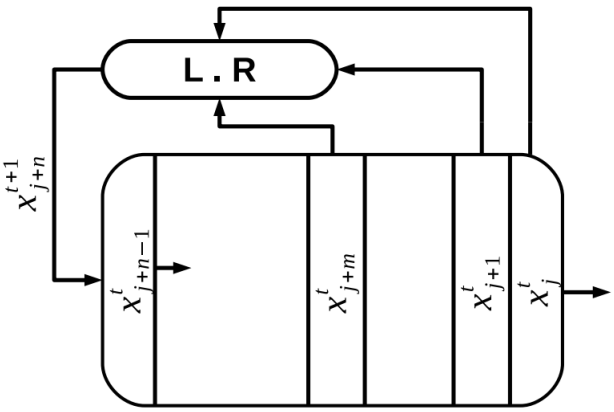


FIGURE 1.8 – Implémentation de la récurrence linéaire (L.R) pour "Mersenne Twister" GNPA en utilisant le tampon circulaire des registres (L.R est la récurrence linéaire de la fonction de transfert de MT)

1.4 Les générateurs de nombres pseudo-aléatoires non linéaires

La non-linéarité est une propriété utilisée pour décrire des relations qui ne peuvent pas être représentées par une ligne droite. Les générateurs de nombres pseudo-aléatoires chaotiques sont basés sur des applications non linéaires définies par des systèmes dynamiques chaotiques qui satisfont la propriété de chaos topologique de Devaney.

1.4.1 Les GNPAs chaotiques différentiels

Ce sont des générateurs basés sur l'implémentation numérique d'un système chaotique évoluant dans un espace temporelle continu. La mise en œuvre de tel système dynamique dans des dispositifs numériques, nécessite une opération de numérisation qui consiste d'une part à sa résolution à partir de différentes méthodes discrétisation qui sont : Runge-Kutta d'ordre 4 [67], les techniques de Euler [68] et la méthode des points milieu [69] et d'autre part en utilisant une opération de quantification appliquée aux différents résultats obtenus après discrétisation du système.

Dans [70] les auteurs présentent un GNPA hyper-chaotique quadridimensionnel non autonome basé sur les équations différentielles de Rössler. Dans un tel système chaotique, certains comportements indésirables peuvent apparaître. Ainsi, un contrôle avancé des processus est nécessaire afin

de retarder l'apparition de l'hyper chaos. Par conséquent, les auteurs ont utilisé l'approximation d'Euler et une fonction de contrôle de 256 bits Linear Feedback Shift Register (LFSR), dont les sorties sont multipliées par le coefficient approprié de la fonction de contrôle. Cependant, un post-traitement de Fibonacci LFSR basé sur 256 bits est utilisé pour supprimer la prédictibilité à court terme du générateur hyper-chaotique et pour réussir les tests statistiques de NIST. Le post-traitement combine deux boucles de rotation et des boucles de rétroaction XOR. La première rotation utilise une rotation statique fixe de 1 bit pour supprimer la prédictibilité à court terme. La seconde est basée sur une rotation variable contrôlée par une suite de Fibonacci de k-bits. Le problème de sensibilité différentielle est résolu en changeant n'importe quel bit pendant que les autres bits se propagent durant n-cycles.

Dans [71] un GNPA basé sur un système memristive hyper-chaotique à quatre ailes 5D (HFWMS) et mise en œuvre sur la carte FPGA est présenté par les auteurs. Le HFWMS 5D a un équilibre multiligne et trois exposants de Lyapunov positifs, ce qui indique que le système a un comportement dynamique très complexe. C'est sur cette base qu'ils proposent un GNPA basé sur le HFWMS $5D$ qui a été implémenté sur le FPGA. Le PRNG proposé est implémenté en langage VHDL, modélisé et simulé sur la plateforme Vivado 2018.3, et synthétisé par le dispositif FPGA ZYNQ-XC7Z020 sur Xilinx. Le module de post-traitement se compose de 16 registres à décalage linéaires et d'une chaîne XOR à 15 niveaux. La fréquence de fonctionnement maximale est de $138,331$ MHz et la vitesse est de $15,37$ Mbit/s. Les ensembles de bits aléatoires générés par PRNG ont été ensuite vérifiés par la norme statistique NIST 800.22. La sécurité est analysée par la dégradation dynamique, l'espace de clés, la sensibilité des clés et la corrélation. Les expériences montrent que la conception peut être appliquée à diverses applications de mot de passe intégrées. Comme le système est un système chaotique autonome en temps continu, il doit être discrétisé avant l'échantillonnage. Pour cela les auteurs ont utilisés L'algorithme RK4 pour discrétiser le modèle numérique du HFWMS $5D$ qu'il propose.

Bien que les GNPA conçus à partir des systèmes dynamiques différentiels passent les tests statistiques et sont utiles pour la conception des crypto-systèmes sécurisés, elles présentent néanmoins des inconvénients liés à la quantité énorme de ressources matériels utilisés lors de leur implémentation. Cela est causes l'usage des opérations de discrétisation. Dans le but de concevoir des PRNG nécessitant le moins de ressource matérielle, les chercheurs du domaine se sont orientés sur l'utilisation des systèmes chaotiques discrets.

1.4.2 Les GNPAs chaotiques récurrentiels

Les GNPA à récurrence chaotique sont basés sur des systèmes dynamiques discrets qui utilise une transformation dynamique non linéaire. La plupart de ces générateurs sont basés sur la récurrence Logistique [72], définie comme suit : $x(t+1) = \alpha(1+x(t))$, où $0 < x(t+1) < 1$ et α est le potentiel biotique $(3,57 < \alpha < 4,0)$. La carte logistique dépend principalement du paramètre α : son comportement chaotique est perdu lorsqu'il est hors de la plage fournie ci-dessus. De plus, si $\alpha > 4$ et pour presque toutes les valeurs initiales, les sorties divergent et sortent de l'intervalle $[0;1]$. La deuxième fonction la plus fréquemment utilisée est la carte chaotique de Hénon [73], qui prend un point $(x(t), y(t))$ dans l'unité carrée du plan et le transforme en un nouveau point $(x(t+1), y(t+1))$. Cette application est définie par ces équations : $x(t+1) = 1 + y(t) + 2a(x(t))$ et $y(t+1) = bx(t)$, où a et b sont appelés paramètres canoniques.

Dans [74], les auteurs ont utilisé la représentation en point fixe [75] pour implémenter la carte logistique en utilisant le logiciel MATLAB DSP System Toolbox. Le format à virgule fixe est une approximation des nombres réels, avec beaucoup moins de précision et de plage dynamique que le format à virgule flottante. Néanmoins, il a le mérite d'être très efficace dans les applications à haut débit et à faible puissance. Cette unité nécessite moins de puissance et de coût pour manipuler ce type de nombres que les circuits à virgule flottante habituels. Ils génèrent de nombreux designs avec des longueurs de bit différents variant de 16 à 64 bits, où les ressources dépendent de la précision (24 à 53 bits). La multiplication est mise en œuvre avec des blocs DSP de FPGA qui effectuent des multiplications $18 \cdot 25$ bits, tandis que la multiplication par une constante est une simple série d'opérations d'addition et de décalage à gauche.

[76], implémente dans le dispositif FPGA quatre récurrences chaotiques différentes, à savoir les récurrences chaotiques dites de Bernoulli, Chebychev [77], Tent et Cubic. L'implémentation se fait avec et sans les blocs DSP du FPGA pour les opérations de multiplication. Les résultats montrent que la carte chaotique de Bernoulli donne un rapport surface/puissance plus élevé par rapport aux autres générateurs chaotiques. Enfin, dans [78] est proposée une méthode pour générer des séquences pseudo-chaotiques unidimensionnelles basées sur la carte de chat d'Arnold discrète définie sur l'anneau d'entiers $\mathbb{Z}_{2^m}$. La période des suites générées est étudiée en utilisant les propriétés de la suite de Fibonacci sur $\mathbb{Z}_{2^m}$. Les séquences pseudo-chaotiques sont utilisées pour concevoir un générateur de nombres pseudo-aléatoires et les propriétés statistiques de ce GNPA sont analysées par la suite de tests statistiques du NIST. Ils ont implémenté le GNPA proposé

dans un réseau de porte logique programmable(FPGA) sur 32 et 64 bits. Ils montrent que pour le même débit de génération de bits, les séquences définies sur $\mathbb{Z}_{2^m}$ nécessitent moins de ressource matérielle que celle définie sur $\mathbb{Z}_{3^m}$ [79].

Cependant, il n'existe pas de solution parfaite aux déficiences dynamiques des systèmes numériques chaotiques. Une proposition donnée peut permettre de résoudre un problème mais en ignorer un autre. Par exemple, la plupart des propositions ne tiennent pas compte du coût de mise en œuvre. En d'autres termes, l'augmentation de la précision de calcul ou la mise en cascade de plusieurs cartes chaotiques augmente le niveau aléa des séquences produites, mais affecte directement le coût de mise en œuvre.

1.4.3 Les GNPAs basés sur des itérations chaotiques

Soit un ensemble $\mathbb{B}$, $f : \mathbb{B}^N \longrightarrow \mathbb{B}^N$ une fonction itérative et $S \in \mathbb{S}$ une stratégie chaotique. Une itération chaotique est définie par [80] :

$$\begin{aligned} & x(0) \in \mathbb{B}^{N^*}, \\ & \forall t \in \mathbb{N}, \forall i \in [1; N], x_i(t) = \begin{cases} x_i(t-1) & \text{si} \quad S^t \neq i \\ f(x(t))_{S^t} & \text{si} \quad S^t = i \end{cases} \end{aligned} \tag{1.13}$$

Les itérations chaotiques génèrent un ensemble de vecteurs (vecteur booléen), ils sont définis par un état initial $x(0)$, une fonction d'itération f et une stratégie chaotique S.

Une première application de l'itération chaotique comme approche GNPA a été présentée dans deux variantes CIPRNG pour FPGA, à savoir le CIPRNG-XOR et le CIPRNG-MultiCycle ([46, 81]). Les auteurs indiquent néanmoins que seul XOR-CIPRNG satisfait à la fois les faibles ressources matérielles et un succès contre la batterie TestU01.

Dans [2], les auteurs ont proposés un nouveau générateur de nombres pseudo-aléatoires basé sur le chaos matériel, qui est principalement basé sur fonction booléenne GCI embarquée et une permutation, ce générateur nommé GCIPRNG est considéré comme un post-traitement sur celui en entrée. Il présente généralement un meilleur profil statistique que son entrée, tout en fonctionnant à une vitesse similaire. La dépendance chaotique entre l'entrée et la sortie est alors prouvée, ainsi que la préservation de la propriété crypto-graphiquement sécurisée. La proposition a été entièrement déployée sur un FPGA, elle fonctionne complètement en parallèle. Dans cet article, le plus rapide

des générateurs conçus sur 32 bit utilise 4936 (LUT et FF) comme ressource pour un débit $6,95$ Gbps. Pour les générateurs conçus sur 64 bits, le plus rapide a un débit de 11.5 Gbps et utilise 10856 comme ressource matérielle totale. Tous les générateurs conçus passent avec succès les tests statistiques du NIST. Et en plus, le générateur ayant un débit de 11.5 Gbps est considéré par les auteurs comme étant le générateur le plus rapide au monde sur un dispositif physique (FPGA) qui passe les tests statistiques.

1.5 Systèmes dynamiques

Un système dynamique est défini à partir d'un ensemble de variables d'état $\{x_i\}$ formant le vecteur d'état $X = \{x_i\} \in \mathbb{R}$, $i = 1...d$, où d représente la dimension du vecteur. Nous appelons état d'un système, l'ensemble des variables qui, étant connues à l'instant initial, permettent de décrire l'évolution du système. L'ensemble de tous les états que le système peut obtenir est appelé l'espace des phases. Le processus évolue de façon déterministe si ses états futurs sont caractérisés par la connaissance de ses états présents et passés. La loi d'évolution dans le temps de ce système dynamique est généralement appelée "dynamique". En conclusion, la notion de déterministe vient du fait que le système est caractérisé par son état initial et sa dynamique [**?**].

D'après Devaney, les systèmes dynamiques sont qualifiés de chaotiques, lorsqu'ils sont définis comme étant [**?**, 66] : une fonction $f : \chi \longrightarrow \chi$ est dite chaotique sur χ si (χ, f) est régulière, topologiquement transitive et sensible des conditions initiales.

Considérons un espace métrique (χ, d) et une fonction continue $f : \chi \longrightarrow \chi$.

— f est dite topologiquement transitive, si pour tout couple d'ouverts $U, V \subset \chi$, il existe $k > 0$ tel que $f^k(U) \cap V \neq \emptyset$.

— (χ, f) est dit régulier si l'ensemble des points périodiques est dense dans X .

— f a une dépendance sensible aux conditions initiales s'il existe $\delta > 0$, tel que, pour tout $x \in \chi$et tout voisinage V de x, il existe $y \in V$ et $t > 0$ tels que $|f(x,t) - f(y,t)| > \delta$. δ est appelé la constante de sensibilité de f.

Les systèmes dynamiques chaotiques évoluent dans le temps de façon déterministe, c'est-a-dire qu'à partir d'une "Condition Initiale" donnée à l'instant "Présent", va correspondre à chaque instant ultérieur un et un seul état "Futur" possible. Cette évolution peut alors se modéliser de deux façons distinctes :

— Système dynamique chaotique continue : pour le cas où la composante "temps" est continue ($t \in \mathbb{R}$).

— Système dynamique chaotique discret : pour le cas où la composante "temps" est discrete ($t \in \mathbb{N}$).

1.5.1 Système dynamique chaotique continue

Un système dynamique chaotique continu est une structure représentée par une équation différentielle ordinaire dont la dynamique évolue en temps continue. Il est défini par :

$$X \in \mathbb{R}^d, t \in \mathbb{R}, \ \frac{dX}{dt} = F(X(t), \nu, t)$$

Avec $X(t) \in \mathbb{R}^d$ qui représente le vecteur d'état du système à chaque instant t, ν le vecteur des paramètres, $\mathbb{R}^d$ l'espace de phase et F une fonction d'un système dynamique continue dans le temps qui pour chaque couple $(X(t_0), t_0)$ on peut identifier une solution unique du système.

Une représentation basique de ce type de système dynamique a été proposé dans [64, 65] par l'équation

$$\begin{aligned} -\dddot{X} &= -\ddot{X} + B(\dot{X}) + X \\ B(\dot{X}) &= \begin{cases} \alpha_1 & \text{if} \quad \dot{X} \geq 1 \\ \alpha_2 & \text{si non} \end{cases} \end{aligned} \tag{1.14}$$

Ou α_1, α_2 sont des valeurs entières dans la condition de basculement. l'idée étant de créer un système avec un seul point d'équilibre à l'origine. Ceci, pour garantir une dynamique chaotique du système. On distingue comme système chaotique :

a. Système de Lorenz

Découvert en 1963, par le chercheur Edward Norton Lorenz lors de l'étude des phénomènes de convection (Rayleigh-Bénard) thermique dans l'atmosphère. Il avait comme idée de chercher un modèle permettant de simplifier les d'équations aux dérivées partielles de Navier Stokes. Il obtient alors un système dynamique différentiel possédant trois degrés de liberté, dont l'intégration numérique est beaucoup plus simple que celles de Navier Stokes. Ce système dynamique différentiel est connu sous nom de : Système de Lorenz et se présente comme suit :

$$\begin{cases} \frac{dx}{dt} = a(y-x) \\ \frac{dy}{dt} = rx - y - xz \\ \frac{dz}{dt} = yx - bz \end{cases} \tag{1.15}$$

Où a est le nombre de Prandtl ; r est le rapport du nombre de Rayleigh sur un Rayleigh critique. Physiquement, r est proportionnel au gradient thermique vertical imposé au fluide ; b est proportionnel à l'élongation de la boite contenant le fluide ; $x(t)$ est proportionnel à l'intensité du mouvement de convection ; $y(t)$ est proportionnel à la différence de température entre les courants ascendants et descendants et $z(t)$ est proportionnel à l'écart du profil de température vertical par rapport à un profil linéaire [**?**].

Cette équation n'admet pas de solution analytique. Pour étudier le comportement du système, on utilise des méthodes d'intégration numérique tel que Runge-Kutta d'ordre 4 . Les conditions initiales étant fixées aux valeurs $(10, \frac{8}{3})$ pour (a, b), Il présente un comportement chaotique pour les valeurs de $r \in [24,75 : 28]$.

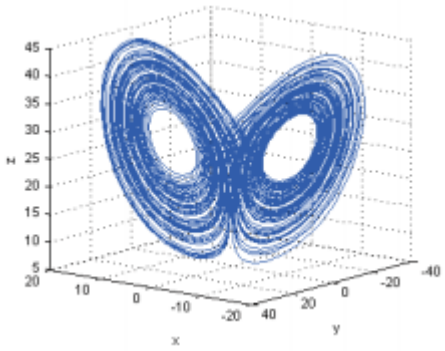


FIGURE 1.9 – L'attracteur du système de Loretz pour $r = 24,75$
;

b. Système de Chua

C'est un circuit RLC composé de quatre composants électroniques (deux capacités, une résistance et une inductance) et d'une diode non linéaire qui présente un comportement chaotique.

Il a été inventé par Léon Chua en 1983 [112]. Ce circuit est modélisé par un système à trois équations différentielles :

$$\begin{cases} \frac{dx}{dt} = c_1(y - x - g(x)) \\ \frac{dy}{dt} = x - y + z \\ \frac{dz}{dt} = -c_2 y \end{cases} \tag{1.16}$$

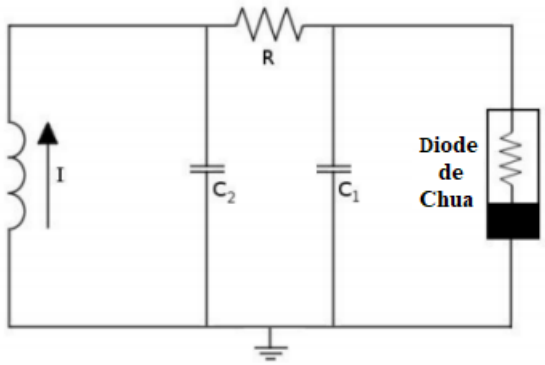


FIGURE 1.10 – Circuit de Chua
;

La fonction $g(x)$ de l'**Eq.1.16** représente la caractéristique courant-tension de la diode non linéaire. Elle est composée de trois parties linéaires, et sa forme dépend de la configuration de ses composants. Les paramètres c_1 et c_2 sont déterminés par les valeurs particulières des composants du circuit. Cette fonction $g(x)$ est définie par :

$$\begin{cases} \frac{dx}{dt} = m_1 x + (m_0 - m_1) \;\; x > 0 \\ \frac{dy}{dt} = m_0 x \;\; |x| \leq 1 \\ \frac{dz}{dt} = m_1 x - (m_0 - m_1) \;\; x \leq -1 \end{cases} \tag{1.17}$$

De même que le système de Lorentz, celui de Chua ne présente pas de solution analytique et Les simulations numériques sont effectuées en utilisant l'algorithme d'intégration numérique de Runge-Kutta d'ordre 4 sous simulateur MATLAB. Pour le paramètre de contrôle $c_1 = 15$ le circuit de Chua présente une évolution chaotique pour les valeurs de $c_2 \in [8,2; 10[$.

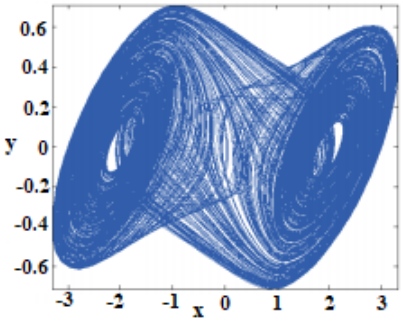


FIGURE 1.11 – L'attracteur du circuit de Chua $c_1 = 9$
;

1.5.2 Système dynamique chaotique discret

L'étude théorique de ces modèles de système dynamique est fondamentale, car elle permet de mettre en évidence des résultats importants, qui se généralisent souvent aux évolutions dynamiques continues. Elle est représentée par le modèle général des équations aux différences finies définies par :

$$X \in \mathbb{R}^d, t \in \mathbb{N}^*,\ X(t+1) = F(X(t), t)$$

Où F représente la fonction d'un système dynamique discret dans le temps, pour lequel les solutions sont obtenues en fixant un état initial $x(t_0)$.

Nous les avons identifiés à partir des noms, équations, dimensions et des plages d'utilisation des paramètres de certains systèmes chaotiques à temps discret les plus utilisés.

TABLE 1.1 – Exemple de systèmes chaotiques à temps discret.

Noms	Équations	Dimensions (d)	Plage des paramètres
Récurrence Logistique	$x(t+1) = \lambda \cdot x(t)(1-x(t))$	1	$\lambda \in [1:4]$
Tent	$x(t+1) = \begin{cases} 2\lambda x(t) & x(t) < 0.5 \\ 2\lambda(1-x(t)) & x(t) \geq 0.5 \end{cases}$	1	$\lambda \in [0:4]$
Skew Tent	$x(t+1) = \begin{cases} \frac{x(t)}{\lambda} & 0 \leq x(t) < \lambda \\ (1-\lambda)(1-x(t)) & \lambda \leq x(t) < 1 \end{cases}$	1	$\lambda \in [0:1]$
Piece Wise Linear Chaotic (PWLC)	$x(t+1) = F[x(t)] \begin{cases} \frac{x(t)}{\lambda} & 0 \leq x(t) < \lambda \\ \frac{x(t)-\lambda}{0.5-\lambda} & \lambda \leq x(t) < 0.5 \\ F[1-x(t)] & 0.5 \leq x(t) < 1 \end{cases}$	1	[0 :1]
Hénon	$\begin{cases} x_1(t+1) = x_2(t) + ax_1(t) + 1 \\ x_2(t+1) = bx_1(t) \end{cases}$	2	$a \in [0; 1,4]$ pour $b = 0,3$
Arnold cat map	$\begin{cases} x_1(t+1) = x_2(t) + \alpha x_1(t) \\ x_2(t+1) = \beta x_1(t+1) + x_2(t) \end{cases} \mod 1$	2	$\alpha, \beta \in \mathbb{R}$

L'étude numérique sur ordinateur des systèmes qui évoluent dans temps discret est facile à réaliser. Car, aucun algorithme de discrétisation numérique n'est utilisé contrairement aux systèmes dynamiques à temps continu. Aussi, il est plus facile de générer des complexes dynamiques dans le cas des systèmes à temps discret que dans le cas de ceux à temps continu.

Étant donné que tous les outils numériques réalisent des opérations sur des grandeurs quantifiées, la plupart de ces systèmes ne sont pas adaptés à la conception des générateurs de nombres pseudo-aléatoires. Généralement, ils évoluent dans un espace réel et la manipulation des valeurs

réelles de précision fixe conduit à des erreurs importantes lors de la production des orbites de grandes tailles. Deuxièmement, la séquence de nombres générés par ces récurrences n'ont pas une distribution uniforme. Enfin, Ils ne présentent une dynamique chaotique que pour des valeurs isolées d'un paramètre. Même de petits écarts par rapport à ces valeurs isolées entraînent la création de sous-régions dans l'espace des phases, c'est-à-dire que l'orbite du point ne couvre pas tout l'espace des phases.

Ainsi, nous allons dans cette thèse utiliser La classe de systèmes dynamiques connue sous le nom de difféomorphisme d'Anosov du tore bidimensionnel. Ces systèmes dynamiques ont reçu beaucoup d'attention dans le contexte de la théorie ergodique. Les systèmes Anosov ont les propriétés probabilistes telles que l'ergodicité, le mélange, la dépendance sensible aux conditions initiales. Ces propriétés sont similaires à certaines propriétés du caractère aléatoire. Chaque difféomorphisme d'Anosov du tore est topologiquement conjugué avec l'automorphisme hyperbolique et peut être considéré comme un système dynamique hamiltonien entièrement chaotique. Les automorphismes hyperboliques sont représentés par des matrices $2 \cdot 2$ avec des entrées entières, des déterminants unités et des valeurs propres réelles. La résolution de ces systèmes ne nécessite aucune opération de discrétisation et de quantification. Ils sont généralement appelés **récurrence de chat d'Arnold**.

1.5.3 Critères d'évaluation d'une fonction Chaotique

Comme critère d'évaluation, une fonction chaotique doit être caractérisée de la manière ci-dessous afin d'être admise comme relativement bonne :

1. **Simplicité** : C'est le premier critère à considérer lors du choix d'un générateur de chaos. En effet, si les itérations chaotiques ne sont pas mathématiquement complexes, les trajectoires peuvent être calculées très rapidement. Sachant que les nombres chaotiques sont utilisés comme clés de chiffrement, ils peuvent être utilisés pour des applications en temps réel. Une autre raison est qu'un système chaotique simple est plus facile à apprendre et sa dynamique peut être modélisée avec plus de précision. En d'autres termes, il permet d'éviter le bruit et les erreurs, ce qui facilite la réduction des potentiels attaques contre les crypto-systèmes.
2. **Comportement aléatoire** : Le processus résultant de la fonction doit être imprédictible. Une séquence de bits est considérée comme aléatoire si la probabilité que chaque bit généré soit $'0'$ ou $'1'$ est de $0,5$. La valeur des éléments suivants dans une séquence ne peut pas être prédite sans tenir compte des valeurs générées précédemment. Les séquences d'images

chiffrées par des crypto-systèmes basés sur des fonctions chaotiques doivent ensuite réussir le test statistique NIST [**?**].

3. **Mise en œuvre** : Une fonction chaotique doit être facile à implémenter. Les techniques cryptographiques peuvent être implémentées dans le logiciel Matlab ou le langage C++, certaines opérations prenant plus de temps que d'autres. Les systèmes cryptographiques, en revanche, sont numériques plutôt qu'analogiques, de sorte que la mise en œuvre de techniques cryptographiques peut se faire, par exemple, dans du matériel DSP, des microprocesseurs dédiés au calcul discret, sous des dispositifs FPGA ou des microcontrôleurs. Ainsi, les ressources limitées des FPGA ou des processeurs numériques (DSP) peuvent compliquer les problèmes de mise en œuvre. Par conséquent, il est souhaitable de choisir un générateur de chaos nécessitant moins de ressources.

1.6 Conclusion

Dans ce chapitre, nous avons fourni un certain nombre de notions concernant les générateurs de nombres pseudo-aléatoires. De cela, il en ressort que les générateurs de nombres pseudo-aléatoires sont gouvernés par des lois déterministes. On distingue les générateurs de nombres pseudo-aléatoires linéaires (LCG, LFSR, LUT, GFSR, TGFSR, Mersenne Twister) et non linéaires. Ces derniers sont basés sur les systèmes dynamiques chaotiques parmi lesquels on distingue les générateurs basés sur le chaos différentiel, récurrent ou sur les itérations chaotiques. A la fin de ce chapitre, nous présentons quelques notions sur les systèmes dynamiques. Afin d'apporter notre contribution à la conception des GNPAs qui réalisent l'équilibrer des compromis entre les exigences de faible coût en ressources, les performances (en débit) et la résistance aux attaques cryptographiques, nous allons au chapitre suivant présenté, un aperçu des fondements mathématiques concernant le GNPA basé sur des récurrences chaotiques, qui sont les principaux objets considérés au cours de notre thèse. En particulier, nous présentons la récurrence du chat d'Arnold en dimension 2, la proposition que nous avons faite sur la généralisation de la récurrence du chat d'Arnold en dimension 2, et une extension en dimension 8 est ensuite déduite du système de dimension 2 que nous avons proposé.

CHAPITRE 2

MATÉRIELS ET MÉTHODE

2.1 Introduction

Dans le chapitre précédent, nous avons passé en revue quelques générateurs de nombres pseudo-aléatoires linéaires et non-linéaires implémentés sur le FPGA. Nous avons remarqué que chaque concepteur essaye de trouver, lors de la mise en œuvre de son générateur, un juste équilibre entre la quantité de ressources matérielles, le débit et les propriétés statistiques des séquences aléatoires générées. Toutes ces caractéristiques sont fortement corrélées au type de système dynamique chaotique utilisé pour l'implémentation du générateur. Dans ce chapitre, des outils mathématiques d'analyse d'aléa que nous avons utilisé seront présentés, par la suite, une description de la récurrence dynamique chaotique discrète que nous proposons dans cette thèse sera présentée. Celle-ci est obtenue en introduisant un terme additif au sein de la récurrence du chat d'Arnold classique, dont le rôle est d'augmenter la période de ses orbites. Enfin, une généralisation de ce système sera effectuée en dimension 8 sur 1 bit.

2.2 Matériels : Les outils de caractérisation d'aléa

Avant d'utiliser les générateurs de nombres pseudo-aléatoires chaotique pour des applications de télécommunication ou dans toutes autres applications, certaines exigences strictes doivent être vérifiées. Pour cela, une série de tests doit être appliquée. L'ensemble des résultats de ces tests donne une idée du degré d'aléa des signaux produits. Parmi ces tests, nous avons : Les tests de l'exposant de Lyapunov ou diagramme de bifurcation qui permettent de fixer les valeurs optimales des paramètres pour atteindre les régions chaotiques des systèmes dynamiques, la sensibilité aux conditions initiales, entropie et aussi les outils traditionnels de traitements numériques du signal (fonction d'auto-corrélation et d'inter-corrélation). Tous les signaux aléatoires qui passent la suite des tests statistiques de NIST sont utilisés pour les applications de cryptographie.

2.2.1 L'exposant de Lyapunov

Certains systèmes dynamiques sont très sensibles aux variations des conditions initiales, et ces variations peuvent rapidement atteindre des proportions énormes. Le mathématicien russe Alexander Markus-Lyapunov $(1857 - 1918)$ a étudié ce phénomène et a développé une quantité qui mesure la vitesse à laquelle ces petites fluctuations augmentent. Cette quantité est appelée "l'exposant de Lyapunov" et elle mesure en fait le degré de sensibilité d'un système dynamique.

L'exposant de Lyapunov [84] est l'une des mesures quantitatives du chaos, et Lyapunov a montré que le nombre d'exposants de Lyapunov est égal à la dimension de l'espace topologique. En outre, Un système non linéaire a un comportement chaotique au sens de Lyapunov si :

1. il a au moins un exposant de Lyapunov positif
2. son plan de phase globalement borné

L'exposant de Lyapunov est utilisé pour prouver qu'un système dynamique peut réaliser un chaos robuste et des comportements dynamiques avec la complexité requise. C'est une mesure, couramment utilisée pour indiquer si le chaos existe ou non [85]. Elle définit le chaos en caractérisant le taux de séparation de deux trajectoires extrêmement proches d'un système non linéaire. Un système dynamique multidimensionnel à plusieurs exposants de Lyapunov qui implique la divergence des trajectoires de plans de phase proches dans différentes directions. Le nombre d'exposants de Lyapunov pour un système dynamique multidimensionnel est égal à la dimension de son plan de phase.

Soit un système dynamique non linéaire définie par :

$$f(x_i, t) : \{x_i(t+1) = f_i(x_i(t), t)\} \tag{2.1}$$

Les exposants de Lyapunov (Λ) de ce système se calcul en utilisant la relation ci-dessous

$$\Lambda_j = \lim_{T \to +\infty} \frac{1}{T} \ln \lambda_j(J) \tag{2.2}$$

Ou $j = 1, 2, ...$ et $\lambda_j(J)$ représentent les valeurs propres de la matrice J avec :

$$J = J(x_i(0)) \cdot J(x_i(1)) \cdot J(x_i(2)) \cdots J(x_i(T-1))$$

. $J(x_i(t))$ est appelé la matrice Jacobienne du système dynamique non linéaire f à l'itération t et

$$J(x_i(t)) = \begin{pmatrix} \frac{\partial f_1(x_i,t)}{\partial x_1} & \cdots & \frac{\partial f_1(x_i,t)}{\partial x_n} \\ \frac{\partial f_2(x_i,t)}{\partial x_2} & \cdots & \frac{\partial f_2(x_i,t)}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_n(x_i,t)}{\partial x_1} & \cdots & \frac{\partial f_n(x_i,t)}{\partial x_n} \end{pmatrix} \tag{2.3}$$

Si un système dynamique a un exposant de Lyapunov positif, alors ses trajectoires proches divergent à chaque unité de temps. Si un système non linéaire peut atteindre deux LE positifs ou plus, alors ses trajectoires proches divergent en plusieurs dimensions. Un exposant de Lyapunov positif plus grand indique que les trajectoires de plan de phase extrêmement proches du système divergent plus rapidement [84].

2.2.2 La sensibilité aux conditions initiales

Les systèmes chaotiques sont connus pour être extrêmement sensibles aux perturbations. Cela est la plupart du temps illustré par l'effet papillon, publier par le météorologue Edward Lorenz, cet effet papillon est vu tel que : on peut considérer qu'un simple battement d'aile de papillon au Cameroun peut entraîner une tempête sur côte de l'Afrique du sud. Entre autre, cela signifie qu'une faible perturbation à l'échelle atmosphérique peut avoir de grandes répercussions. Bien que cela ne soit qu'une vue de l'esprit, elle permet néanmoins de comprendre le phénomène de sensibilité aux perturbations, plus souvent appelé "sensibilité aux conditions initiales".

Soit une application chaotique $f(x,t)$, initialisée par deux valeurs initiales qui sont proches, $x(0)$ et $x(0)+\epsilon$ ou ϵ représente la faible perturbation qui peut être de l'ordre de 10^{-5}. Ces deux valeurs initiales génèrent respectivement deux orbites x et y. L'erreur produite par ces deux orbites s'obtient en appliquant la relation suivante :

$$E(t) = |f(x,t) - f(y,t)| \tag{2.4}$$

Si :

— $E(t) \neq 0$ le système dynamique est chaotique,

— $E(t) = 0$ le système dynamique ne peut pas être qualifié de chaotique

2.2.3 L'entropie

Le concept d'entropie a été introduit au milieu du XIXe siècle par Rudolf Clausius et a été continuellement enrichi, développé et interprété par des chercheurs de nombreuses disciplines scientifiques. Parfois définie comme une mesure du désordre d'un système ou de sa prédictibilité, l'entropie est considérée comme une fonction de l'état du système, augmentant jusqu'à une valeur maximale lorsque le système atteint l'équilibre.

L'entropie de l'information détermine l'imprédictibilité de certains messages et c'est une mesure de l'incertitude du flux binaire aléatoire généré par les différents GNPA proposés. Lorsque la séquence aléatoire est uniformément distribuée, l'entropie est la plus grande. Il est très important dans l'analyse de sécurité, une entropie élevée signifie un générateur pseudo-aléatoire robuste, tandis qu'une entropie faible signifie un générateur pseudo-aléatoire faible avec une certaine prédictibilité.

L'entropie est l'information que le système reçoit en moyenne. Si on appelle la probabilité $p(x_i)$ que le système soit dans l'état x_i avec $1 \leq i \leq 2^n$, alors l'entropie d'une séquence se calcul en utilisant l'**Eq.2.5** :

$$H(x) = -\sum_{i=1}^{2^n} p(x_i) \log_2 p(x_i) \tag{2.5}$$

où n est le nombre de bits sous lequel est codé chaque variable état x_i du système, 2^n est le nombre de symboles possibles dans la séquence. x_i peut prendre 2^n symboles différents dans la séquence x et $p(x_i)$ représente la probabilité de distribution de l'élément x_i dans la séquence x. Si la séquence x a 2^n éléments possibles, l'entropie la plus grande doit être $H(x) = n$.

2.2.4 La corrélation

Cette analyse est une opération importante pour détecter la corrélation entre deux séquences aléatoires de même longueur. Elle a été largement utilisée dans de nombreux domaines de l'ingénierie et des sciences fondamentales, y compris les applications électriques, acoustiques et géophysiques, pour n'en nommer que quelques-unes, où l'isolation du bruit, la similarité spatiale entre les signaux et la détection des caractéristiques sont intéressantes. Cette analyse regroupe deux méthodes qui sont : L'auto-corrélation et l'intercorrélation. L'auto-corrélation, également connue comme la corrélation sérielle, mesure le caractère aléatoire d'une séquence avec une copie retardée d'elle-même comme fonction de retard. L'analyse d'auto-corrélation est un outil mathématique

utilisé pour trouver les motifs répétitifs, tels que la présence d'un signal périodique couvert par des bruits ou l'identification de la fréquence fondamentale manquante dans un signal impliqué par ses fréquences harmoniques. Elle est souvent utilisée dans le traitement d'un signal pour l'analyse des fonctions ou des séries de valeur. La méthode de corrélation croisée, quant à elle permet de vérifier davantage la sensibilité d'une fonction ou d'une méthode de conception à la clé initiale. Elle mesure la corrélation entre deux séquences générées par deux clés adjacentes.

En Statistique, l'auto-corrélation d'un processus aléatoire est la corrélation de Pearson entre les valeurs du processus à différents moments, comme une fonction de deux temps ou d'un écart de temps. L'implémentation numérique du coefficient de corrélation d'un échantillon de Pearson entre deux séquences $X = f(x_0; x_1; ...; x_{L-1})$ et $Y = f(y_0; y_1; ...; y_{L-1})$, est donné par l'équation 2.6. Le résultat de l'analyse de corrélation générale (les coefficients d'autocorrélation (C_{xx} ou C_{yy}) et d'inter-corrélation (C_{xy}) est un nombre compris entre -1 et 1. Une valeur de corrélation positive indique que les signaux variant dans le temps augmentent et diminuent ensemble, et une valeur négative indique une relation inverse (c'est-à-dire qu'un signal augmente tandis que l'autre signal diminue). Avec l'autocorrélation, un signal variant dans le temps $(x(t))$ est corrélé à lui-même. La corrélation croisée consiste à corréler 2 signaux différents variant dans le temps, $x(t)$ et $y(t)$, l'un par rapport à l'autre.

$$C_{xy} = \frac{\sum_{i=1}^{L}(x_i - \bar{x}) \cdot (y_i - \bar{y})}{\sqrt{\sum_{i=1}^{L}(x_i - \bar{x})^2} \cdot \sqrt{\sum_{i=1}^{L}(y_i - \bar{y})^2}} \tag{2.6}$$

Ou $\bar{x} = \sum_{i=1}^{L} \frac{x_i}{L}$, $\bar{y} = \sum_{i=1}^{L} \frac{y_i}{L}$ et L la longueur de la séquence.

Les corrélogrammes sont des outils graphiques utilisés pour vérifier le caractère aléatoire d'un ensemble de données. Ils sont obtenus en calculant le coefficient de corrélation $C_{xy}(r)$ pour cet ensemble de données avec différents écarts de temps r. Ce processus implique un décalage itératif d'un signal vers l'avant et vers l'arrière dans le temps par rapport à l'autre signal qui est maintenu stationnaire avec une valeur de corrélation $C_{xy}(r)$ et $C_{xx}(r)$, calculée à chaque incrément de r.

$$C_{xy}(r) = \frac{\sum_{i=1}^{L-r}(x_i - \bar{x}) \cdot (y_{i+r} - \bar{y})}{\sqrt{\sum_{i=1}^{L}(x_i - \bar{x})^2} \cdot \sqrt{\sum_{i=1}^{L}(y_i - \bar{y})^2}} \tag{2.7}$$

$$C_{xx}(r) = \frac{\sum_{i=1}^{L-r}(x_i - \bar{x}) \cdot (x_{i+r} - \bar{x})}{\sum_{i=1}^{L}(x_i - \bar{x})^2} \tag{2.8}$$

2.2.5 Le test statistique de NIST SP800-22

La suite de tests statistiques NIST [**?**, 18] , SP 800-22, est une suite de test avec 15 sous-tests pour évaluer de manière exhaustive le caractère aléatoire des séquences binaires générées par les GNPAs. Ces sous-tests sont conçus pour trouver les zones non aléatoires de tous les côtés d'une séquence de test. La longueur en bits de la séquence binaire de test est suggérée à 10^8. Le niveau de signification dans NIST SP800-22 est préréglé à $\alpha = 0,01$. Étant donné que la taille de l'échantillon des séquences binaires dans chaque test ne doit pas être inférieure à α^{-1}. Ce qui implique que pour le test, nous devons utiliser 100 séquences binaires d'une longueur de 10^6 bits générées par le GNPA conçu. Chaque sous-test produit une $p-value$ dont les résultats empiriques peuvent être interprétés de trois manières : $P-value$; taux de réussite; et $P-valeur_T$. La valeur P consiste à vérifier si les valeurs P générées par les sous-tests se situent dans la plage de $[\alpha; 1], [0,01; 1]$ dans nos expériences. Le taux de réussite est indiqué par la valeur P. Il prend la proportion statistique des séquences qui réussissent le test de $p-value$ parmi toutes les séquences impliquant le test. Le taux de réussite minimum dans nos expériences est de 0,9602, qui peut être calculé par la méthode de [**?**] avec le niveau de signification du test et la taille de l'échantillon. $P-value_T$ est une statistique de la distribution des valeurs P. Si $P-valueT \geq 0,0001$, les séquences du test sont uniformément distribuées et réussissent le sous-test correspondant.

Cette batterie de tests suppose que les séquences testées ont les propriétés suivantes :

— uniformité (les probabilités d'apparition des bits 0 et 1 sont égales à $0,5$),

— scalabilité (si une séquence est aléatoire, alors ses sous-séquences doivent aussi réussir les tests),

— consistance (un générateur doit avoir le même comportement quel que soit sa graine et ne doit donc pas être testé qu'avec une seule graine).

Les 15 tests de NIST sont :

1. ***Frequence*** :Ce test détermine si les occurrences de bits à 0 et à 1 dans la séquence examinée sont égales. Une séquence vraiment aléatoire aura ces bits en proportions égales (uniformité),

la statistique suit donc une distribution normale. Il détecte aussi l'importance du biais dans la distribution des bits selon que leur proportion d'apparitions se rapproche de $0,5$ ou pas.

2. ***Frequency within a block*** : Ce test est identique au précédent, mais il calcule les occurrences des bits sur des sous-séquences de taille M. Il mesure à quel point cela est vrai grâce à un test de khi deux.
3. ***Runs :*** Ce test détermine si les oscillations entre des runs de bits à 0 ou à 1 sont trop lentes (pour des transitions peu rapprochées, il y a plus de runs) ou trop rapides (moins de runs). Il calcule le nombre total de runs et le compare à sa valeur attendue par un test de khi deux.
4. ***Longest run of ones in a block*** :Ce test examine la longueur maximale des runs de bits à 1 dans des sous-séquences de M bits. Il compare la valeur obtenue à celle attendue pour une séquence aléatoire par un test de khi deux. Une déviation observée sur cette longueur implique la réciproque sur des runs de bits à 0, ceux-ci ne sont donc pas examinés.
5. ***Binary matrix rank*** : Ce test détermine si des dépendances linéaires existent dans la séquence. Des matrices sont formées par les suites de bits dans la séquence testée et leur rang est calculé. Les occurrences des différents rangs possibles sont comparés à celles théoriques pour une séquence aléatoire par un test de khi deux.
6. ***Discrete Fourier transform*** :Le but de ce test est de déterminer si la séquence détient des caractéristiques de périodiques (motifs se répétant) ce qui se traduirait par des valeurs de pics non homogènes à certaines fréquences. Il calcule la DFT (Discrete Fourier Transform) de la séquence et compte le nombre de pics de la DFT dépassant un seuil qu'en théorie 95% des pics ne devraient pas dépasser. Ce nombre est comparé aux 5% de pics dépassant le seuil en temps normal. La statistique suit donc une distribution normale.
7. ***Non-overlapping template matching*** : Le but de ce test est de détecter si la séquence contient un ou plusieurs motifs se répétant trop ou pas assez souvent. Ce test compte les occurrences de toutes les combinaisons possibles d'un motif ayant un certain nombre de bits et les compare à des valeurs théoriques par un test de khi deux. Si le motif n'est pas trouvé, la séquence est examinée en la décalant d'un bit, sinon elle est décalée de m bits.
8. ***Overlapping template matching*** :Ce test est similaire au précédent, à la différence que lorsqu'un motif est trouvé, la séquence n'est ensuite décalée que d'un bit. Il y a donc un recoupement des données.
9. ***Maurer's "Universal Statistical" test*** : Ce test estime la compressibilité sans perte d'information d'une séquence. Pour cela, il compte le nombre de bits entre deux motifs identiques,

une mesure liée à la longueur de la séquence compressée. Il somme le log2 de ces distances et la statistique suit une distribution semi-normale (seulement un côté de la distribution).

10. ***Linear complexity*** :Ce test détermine la complexité de la séquence en la décrivant par des LFSRs. Plus une séquence est aléatoire, plus elle est caractérisée par un LFSR long. En pratique, les occurrences des LFSRs de différentes tailles sont comptées et comparées aux valeurs attendues pour une séquence aléatoire par un test de khi deux.
11. ***Serial*** :Ce test estime l'uniformité de la séquence. Des motifs de m bits sont pris dans la séquence en se décalant d'un bit à chaque fois. Les occurrences de chaque motif sont comptées puis comparées à leurs valeurs théoriques (motifs équiprobables) par un test de khi deux.
12. ***Approximate entropy*** :Ce test estime la régularité de la séquence. Des motifs de m bits et de m- 1 bits sont pris dans la séquence en décalant d'un bit à chaque fois. Leurs occurrences sont comptées et ensuite comparées aux valeurs théoriques attendues par un test de khi deux.
13. ***Cumulative sum*** : Ce test montre si la séquence contient des suites trop importantes de bits à 0 ou à 1 ou au contraire si les bits sont trop bien mélangés. Les bits sont additionnés (un bit à 0 donne la valeur -1 et un bit à 1 donne 1) à mesure qu'ils sont parcourus (du début de la séquence à la fin et vice-versa). La valeur absolue maximale des sommes partielles est retenue, elle doit suivre une distribution normale.
14. ***Random excursions*** : Ce test considère la suite des sommes partielles de la séquence (constituée de bits à 0 et à 1). Quand un bit à 1 est rencontré alors la somme est incrémentée de 1 et s'il s'agit d'un bit à 0 elle est décrémentée de 1. Un cycle est une partie de cette suite débutant et finissant par une somme à 0. Pour chaque cycle, les occurrences des différentes valeurs des sommes (états) sont comptées puis comparées aux valeurs attendues par un test de khi deux.
15. ***Random excursions variants*** :Ce test considère la suite des sommes partielles de la séquence comme précédemment. Les occurrences de chaque état sont comptées sur toute la séquence et doivent suivre une distribution normale.

2.3 Méthodes

2.3.1 Les généralités sur la récurrence du chat d'Arnold (RCA)

a. Description mathématique

La RCA est un système hamiltonien qui évolue dans un espace de phase de dimension 2, ou ces deux variables d'état (position et la quantité de mouvement) sont fortement dépendants l'une de l'autre. Nommée ainsi par le mathématicien russe Vladimir Arnold, il y a deux raisons qui permettent d'expliquer cette terminologie. Premièrement, chat vient de la traduction française du mot Anglais CAT qui désigne un acronyme pour Automorphisme Continu sur un Tore, deuxièmement, le comportement chaotique de cette récurrence est traditionnellement décrit en montrant le résultat de leur action sur le visage du chat [86] comme illustré à la **Fig.2.1** [87].

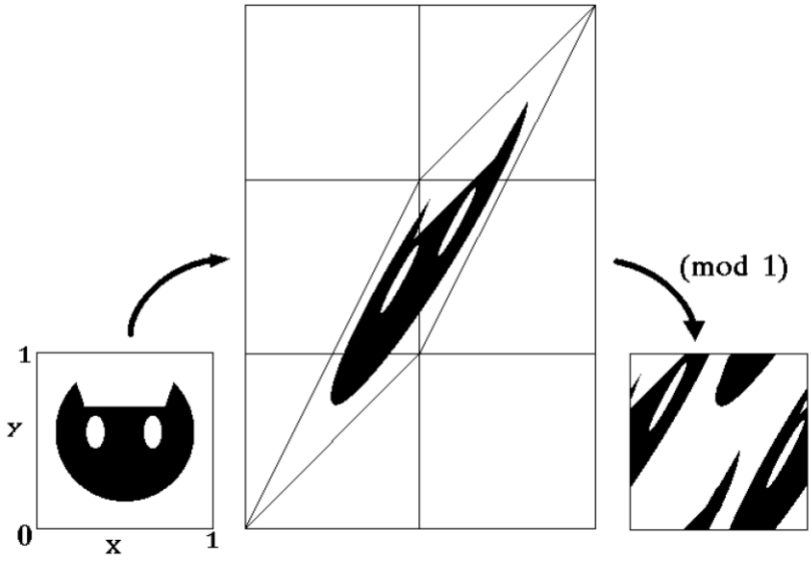


FIGURE 2.1 – Récurrence de la carte du chat d'Arnold

Pour définir la RCA, nous devons d'abord définir un tore et l'espace de phase. Un tore est la surface obtenue en faisant tourner un cercle dans un espace tridimensionnel autour d'un axe déconnecté coplanaire avec le cercle. Un espace de phase représente tous les états possibles d'un système et chaque état correspond à un point unique. La récurrence peut maintenant être définie comme un système discret dans lequel les trajectoires dans l'espace des phases sont étirées et repliées pour obtenir un tore [87].

Largement utilisée dans diverses applications liées à la communication, le tatouage numérique et la cryptographie pour la permutation des pixels d'une image et la génération des nombres pseudo-aléatoire, elle est modélisée sur sa forme générale comme suite [29, 88, 89] :

$$\begin{cases} x(t+1) = x(t) + \alpha y(t) \\ y(t+1) = \beta x(t+1) + y(t) \end{cases} \mod m \tag{2.9}$$

ou : $x(t), y(t)$, sont les variables d'état du système α et β les paramètres de contrôle de la récurrence et m le modulo tels que $x(t), y(t), \alpha$ et $\beta \in [0, 1, 2, ..., m-1]$ et $m \in \mathbb{N}$.

Le système (2.9) peut être réécrit sous sa forme matricielle tel que :

$$X(t+1) = AX(t) \mod m \tag{2.10}$$

Ou :

$$A = \begin{pmatrix} 1 & \alpha \\ \beta & \alpha \cdot \beta + 1 \end{pmatrix}$$

La matrice A est appelée matrice de la RCA. Pour $m = 1$, ce système dynamique discret dans le temps est continue dans son espace de phase tel que : $(x, y) \in [0, 1]^2$. En prenant comme condition initiale $x(0) = 0$ et $y(0) = 0.1$ nous montrons à La **Fig.2.2** le résultat de la représentation graphique de la distribution des points de l'espace de phase système obtenu après 5000 itérations. Cette distribution met en évidence le caractère chaotique de la RCA pour un modulo unité.

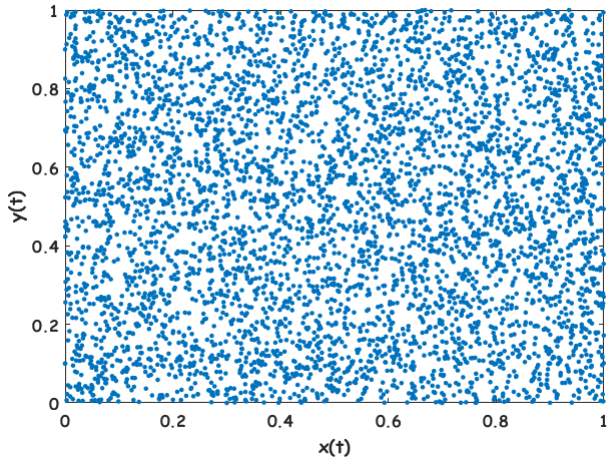


FIGURE 2.2 – Portrait de phase de la récurrence du chat D'Arnold pour $\alpha = \beta = 1$, $x(0) = 0$, $y(0) = 0$ et $m = 1$

b. Propriété de la récurrence du Chat D'Arnold

Parmi les différentes récurrences chaotiques qui ont été développées, la RCA est une récurrence chaotique bidimensionnelle qui possède en plus des propriétés communes des systèmes chaotiques de nombreuses autres propriétés qui la rend uniques.

1. A l'origine $(m = 1)$, elle génère des valeurs réelles et peut être facilement adapté à une précision finie arbitraire juste en modifiant la valeur de m pour une valeur plus grande que 1 dépendant la valeur de la précision de l'application. Permettant ainsi une conception beaucoup plus facile des crypto-systèmes numériques sans utiliser les méthodes de quantification ou de discrétisation des orbites chaotiques.

2. Elle est topologiquement transitive [91] et donc chaotique avec un fort caractère aléatoire.

   Les exposants(σ) de Lyapunov de la récurrence du RCA sont obtenus en calculant les valeurs propres(λ) de sa matrice jacobienne qui correspond à la matrice A :

$$\begin{vmatrix} 1-\lambda & \alpha \\ \beta & (\alpha\cdot\beta+1)-\lambda \end{vmatrix} = \lambda^2 - (\alpha\cdot\beta+2)\lambda + 1 = 0$$

   D'ou :

$$\lambda = 1 + \frac{\alpha\cdot\beta}{2} \pm \frac{1}{2}\sqrt{(\alpha\cdot\beta)^2 + 4\alpha\cdot\beta} \tag{2.11}$$

   Par conséquent, deux exposants de Lyapunov (σ_1, σ_2) sont :

$$\sigma_1 = log_2(\lambda_+) = log_2(1 + \frac{\alpha\cdot\beta}{2} + \frac{1}{2}\sqrt{(\alpha\cdot\beta)^2 + 4\alpha\cdot\beta})$$

$$\sigma_2 = log_2(\lambda_-) = log_2(1 + \frac{\alpha\cdot\beta}{2} - \frac{1}{2}\sqrt{(\alpha\cdot\beta)^2 + 4\alpha\cdot\beta})$$

   En posant par exemple : $\alpha = \beta = 1$, nous avons $\sigma_1 = log_2(\frac{3+\sqrt{5}}{2})$ et $\sigma_2 = log_2(\frac{3-\sqrt{5}}{2})$. σ_1 représente dans ce cas le plus grand exposant de Lyapunov du système. Cet exposant de Lyapunov étant positif, cela implique que la récurrence D'Arnold est un système chaotique.

3. C'est un système dynamique conservatif, car le déterminant de la matrice du chat d'Arnold vaut 1. Ce qui indique que la RCA peut être directement utilisée comme fonction de permutation, ce qui représente un élément important en cryptographie.

4. Elle est inversible et son inverse possède les mêmes caractéristiques que la version directe, car la matrice du chat d'Arnold possède un déterminant unitaire. L'**Eq.2.12** représente La récurrence inverse de la RCA.

$$\begin{cases} y(t) = y(t+1) - \beta x(t+1) \\ x(t) = x(t+1) - \alpha y(t) \end{cases} \mod m \tag{2.12}$$

5. C'est un difféomorphisme d'Anosov et structurellement stable [92], indiquant que de petites perturbations dans le système n'affectent pas les comportements qualitatifs de la trajectoire de la récurrence, et donc la RCA elle-même peut résister à un certain niveau de bruit.

c. L'analyse périodique

Du point de vue des applications pratiques dans les appareils numériques, la récurrence du Chat D'Arnold (2.9) La forme plus importante sous laquelle elle est utilisée est celle pour laquelle $m = 2^n$ (l'anneau de Galois). Car elle est isomorphe à l'ensemble des nombres représentés par le format arithmétique à virgule fixe $n-$bit avec les opérations définies dans la norme pour l'arithmétique des ordinateurs.

Lorsque cette récurrence est utilisée pour le tatouage numérique ou le cryptage d'image,$(x(0), y(0))$ désigne généralement la position initiale d'un pixel dans une image tandis que $(x(t), y(t))$ la position du pixel après la $t-$ième itération de la récurrence. Dans les cryptosystèmes à clé privée et à clé publique, l'ensemble des $(x(0), y(0))$ et t joue le rôle de la clé secrète.

Une discrétisation de la récurrence du Chat D'Arnold continue conduit à la conséquence qu'elle doit être périodique de période Π. Cette période pour un espace de phase donné correspond au plus petit commun multiple des périodes de tous les points de l'espace de phase. C'est-à-dire qu'après Π itération $(x(t+\Pi), y(t+\Pi)) = (x(t), y(t))$.

Cette période a un grand impact sur les applications pratiques. Bien que la valeur de la période exigée dépende de l'application que l'on souhaite réaliser, une grande période est souvent recommandée à des fins cryptographiques. Si la période n'est pas suffisamment longue, les algorithmes conçus pour la protection des données seront vulnérables aux attaques. Par conséquent, la connaissance complète de la distribution de la période de la récurrence du Chat D'Arnold est utile dans la conception et l'analyse des systèmes numériques.

Dans le but de faciliter la recherche de la variété de récurrence du Chat D'Arnold adapter au application cryptographie plusieurs études ont été menés par les chercheurs du domaine.Dans

l'anneau de Galois [29, 89, 93], les différentes formes de représentation possible de la période de la récurrence du chat d'Arnold en fonction de la parité de α et β, sont données par l'**Eq2.13**.

$$\Pi_n = \begin{cases} 2^k \text{ Pour } \alpha \text{ ou } \beta \text{ pair} \\ 3 \cdot 2^{n-2} \text{ Pour } \alpha \text{ et } \beta \text{ impaire} \end{cases} \tag{2.13}$$

Le niveau de sécurité que proposent les algorithmes de chiffrement basé dépend de sa période. Elles sont vulnérables aux attaques si la période de la carte chaotique sous-jacente est courte, c'est-à-dire inférieure aux nombres de points non triviaux de son espace de phase. La forme la plus simple de la récurrence du Chat D'Arnold est celle pour laquelle $(\alpha, \beta) = (1, 1)$. Et pour cette configuration, le problème de période de la RCA peut être transformé en propriétés de divisibilité des nombres de Fibonacci [94]. Ensuite, le théorème connus sur le nom de nombre de Fibonacci est utilisé pour obtenir les bornes supérieure et inférieure de la période de la récurrence du Chat D'Arnold tel que,

$$\log_2(\lambda_+) < \Pi \leq 3m \tag{2.14}$$

Étant donnée que la taille de son espace de phase pour $m = 2^n$ est 2^{2n}, on constat que la plus grande période produite par le récurrence du chat D'Arnold est inférieure à la taille de son espace de phase. Cela implique que dans une orbite produite par ce système dynamique, toutes les valeurs de l'espace de phase ne sont pas représentées. Ce qui est un aspect négatif sur le plan statistique.

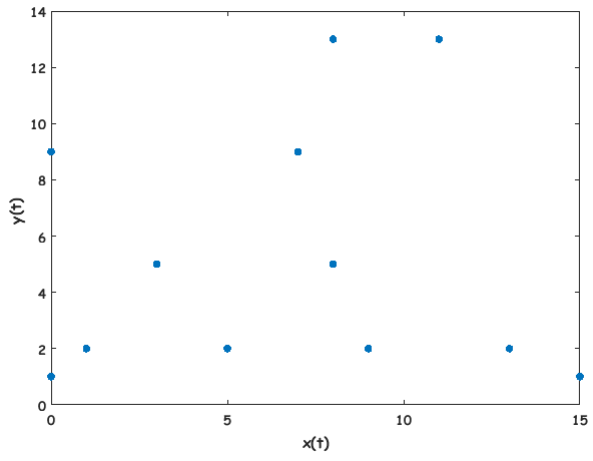


FIGURE 2.3 – Portrait de phase de la récurrence du chat D'Arnold discret pour $\alpha = \beta = 1$, $x(0) = 1$, $y(0) = 5$ et $m = 1$

La **Fig.2.1** représente le portrait de phase de la récurrence du chat d'Arnold dans le domaine des nombres entiers. Nous constatons que sur 256 points que compte l'espace de phase, 12 seulement

sont représentés ce qui fait une différence de 244 points absents de l'espace de phase après 5000 itérations.

2.3.2 La récurrence du chat d'Arnold linéaire par morceau 2D sur 4 bits

a. Synthèse

pour résoudre le problème de faibles périodicités des orbites de la RCA discrète conventionnelle. Nous avons modifié sa structure en introduisant un terme non linéaire de la forme $x+a \bmod c$. Le système obtenu est représenté par l'**Eq.2.15**. Pour limiter sa complexité en évitant les opérations de multiplication, nous avons considéré $((\alpha,\beta)=(1,1))$.

$$\begin{cases} x(t+1) = x(t) + y(t) + \sum_{i=1}^{M}(y(t)+a_i) \bmod c_i \\ y(t+1) = x(t+1) + y(t) + \sum_{i=1}^{M}(x(t+1)+b_i) \bmod d_i \end{cases} \bmod m \tag{2.15}$$

Celui-ci est obtenu en addition le vecteur $X_c(t)$ à l'**Eq.2.10** tel que :

$$X(t+1) = AX(t) + X_c(t) \bmod m \tag{2.16}$$

Avec :

$$X_c(t) = \begin{pmatrix} \sum_{i=1}^{M}(a_i + y(t)) \bmod c_i \\ \\ \sum_{j=1}^{N}(b_j + x(t+1)) \bmod d_j \end{pmatrix} \tag{2.17}$$

Avec : $(i,j) \in \mathbb{N}^2_{\geq 1}$, c_i, d, j sont deux nombres entiers naturels telque $0 \leq c_i < m + a_i$ et $0 \leq d_j < m + b_j$; $0 \leq a_i, b_j < m$ si $(c_i, d_j) = (0,0)$; $0 \leq a_i < c_i$ si $c_i \neq 0$ et $0 \leq b_j < d_i$ si $d_j \neq 0$.

$X_c(t)$ représente le vecteur des termes non-linéaires que nous avons ajoutés à la récurrence d'Arnold conventionnelle. Son rôle est de complexifier sa dynamique en d'augmentant les périodes de ses orbites pour qu'elle soit supérieure aux nombres de points non triviaux de son espace de phase.

Pour faciliter l'analyse du système dynamique obtenu, ce vecteur de perturbation des dynamiques périodiques de la récurrence d'Arnold classique $X_c(t)$ peut se mettre sous la forme : $X_c(t) = (x_c(t), y_c(t))^T$ avec :

$$x_c(t) = \sum_{i=1}^{M} x_{c,i}(t) \tag{2.18}$$

$$y_c(t) = \sum_{j=1}^{N} y_{c,j}(t) \tag{2.19}$$

$$x_{c,i}(t) = (a_i + y(t)) \bmod c_i \tag{2.20}$$

$$y_{c,j}(t) = (b_j + x(t+1)) \bmod d_j \tag{2.21}$$

Les **Eq.2.20** et L'**Eq.2.21** peuvent se mettre sous forme de fonction linéaire par morceau tel que :

$$x_{c,i}(t) = \begin{cases} (a_i + y(t)) & \text{si } (a_i + y(t)) < c_i \\ (a_i + y(t)) - q_i c_i & \text{ailleur} \end{cases} \tag{2.22}$$

Ou $q_i = \lfloor \frac{a_i + y(t)}{c_i} \rfloor$ et

$$y_{c,i}(t) = \begin{cases} (b_j + x(t+1)) & \text{si } (b_j + x(t+1)) < d_j \\ (b_j + x(t+1)) - q_j d_j & \text{ailleur} \end{cases} \tag{2.23}$$

Ou $q_j = \lfloor \frac{b_j + x(t+1)}{d_j} \rfloor$ et en développant l'ensemble des équations, on obtient la récurrence du Chat D'Arnold linéaire par morceau, "piece-wise Linear Arnold Cat Map (PWLCM)" en anglais, définie par :

$$X(t+1) = BX(t) + C(t) \bmod m \tag{2.24}$$

Ou

$$B = \begin{pmatrix} 1 & \alpha' \\ \beta' & \alpha' \cdot \beta' + 1 \end{pmatrix} \tag{2.25}$$

Avec : $\alpha' = \alpha + M$, $\beta' = \beta + N$ et

$$C(t) = \begin{pmatrix} \sum_{i=1}^{M} (a_i - q_i c_i \cdot u(a_i + y(t) - c_i)) \\ \\ \beta' \sum_{i=1}^{M} (a_i - q_i c_i \cdot u(a_i + y(t) - c_i)) + \sum_{j=1}^{N} (b_j - q_j d_j \cdot u(b_j + y(t) - d_i)) \end{pmatrix} \tag{2.26}$$

$u(t)$ représente la fonction d'Heaviside définie par :

$$u(t) = \begin{cases} 0, & \text{si } t < 0 \\ 1, & \text{aillieur} \end{cases} \tag{2.27}$$

La nouvelle généralisation de la récurrence du chat d'Arnold ainsi obtenue présente un terme conservatif linéaire $BX(t)(\det(B) = 1)$ qui possède la même dynamique chaotique que celle de la récurrence du chat d'Arnold conventionnelle, et un terme non linéaire $C(t)$ qui contribue à l'augmentation de la période du terme linéaire en modifiant sa trajectoire pour chaque condition initiale. a_i et b_j, c_i et d_j sont définis comme les paramètres de perturbation, pour les valeurs de $b_j = d_j = 1$ nous retrouvons la récurrence du chat d'Arnold conventionnelle. C'est pour cela que le système dynamique que nous avons obtenu, représente une forme généralisée de celle-ci.

De même que la récurrence du chat d'Arnold classique, la PWLCM est un système dynamique inversible et la forme généralisée de la PWLCM inverse est obtenue en calculant $x(t)$ et $y(t)$ dans l'équation (2.16) tels que :

$$\begin{cases} y(t) = y(t+1) - \beta x(t+1) - \sum_{j=1}^{N} (b_j + x(t+1)) \bmod d_j \\ x(t) = x(t+1) - \alpha y(t) - \sum_{i=1}^{M} (a_i + y(t)) \bmod c_i \end{cases} \bmod m \tag{2.28}$$

$x(t+1)$ et $y(t+1)$ deviennent les conditions initiales du système inverse, $x(t)$ et $y(t)$ les valeurs obtenues après itération. En inversant le sens de l'évolution temporelle, l'équation (2.28) s'écrit :

$$\begin{cases} y(t+1) = y(t) - \beta x(t) - \sum_{j=1}^{N} (b_j + x(t)) \bmod d_j \\ x(t+1) = x(t) - \alpha y(t+1) - \sum_{i=1}^{M} (a_i + y(t+1)) \bmod c_i \end{cases} \bmod m \tag{2.29}$$

La relation (2.29) représente la PWLCM inverse généralisée.

Du point de vue des applications réelles dans les dispositifs numériques, le cas de l'anneau de Galois $\mathbb{Z}_{2^n}$,$(m = 2^n)$ est le plus important, car il est isomorphe à l'ensemble des nombres représentés par le format arithmétique à virgule fixe sur n-bit avec les opérations définies dans la norme pour l'arithmétique des ordinateurs. Sachant que la plupart de nos données numériques sont codées sur 8 bits, le nombre de bit nécessaire pour les chiffrées au travers d'un système dynamique de dimension 2 est $n = 4$, car la concaténation de deux valeurs binaires conduit à une valeur binaire de 8-bits. Pour mettre en évidence la différence considérable qui existe

entre la dynamique chaotique produite par la nouvelle récurrence du chat d'Arnold généralisée et la récurrence du chat d'Arnold conventionnelle, nous les avons appliqués sur un espace de phase de taille $2^n \cdot 2^n$ avec $n = 4$. Le résultat est présenté à la **Fig.2.4**. D'après cette figure, la récurrence du chat d'Arnold conventionnelle présente une dynamique (**Fig.2.4(a)**) périodique de très faible période avec $a = [0, 0], c = [1, 1], b = [0, 0], d = [1, 1]$ par contre, la dynamique du système que nous avons obtenue présente une évolution presque aléatoire(**Fig.2.4(b)**) avec $a = [1, 1], c = [5, 7], b = [1, 0], d = [1, 11]$. C'est ce qui fait de la PWLCM, un meilleur candidat pour la génération des nombres pseudo-aléatoires que la version conventionnelle.

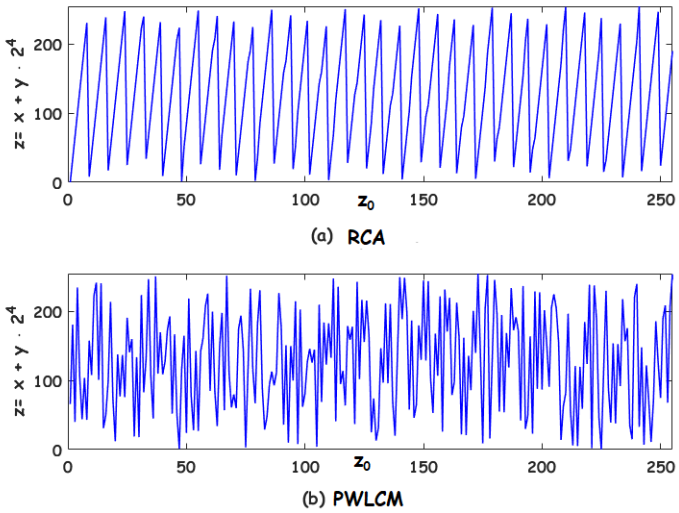


FIGURE 2.4 – L'évolution de dynamique de la récurrence du chat d'Arnold conventionnelle (a) et de la PWLCM (b) dans un espace de phase de précision 4-bits

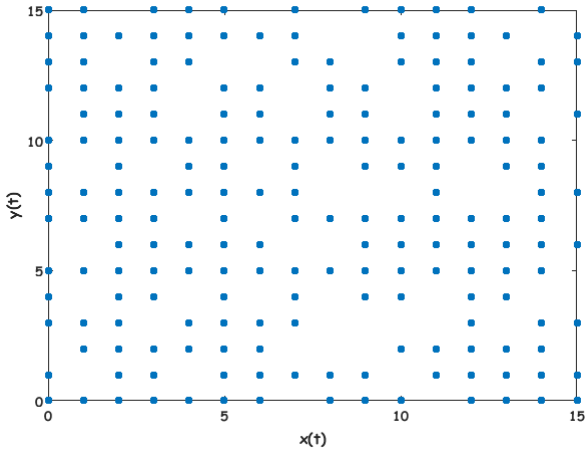


FIGURE 2.5 – Le portrait de phase de la PWLCM dans un espace de phase codé sur $n = 4$-bits

b. L'étude de Stabilité

Dans cette partie, nous étudions la stabilité de la PWLCM. Le jacobien fonctionnel du système (2.3) défini par (1.3) s'écrit :

$$J = \begin{pmatrix} 1 & \alpha' - \sum_{i=1}^{M} c_i \delta_i(y(t) - \tau_y^i) \\ \beta' - \sum_{j=1}^{N} d_j \delta_i(x(t+1) - \tau_x^j) & 1 + (\alpha' - \sum_{i=1}^{M} c_i \delta_i(y(t) - \tau_y^i))(\beta' - \sum_{j=1}^{N} d_j \delta_i(x(t+1) - \tau_x^j)) \end{pmatrix} \tag{2.30}$$

Ou $\tau_y^i = a_i - c_i$ et $\tau_x^j = d_j - b_j$.En effet $q_i = 1$ quand $\delta_i(y(t) - \tau_y^i) = 1$ et $q_j = 1$ quand $\delta_i(x(t+1) - \tau_x^j) = 1$. De cette matrice jacobienne, nous déduisons les valeurs propres

$$\Lambda_1(t) = 1 + \frac{1}{2}\left((\alpha' - \sum_{i=1}^{M} c_i \delta_i(y(t) - \tau_y^i))(\beta' - \sum_{j=1}^{N} d_j \delta_i(x(t+1) - \tau_x^j))\right) \left(1 + \sqrt{1 + \frac{4}{(\alpha' - \sum_{i=1}^{M} c_i \delta_i(y(t) - \tau_y^i))(\beta' - \sum_{j=1}^{N} d_j \delta_i(x(t+1) - \tau_x^j))}}\right) \tag{2.31}$$

$$\Lambda_2(t) = 1 + \frac{1}{2}\left((\alpha' - \sum_{i=1}^{M} c_i \delta_i(y(t) - \tau_y^i))(\beta' - \sum_{j=1}^{N} d_j \delta_i(x(t+1) - \tau_x^j))\right) \left(1 - \sqrt{1 + \frac{4}{(\alpha' - \sum_{i=1}^{M} c_i \delta_i(y(t) - \tau_y^i))(\beta' - \sum_{j=1}^{N} d_j \delta_i(x(t+1) - \tau_x^j))}}\right) \tag{2.32}$$

Le déterminant de $J = 1$, ce qui implique la somme des exposants de Lyapunov $\lambda_1(t)$ et $\lambda_2(t)$ est nulle, donc $\Lambda_1(t) \geq 1$ et $0 \leq \Lambda_2(t) \leq 1$. La PWLCM est donc un système conservatif de même que la RCA discrète, indépendant des valeurs de ses paramètres de contrôle a_i, c_i, b_j et d_j. Sachant que $\Lambda_{1,2}(t) \geq 0$, nous pouvons conclure que la PWLCM est un système dynamique instable.

c. La sensibilité aux conditions initiales

Dans cette partie, nous allons discuter de la période et du plus grand exposant de Lyapunov de la PWLCM en fonction de ses paramètres de contrôle et de ses conditions initiales. Pour une précision donnée, chaque condition initiale donnée génère une dynamique périodique donc la

période peut être déterminée. La période de la PWLCM correspond alors au plus petit commun multiple (ppcm) de l'ensemble des périodes produites par chaque condition initiale de l'espace de phase. Le nombre de conditions initiales est directement lié à la précision n de l'espace de phase, donc la taille est $2^n \cdot 2^n$. Par exemple, pour $n = 1$ nous avons comme condition initiale $(0, 0)$, $(0, 1)$, $(1, 0)$ et $(0, 0)$.

d. La sensibilité de l'exposant de Lyapunov

Il est question dans cette partie d'évaluer les exposants de Lyapunov de la PWLCM pour chaque condition initiale avec les trois réglages de paramètres que nous allons comparer au plus grand exposant de Lyapunov de la RCA discrète correspondante, c'est-à-dire

$$\lambda_0(\alpha^{'}, \beta^{'}) = log(1 + \frac{\alpha^{'} \cdot \beta^{'}}{2} + \frac{1}{2}\sqrt{(\alpha^{'} \cdot \beta^{'})^2 + 4\alpha^{'} \cdot \beta^{'}}) \tag{2.33}$$

Nous avons considéré 20000 itérations du PWLCM pour évaluer les exposants de Lyapunov. Pour le premier paramètre, le tableau 2.1 montre les plus grands exposants de Lyapunov correspondants du PWLCM qui doivent être comparés à $\lambda_0(3, 3) = 2,3895$. Il ressort de ce tableau que le fait de fixer $c_i \neq 0$ et $d_j \neq 0$ contribue à réduire l'exposant de Lyapunov, alors qu'il contribue à augmenter la période du système.

Le deuxième paramétrage conduit au tableau 2.2, avec des valeurs à comparer à $\lambda_0(3, 3) = 2,3895$. Nous pouvons confirmer que l'exposant de Lyapunov ne dépend pas de a_i et b_j dans le cas $c_i = d_j = 0$.

Le troisième paramétrage conduit au tableau 2.3, avec des valeurs qui sont également à comparer à $\lambda_0(3, 3) = 2,3895$. Ce tableau comparé au tableau 2.2 montre que le fait de fixer $c_i = 0$ ou $d_j = 0$ réduit la valeur de l'exposant de Lyapunov, tout en augmentant la période du PWLCM.

La combinaison des observations faites à partir de ces trois tables implique qu'il n'y a pas de relation directe entre l'exposant de Lyapunov et la période du système. De plus, un grand exposant de Lyapunov dans ce cas n'induit pas une grande complexité de la PWLCM, car il correspond à la plus petite période. Cependant, nous observons que la période du système augmente avec la diversité ou la variabilité de l'exposant de Lyapunov. En effet, une grande sensibilité de l'exposant de Lyapunov aux conditions initiales contribue à augmenter la période du système. La figure 2.6 montre le comportement de l'exposant de Lyapunov dans le cas où $n = 4$ pour les trois réglages de paramètres ci-dessus.

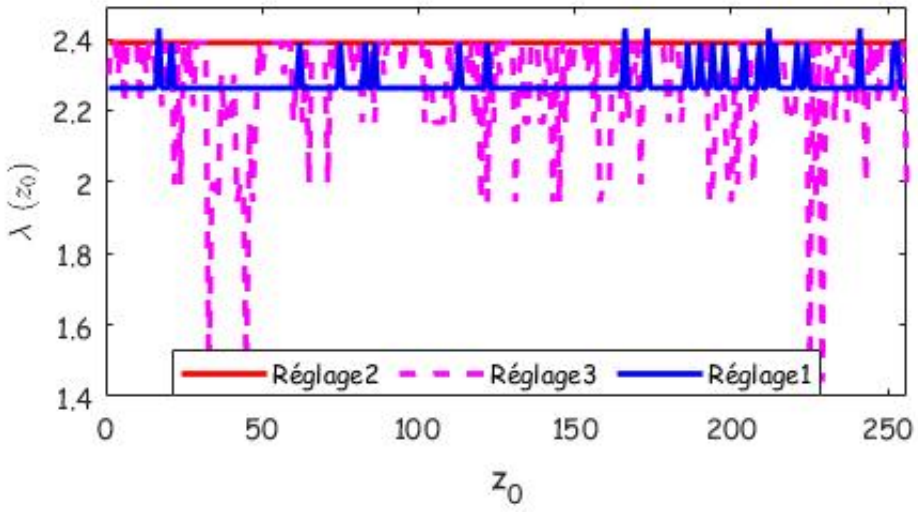


FIGURE 2.6 – Dynamique des exposants de Lyapunov pour les conditions initiales respective $z_0 = 2^n x_0 + y_0$, pour $n = 4$ et a_i, b_i, c_i et d_i suivant celle du tableau 2.3 (réglage1),du tableau 2.1 (réglage3) et du tableau 2.2 (réglage2). Les périodes correspondent respectivement à $T_1 = 9320$, $T_3 = 2520$, $T_2 = 12$. Les exposants de Lyapunov sont évalues après 1000 itérations de PWLCM

TABLE 2.1 – Matrice du plus grand exposant de Lyapunov individuel des différentes conditions initiales du PWLCM pour $n = 2$, $\alpha = \beta = 1$, $M = N = 2$, $c_1 = 0, c_2 = 3$, $d_1 = 3, d_2 = 5$, $a_1 = 1$, $a_2 = 1$, $b_1 = 0, b_2 = 2$, $\lambda_0(3,3) = 2,3895$

x_0/y_0	0	1	2	3
0	1.6777	1.2718	1.6778	1.2719
1	1.6778	2.4554	1.2721	1.6778
2	1.6778	2.4554	1.2720	2.3895
3	2.4554	1.2719	1.6777	1.6777

TABLE 2.2 – Matrice du plus grand exposant de Lyapunov individuel des différentes conditions initiales du PWLCM pour $n = 2$, $\alpha = \beta = 1$, $M = N = 2$, $c_1 = c_2 = 0$, $d_1 = d_2 = 0$, $a_1 = 0$, $a_2 = 1$, $b_1 = b_2 = 0$, $\lambda_0(3,3) = 2,3895$

x_0/y_0	0	1	2	3
0	2.3895	2.3895	2.3895	2.3895
1	2.3895	2.3895	2.3895	2.3895
2	2.3895	2.3895	2.3895	2.3895
3	2.3895	2.3895	2.3895	2.3895

TABLE 2.3 – Matrice du plus grand exposant de Lyapunov individuel des différentes conditions initiales du PWLCM pour $n = 2$, $\alpha = \beta = 1$, $M = N = 2$, $c_1 = 0$, $c_2 = 3$, $d_1 = d_2 = 0$, $a_1 = 0$, $a_2 = 1$, $b_1 = b_2 = 0$, $\lambda_0(3,3) = 2,3895$

x_0/y_0	0	1	2	3
0	2.3895	2.3895	0.0006	2.3895
1	1.8542	1.8542	1.8542	2.3895
2	1.8542	1.8542	1.8542	1.8542
3	2.3895	1.8542	1.8542	1.8542

Dans le premier cas (réglage 1), il y a 187 valeurs distinctes de λ, cinq valeurs de période distinctes $(T = 1, 4, 5, 8, 233)$, et la période correspondante du PWLCM est $T_1 = 9320$; dans le second cas, on obtient une seule valeur de λ et quatre valeurs de période distinctes $(T = 1, 3, 6, 12)$ conduisant ainsi à $T_2 = 12$; tandis que dans le troisième cas (réglage 3), il y a 63 valeurs distinctes de λ et 9 périodes distinctes $(T = 1, 2, 4, 8, 10, 12, 14, 18, 40)$, ce qui correspond à $T_3 = 2520$. On peut observer que dans le cas d'un seul exposant de Lyapunov, la plus grande valeur de période est un multiple des autres valeurs, ce qui contribue à réduire la période de l'ensemble du système.

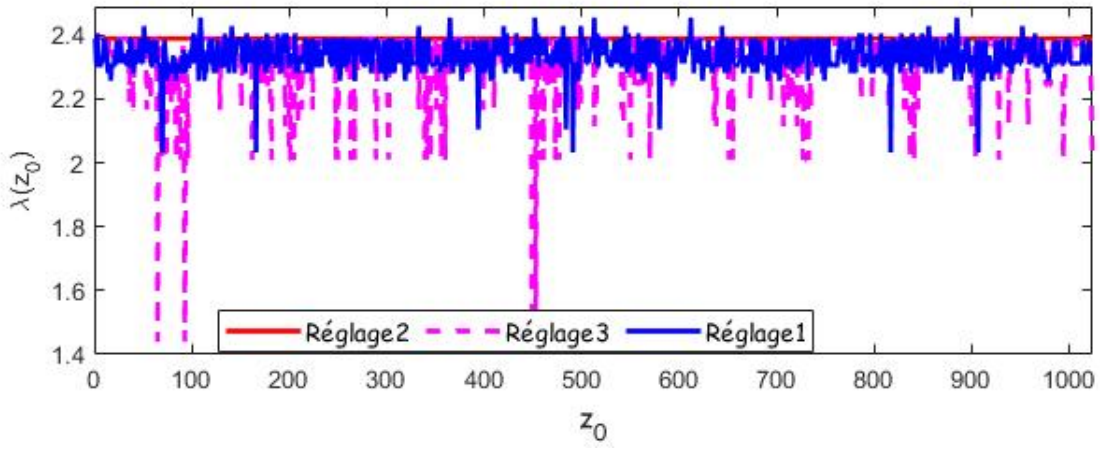


FIGURE 2.7 – Dynamique des exposants de Lyapunov pour les conditions initiales respective $z_0 = 2^n x_0 + y_0$, pour $n = 4$ et a_i, b_i, c_i et d_i suivant celle du tableau 2.3 (réglage1),du tableau 2.1 (réglage3) et du tableau 2.2 (réglage2). les périodes correspondent respectivement à $T_1 = 1,51 \cdot 10^{15}$, $T_3 = 1,02 \cdot 10^{10}$, $T_2 = 24$. Les exposants de Lyapunov sont évalues après 1000 itérations de PWLCM

Une telle observation implique également une forte sensibilité de la période à la précision n. En effet, pour un paramétrage avec une forte sensibilité de l'exposant de Lyapunov aux conditions initiales, la diversité des valeurs des exposants de Lyapunov augmente avec n, comme le nombre de conditions initiales lui-même augmente. En réglant $n = 5$ pour les réglages de paramètres ci-dessus, les périodes deviennent respectivement $T_1 = 1,51 \cdot 10^{15}$, $T_2 = 24$ et $T_3 = 1,02 \cdot 10^{10}$. Comme le montre la figure 2.7, les valeurs du plus grand exposant de Lyapunov sont dans la même plage $(1.4, 2.5)$, mais le nombre de valeurs distinctes a sensiblement augmenté, 450 pour le premier paramètre (reglage 1), et 163 pour le troisième (reglage 3). Il n'y a pas de changement pour le deuxième paramétrage, car la matrice jacobienne correspondante ne dépend pas des conditions initiales.

e. La sensibilité aux paramètres de contrôle a_i et b_j

Nous analysons dans cette partie l'impact des valeurs de a_i et b_j sur la dynamique périodique de la PWLCM. Nous avons considéré pour cette étude : $M = N = 1$, $n = 3$, $\alpha = \beta = 1$, $c_1 = 3$ et $d_1 = 5$. Les périodes obtenues ont été inscrites dans le tableau 2.4 ou nous observons une grande sensibilité du système pour les différentes combinaisons de paramètre de contrôles (a_i, b_j). Ces périodes ont été comparés à $\Pi = 6$ qui correspond à la période de la RCA discrète.

TABLE 2.4 – Sensibilité de la période de la Π_{PWLCM} en fonction de a_i et b_i pour $n = 3$, $M = N = 1$, $\alpha = \beta = 1$, $c_1 = 3$, $d_1 = 5$ et différentes valeurs de (a_1, b_1)

a_1/b_1	0	1	2	3	4
0	504	108	60	252	45
1	63	456	72	120	462
2	350	429	252	13566	12558

f. La sensibilité aux paramètres de contrôle c_i et d_j

Le tableau 2.5 montre que la période de la PWLCM est sensible au changement des paramètres de contrôle c_1 et d_1 pour $n = 2$, $M = N = 1$, $\alpha = \beta = 1$, $a_1 = 1$ et $b_1 = 3$. Il apparait dans ce tableau que la période maximale est obtenue pour $(c_1, d_1) = (1, 6)$. Ce tableau montre qu'avec un très petit nombre de bits $(n = 2)$ il est possible d'obtenir de grandes périodes. Ce qui n'est pas le cas avec la RCA discrète conventionnelle.

TABLE 2.5 – Sensibilité de la période de la Π_{PWLCM} en fonction de a_i et b_i pour $n = 3$, $M = N = 1$, $\alpha = \beta = 1$, $c_1 = 3$, $d_1 = 5$ et différentes valeurs de (c_1, d_1)

c_1/d_1	0	1	2	3	4	5	6
0	2	4	4	16	2	6	4
1	4	3	12	70	4	105	12
2	4	12	6	16	4	14	8
3	6	28	14	8	6	36	14

En effet, la période de la PWLCM dépend des paramètres de contrôles a_i,b_j,c_i,d_j.

La PWLCM que nous proposons est un système dynamique conservatif donc l'évolution dépend de ses paramètres de contrôles. Etant donné qu'il possède toutes les propriétés de la QACM, il peut être considérer comme étant une forme généralisée de la RCA discrète conventionnelle qui produit des dynamique de grandes périodes. Suivant la parité de n, nous proposons l'expression générale des paramètres de contrôle permettant d'obtenir des dynamiques de grandes périodes tels que :

$$c_i(resp.d_j) = \begin{cases} n+2, & \text{si } n = 2p+1 \\ 2^{n+1} - (n+1), & \text{si } n = 2p \end{cases} \tag{2.34}$$

$$a_i(resp.b_j) = \begin{cases} p, & \text{si } n = 2p+1 \\ 2^{n+1} - (2^n+1), & \text{si } n = 2p \end{cases} \tag{2.35}$$

Avec : $p \in \mathbb{N}_{\geq 1}$

g. La sensibilité à la précision de calcul n

Lorsqu'une récurrence chaotique est utilisée pour des applications numériques, elle doit être numérisée avec une longueur de mot de n bits, chaque état $x(t)$ ne peut prendre que 2^n valeurs différentes. Raison pour laquelle chaque orbite chaotique finira par être périodique [101], c'est-à-dire qu'elle passera finalement à un cycle de longueur limitée ne dépassant pas 2^n.

La vue schématique d'une orbite typique d'un système chaotique numérique est illustrée à la **Fig.2.8**. Généralement, chaque orbite chaotique numérique comprend deux parties connectées : $x_0, x_1, ..., x_{l-1}$ et $x_l, x_{l+1}, ..., x_{l+L}$, qui sont respectivement appelés transitoires et cycles. En conséquence, l et L sont respectivement appelées longueur transitoire et période du cycle, et $l + L$ est appelée longueur d'orbite.

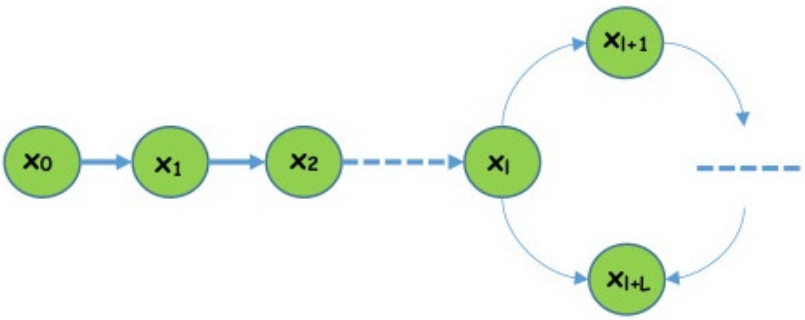


FIGURE 2.8 – Représentation générale de l'orbite d'un système chaotique numérique

Bien que la période maximale soit de 2^n, on trouve généralement des périodes beaucoup plus courtes, de l'ordre de $\sim 2^{n/2}$. Ces séquences de courte période échouent à la plupart des tests aléatoires du NIST [33]. De plus, si un grand nombre de nombres aléatoires étaient nécessaires pour le traitement du signal ou la simulation, la séquence commencerait à se répéter, ce qui pourrait

affecter les résultats de la simulation.

Compte tenu de l'effet de la précision finie des systèmes chaotiques numériques, certaines solutions ont été proposées pour lutter contre la dégradation des systèmes chaotiques numériques. La première consiste à utiliser une grande précision finie [102]. Wheeler et Matthews [103] ont proposé que l'utilisation de superordinateurs et d'autres dispositifs matériels avec des précisions de calcul plus grandes augmenterait la période de la séquence chaotique numérique, mais ne pourrait pas résoudre le problème essentiel. La seconde consiste à cascader plusieurs systèmes chaotiques [104–107]. Plusieurs systèmes chaotiques identiques ou différents sont mis en cascade pour prolonger la période d'une séquence chaotique numérique et résister à la prévisibilité [18]. Il est plus imprévisible et plus complexe par rapport au système chaotique d'origine et a un espace de paramètres plus grand.

Dans le but d'étudier la dynamique du système proposé en fonction de la précision de calcul, Nous avons réalisé l'analysé numérique de sa période pour $2 \leq n \leq 10$ et l'avons comparée à celle de la récurrence du chat d'Arnold correspondant. Les résultats obtenus sont résumés dans le **Tab.2.6**, à partir duquel nous pouvons confirmer l'efficacité de l'élément non linéaire pour augmenter la période de la récurrence du chat D'Arnold. Le terme non-linéaire que nous avons introduit augmente de façon exponentielle sa période par rapport à la précision n.

TABLE 2.6 – Dépendance de la période à la précision pour $a_1 = a_2 = 0, c_1 = 0, c_2 = 11$ et $\alpha^{'} = 3$, $\beta^{'} = 1$

n	π_{RCA}	π_{PWLCM}
2	3	6
3	6	6
4	12	$4.81 \cdot 10^{12}$
5	24	$1.06 \cdot 10^{15}$
6	48	$1.03 \cdot 10^{37}$
7	96	$5.49 \cdot 10^{53}$
8	192	$4.28 \cdot 10^{114}$
9	348	$1.96 \cdot 10^{260}$
10	768	$1.09 \cdot 10^{513}$

Les cas $n = 2$ et $n = 3$ correspondent aux périodes de la RCA avec $\alpha = 3$ et $\beta = 1$, car $c2 > 2^n$, tandis que les périodes des cas $4 \leq n \leq 10$ sont évaluées à l'aide du logiciel Maple pour éviter les erreurs de calcul dans Matlab. Un exemple de distribution des périodes d'orbite $T(x0, y0)$ par rapport à la condition initiale $(x0, y0)$ pour le cas $n = 8$ est représenté sur la Figure 2.9. On

peut observer qu'il existe certaines conditions initiales pour lesquelles la période est supérieure à la plus grande des périodes orbitales du RCA, c'est-à-dire $\pi = 2^n$.

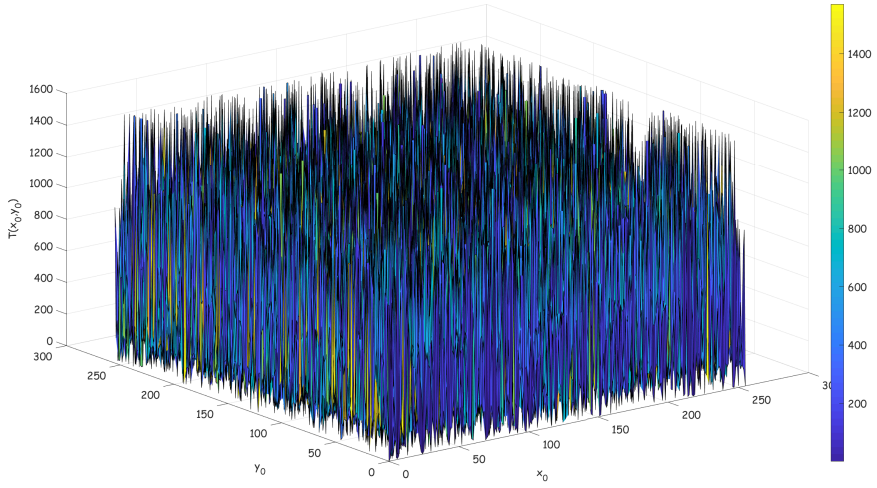


FIGURE 2.9 – Distribution des périodes des orbites de la PWLCM $T(x_0, y_0)$ pour chaque condition initiale avec $a_1 = 0, a_2 = 0, c_1 = 0, c_2 = 11$ et $\alpha^{'} = 3$, $\beta^{'} = 1$.

La figure 2.10 montre la distribution de probabilité de $T(x0, y0)$ pour $n = 6, 7$ et 8. Il ressort de cette figure que la fréquence ou probabilité augmente avec la période : la probabilité la plus élevée correspond à la plus grande période $T(x0, y0)$. Un tel résultat est intéressant, car notre objectif est d'obtenir de grandes périodes pour tous les points non triviaux du PWLCM.

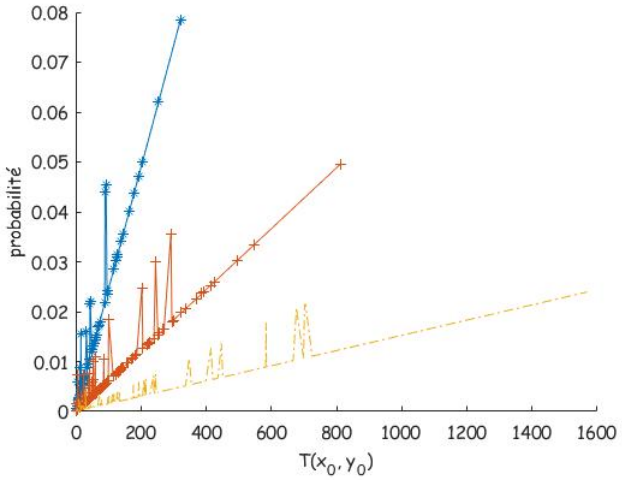


FIGURE 2.10 – Distribution des probabilités de la PWLCM $T(x_0, y_0)$ pour chaque conditions initiales avec $a_1 = 0, a_2 = 0, c_1 = 0, c_2 = 11$ et $\alpha^{'} = 3$, $\beta^{'} = 1$

Dans le but de comparer l'éffet de la présicion sur la propriété de mélange du système que nous proposons PWLCM à la RCA discret, nous avons appliqué le test statistique NIST800-22 à une image de taille $2^{13} \cdot 2^{13}$ mélangée avec les deux systèmes. L'image périodique est obtenue en répétant des séquences d'entiers codés sur 8 bits allant de 0 à 255. Un tel ensemble de données peut être divisé en 50 flux binaires de longueur 10^6 chacun. Le test NIST a été appliqué aux images mélangées, après 50 itérations. Les paramètres PWLCM ont été définis comme $a_1 = 0, a_2 = 0$, $c_1 = 0$, $c_2 = 11$ et $d_1 = d_2 = 1$, et le nombre de bits était $n = 13$ pour les deux systèmes. Les résultats correspondants sont présentés dans le **Tab.2.7**, d'où il en ressort que l'image mélangée PWLCM réussit le test NIST, tandis que l'image RCA discrète échoue. Le PWLCM mélange donc mieux les pixels de l'image que la RCA discret, donc adapté au brassage d'images.

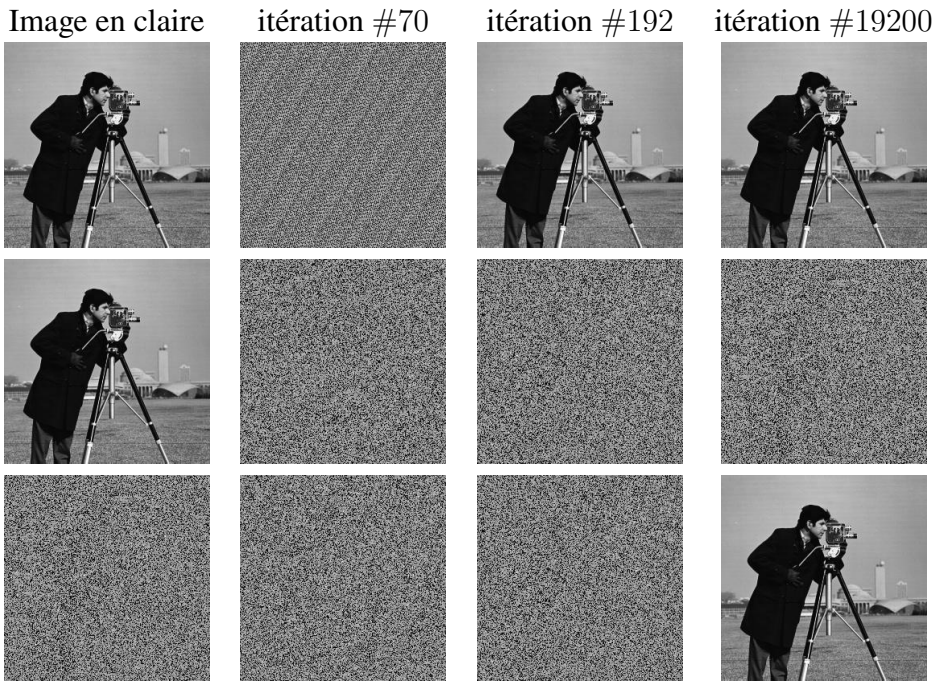


FIGURE 2.11 – Mixage des images à l'aide des transformations RCA discret et PWLCM. La première ligne montre les résultats du RCA discret, la deuxième ligne représente les résultats du PWLCM, tandis que la troisième ligne montre l'image inversée obtenue à partir du PWLCM inversé.

La **Fig.2.11** illustre le réarrangement d'image périodique utilisant à la fois RCA discret 8 bits et PWLCM sur 8 bits. La période du RCA discret dans ce cas est de 192, raison pour laquelle a cette itération l'image initiale réapparaît , tandis que celle du PWLCM, avec $a_1 = 0, a_2 = 0, c_1 = 0, c_2 = 11$ et $d_1 = d_2 = 1$, est de $4,2810^{114}$. Sachant que cette valeur étant très grande et difficile à atteindre ,l'application du système inverse à l'image brassée avec le même nombre d'itérations permet d'obtenir celle originale sans avoir besoin d'exécuter le processus de brassage sur toute la période du système.

TABLE 2.7 – Les résultats du test de NIST 800-22 :

	RCA discret		PWLCM	
Statistical test	P-value	Proportion %	P-value	Proportion %
Frequency	0.0	7/50	0.3505	50/50
Block-frequency	0.0	0/50	0.6993	49/50
Cumulative-sums (forward)	0.0	0/50	0.3191	50/50
Cumulative-sums (reverse)	0.0	0/50	0.1538	50/50
Runs	0.0	5/50	0.6163	49/50
Longest-runs	0.0	0/50	0.5749	49/50
Rank	0.0	0/50	0.1223	49/50
FFT	0.0	0/50	0.1719	48/50
Non-overlapping-templates	0.0	0/50	0.9114	50/50
Overlapping-templates	0.0	0/50	0.6579	49/50
Universal	0.0	0/50	0.3191	50/50
Approximate entropy	0.0	0/50	0.4190	49/50
Random-excursions	0.0	0/50	0.3505	36/50
Random-excursions variant	0.0	0/36	0.8044	36/50
Serial	0.0	0/36	0.6163	50/50
Linear-complexity	0.0	0/50	0.5749	50/50

2.3.3 La récurrence du chat d'Arnold linéaire par morceau monobit en 8D

Les systèmes chaotiques utilisés dans le PRNG peuvent être grossièrement divisés en deux catégories, à savoir le système chaotique unidimensionnel et le système chaotique multidimensionnel. Comparé au système chaotique unidimensionnel, le système chaotique multidimensionnel présente plusieurs avantages :

— **Un espace de clé plus grand** : Ceci augmente avec la dimension et sont utilisés comme clés de chiffrement/déchiffrement des différents cryptosystèmes.

— **Une vitesse de diffusion plus rapide** : Cela signifie que les récurrences chaotiques de grande dimension rendent les textes en clair plus confus que les récurrence chaotiques à faible dimension après les mêmes temps de permutation.

— **Champs d'application plus larges** : Par rapport aux récurrences $2D$ proposées par Fridrich par exemple, les récurrences du Chat D'Arnold en $3D$ peuvent réaliser une permutation de lieu dans l'espace $3D$. Ils sont donc applicables dans le chiffrement de séquences vidéo ou de séquences d'images multispectrales.

— **Une trajectoire plus complexe et une période plus longue dans un environnement discrétisé**.

En fait, les récurrences chaotiques de grande dimension peuvent être obtenues de différentes manières, telles que la formation des réseaux de récurrences couplées [18, 95, 96], la récurrence

couplée globalement [97], l'extension spatiale [98] et la généralisation multidimensionnelle [99]. Ces méthodes permettent d'obtenir des systèmes chaotiques de haute dimension ayant des comportements dynamiques très complexes, et peuvent résister efficacement aux s'attaques de prédiction des séquences chaotiques.

Bien que les systèmes chaotiques soient adaptés à la conception des cryptosystèmes sécurisés, leur itération dans le temps nécessite plus de calculs. Et en ce qui concerne les méthodes d'extension de la dimension de la récurrence du chat d'Arnold, l'utilisation d'un grand nombre de multiplications matricielles et de relations de dépendance entre les éléments de sa matrice de transformation, produisent des systèmes de nD avec des éléments de matrice hautement corrélés, ce qui peut dégrader la sécurité des applications de cryptographie [100] et rendent difficile leur implémentation matérielle. D'où la construction d'une récurrence chaotique avec un excellent compromis entre complexité et efficacité devient un autre point clé pour la conception de PRNG. Dans cette section nous avons l'intention de mettre en œuvre une approche simple pour concevoir un système dynamique de grande dimension. Pour le faire, nous allons considérer un ensemble de N particules dont la dynamique individuelle est modélisée par des équations linéaires discrètes dans un espace d'état discret et fini. Par la suite, en considérant les interactions entre les particules, nous nous attendons à obtenir un comportement de l'ensemble du système qui est sensible aux conditions initiales et dont la période peut être infiniment augmentée.

2.3.3.1 La synthèse de la PWLCM monobit en dimension 8

Pour modéliser notre système de grande dimension, nous avons considéré la PWLCM présenté précédemment à l'**Eq.2.36** pour modéliser l'évolution du comportement d'une particule. Pour laquelle, les deux variables $x_1(t)$ et $x_2(t)$ représentent respectivement la position et quantité de mouvement de la particule. Ces deux grandeurs dépendent complètement l'une de l'autre comme présentée à l'**Eq.2.36** :

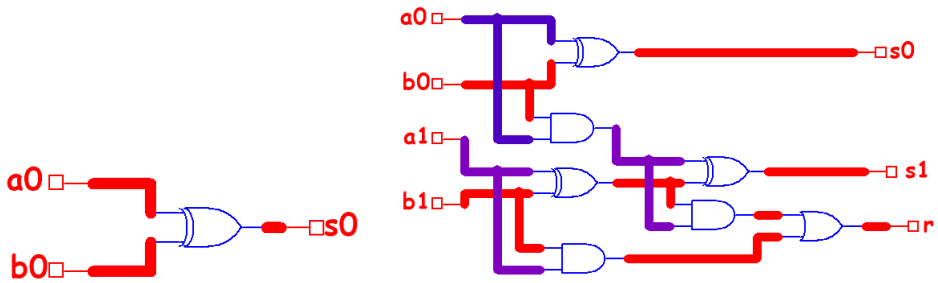
$$\begin{cases} x_1(t+1) = x_1(t) + \alpha_1 x_2(t) + \sum_{i=1}^{M} (a_i + x_2(t)) \bmod c_i \\ x_2(t+1) = x_2(t) + \alpha_2 x_1(t+1) + \sum_{j=1}^{N} (b_j + x_1(t+1)) \bmod d_j \end{cases} \bmod m \qquad (2.36)$$

Pour des applications numériques $m = 2^n$,$n \in \mathbb{N}$ et n représente la précision de calcule ou la longueur binaire des variables d'état $x_1(t)$ et $x_2(t)$. Nous fixons pour la suite de nos travaux $n = 1$. En effet, la précision peut, bien sûr, être choisie plusieurs fois plus grande dans les applications

pratiques. Les raisons pour lesquelles nous avons choisi une précision de calcul monobit sont les suivant :

1. plus la précision est faible, plus le phénomène de dégradation dynamique des séquences chaotiques est évident. La solution de conception des PRNG que nous proposons permet de toujours maintenir de bonnes performances chaotiques même dans des situations de faible précision, ce qui montrera sans aucun doute l'efficacité de notre méthode.
2. plus la précision est élevée, plus la mise en œuvre d'un cryptosystème numérique utilise beaucoup de ressources matérielles et par conséquence prend plus temps, d'espace, de puissance et pour la plupart génère des valeurs avec un débit relativement très faible.

Par exemple, une opération d'addition binaire avec des variables monobites $(a_0 + b_0)$ s'effectue en utilisant une porte logique XOR **Fig.2.12(a)**. Par contre, la même opération d'addition avec des variable de plus d'un bit $(a_1a_0 + b_1b_0)$, s'effectue en utilisant en plus des porte logique XOR, elle utilise aussi les porte logique AND et OR à cause de la retenue anticipée qui est produite à chaque addition monobit **Fig.2.12(b)**.



(a) (b)

FIGURE 2.12 – Niveau de Complexité d'une opération d'addition en fonction de la précision. (a) est une addition monobit , elle contient 1 porte logique XOR, (b) est une addition 2 bits, elle contient 3 portes logiques XOR, 3 portes logiques AND et une porte logiques OR.

Par conséquent, il est nécessaire d'étudier la réalisation d'un système chaotique numérique avec une faible précision.

Sachant que la plupart des données numériques sont codées sur 8 bits, pour réaliser une modification aléatoire de cet octet de données, nous devons utiliser un système dynamique chaotique monobit de dimension 8. Pour cela, nous modélisons un ensemble de L particules numériques qui ne peuvent occuper que $m = 2^n$ états dans l'espace des phases. n est le nombre de bits nécessaires pour représenter les états de la particule individuelle. Pour une implémentation monobit $(n = 1)$, une particule ne peut prendre que deux états 0 et 1. Nous supposons que l'état du système est décrit par la quantité de mouvement x_1 et la position x_2 de la particule.

Pour étendre cette hypothèse au système $8D$, supposons que x_i, $1 \leq i \leq 4$, représentent les impulsions de quatre particules et x_i, $5 \leq i \leq 8$ les positions correspondantes. Alors le comportement de la première particule est décrit par (x_1, x_5), la seconde par (x_2, x_6), la troisième particule par (x_3, x_7) et la quatrième par (x_4, x_8) . L'état du système est décrit par $x = (x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8)^T$, où $(\cdot)^T$ est la transposée de $(\cdot)$. Pour l'ensemble des L particules, le nombre de bits nécessaires pour représenter les états du système est de $2nL$.C'est-à-dire, Pour $n = 1$ et $L = 4$ par exemple, il y'a 256 états différents $(x_1; x_2; x_3; x_4; x_5; x_6; x_7; x_8)$ que le système peut occuper, allant de $(0; 0; 0; 0; 0; 0; 0; 0)$ à $(1; 1; 1; 1; 1; 1; 1; 1)$.

Considérant maintenant un couplage non linéaire entre particules pour décrire les interactions et incluant les chocs entre elles, nous définissons le terme général de couplage suivant :

$$\begin{aligned} x_i(t+1) = x_i(t) + \lambda \left(\alpha_i x_{i+1}(t) + \sum_{j=1}^{M} (\mathbf{a}_{i,j} + x_{i+1}(t)) \bmod \mathbf{c}_{i,j} \right) + \\ \gamma \left(\beta_i x_{i-1}(t+1) + \sum_{k=1}^{M} (\mathbf{a}_{i,k} + x_{i-1}(t+1)) \bmod \mathbf{c}_{i,k} \right) \bmod 2^n \end{aligned} \tag{2.37}$$

$$\lambda = \begin{cases} 1, & \text{si } i = 1 \\ 0, & \text{si non} \end{cases} \tag{2.38}$$

Et

$$\gamma = \begin{cases} 0, & \text{si } i = 1 \\ 1, & \text{si non} \end{cases} \tag{2.39}$$

Considérons le système de quatre particules dont le comportement est décrit par :

$$\begin{cases} x_1(t+1) = x_1(t) + \alpha_1 x_2(t) + \sum_{k=1}^{M_1} (\mathbf{a}_{1,k} + x_2(t)) \bmod \mathbf{c}_{1,k} \\ x_2(t+1) = x_2(t) + \beta_2 x_1(t+1) + \sum_{k=1}^{M_2} (\mathbf{a}_{2,k} + x_1(t+1)) \bmod \mathbf{c}_{2,k} \\ x_3(t+1) = x_3(t) + \beta_3 x_2(t+1) + \sum_{k=1}^{M_3} (\mathbf{a}_{3,k} + x_2(t+1)) \bmod \mathbf{c}_{3,k} \\ x_4(t+1) = x_4(t) + \beta_4 x_3(t+1) + \sum_{k=1}^{M_4} (\mathbf{a}_{4,k} + x_3(t+1)) \bmod \mathbf{c}_{4,k} \\ x_5(t+1) = x_5(t) + \beta_5 x_4(t+1) + \sum_{k=1}^{M_5} (\mathbf{a}_{5,k} + x_4(t+1)) \bmod \mathbf{c}_{5,k} \\ x_6(t+1) = x_6(t) + \beta_6 x_5(t+1) + \sum_{k=1}^{M_6} (\mathbf{a}_{6,k} + x_5(t+1)) \bmod \mathbf{c}_{6,k} \\ x_7(t+1) = x_7(t) + \beta_7 x_6(t+1) + \sum_{k=1}^{M_7} (\mathbf{a}_{7,k} + x_6(t+1)) \bmod \mathbf{c}_{7,k} \\ x_8(t+1) = x_8(t) + \beta_8 x_7(t+1) + \sum_{k=1}^{M_8} (\mathbf{a}_{8,k} + x_7(t+1)) \bmod \mathbf{c}_{8,k} \end{cases} \bmod 2^n \tag{2.40}$$

En décomposant des termes de perturbation de l'**Eq.2.40** comme indiquer par les relations 2.22 et 2.23, Il peut être mis sous forme matricielle suivante :

$$X(t+1) = DX(t) + E(t) \bmod 2^n \tag{2.41}$$

ou :

$$D = \begin{pmatrix} 1 & D_{12} & 0 & 0 & 0 & 0 & 0 & 0 \\ D_{21} & D_{22} & 0 & 0 & 0 & 0 & 0 & 0 \\ D_{31} & D_{32} & 1 & 0 & 0 & 0 & 0 & 0 \\ D_{41} & D_{42} & D_{43} & 1 & 0 & 0 & 0 & 0 \\ D_{51} & D_{52} & D_{53} & D_{54} & 1 & 0 & 0 & 0 \\ D_{61} & D_{62} & D_{63} & D_{64} & D_{65} & 1 & 0 & 0 \\ D_{71} & D_{72} & D_{73} & D_{74} & D_{75} & D_{76} & 1 & 0 \\ D_{81} & D_{82} & D_{83} & D_{84} & D_{85} & D_{86} & D_{87} & 1 \end{pmatrix} \tag{2.42}$$

Avec :

$$\begin{array}{llll} D_{1,2} = 1 + M_1 & D_{4,3} = (1 + M4) & D_{6,4} = D_{6,5}D_{5,4} & D_{8,1} = D_{8,7}D_{7,1} \\ D_{2,1} = 1 + M_2 & D_{5,1} = D_{5,4}D_{4,1} & D_{6,5} = (1 + M_6) & D_{8,2} = D_{8,7}D_{7,2} \\ D_{2,2} = D_{1,2}D_{2,1} + 1 & D_{5,2} = D_{5,4}D_{4,2} & D_{7,1} = D_{7,6}D_{6,1} & D_{8,3} = D_{8,4}D_{4,3} \\ D_{3,1} = (1 + M_3)D_{2,1} & D_{5,3} = D_{5,4}D_{4,3} & D_{7,2} = D_{7,6}D_{6,2} & D_{8,4} = D_{8,5}D_{5,4} \\ D_{3,1} = (1 + M_3)D_{2,1} & D_{5,4} = (1 + M_5) & D_{7,3} = D_{7,4}D_{4,3} & D_{8,5} = D_{8,6}D_{6,5} \\ D_{3,2} = (1 + M_3)D_{2,2} & D_{6,1} = D_{6,5}D_{5,1} & D_{7,4} = D_{7,5}D_{5,4} & D_{8,6} = D_{8,7}D_{7,6} \\ D_{4,1} = D_{4,3}D_{3,1} & D_{6,2} = D_{6,5}D_{5,2} & D_{7,5} = D_{7,6}D_{6,5} & D_{8,7} = (1 + M_8) \\ D_{4,2} = D_{4,3}D_{3,2} & D_{6,3} = D_{6,4}D_{4,3} & D_{7,6} = (1 + M_7) & \end{array}$$

Et

$$E(t) = \Big(E_1(t), E_2(t), E_3(t), E_4(t), E_5(t), E_6(t), E_7(t), E_8(t)\Big)^T$$

L'**Eq.2.40** représente la PWLCM $8D$ avec les états $(x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8)$, il est qualifié de monobit lorsque $n = 1$. De même que la PWLCM $2D$, la PWLCM $8D$ est composé d'une partie conservatrice $DX(t)$ $(\det(D) = 1)$ et d'une partie perturbatrice $E(t)$(**Eq.2.43**). Par la suite nous nous proposons de mettre le système monobit obtenu en dimension 8 en réseau pour concevoir un générateur de nombre pseudo-aléatoire que nous avons nommé : Générateur de nombre pseudo-aléatoire - Piece Wise Linear Arnold Cat Map (GNPA-PWLCM)".

pour la suite du document, nous allons considérer que les paramètres du système $\mathbf{a}(i,k)$, $\mathbf{c}(i,k)$ $\beta_2, \beta_3, \beta_4, \beta_5, \beta_6, \beta_7, \beta_8, \alpha_1$ sont tous positifs et $\beta_2 = \beta_3 = \beta_4 = \beta_5 = \beta_6 = \beta_7 = \beta_8 =$

$\alpha_1 = 1$.

$$E(t) = \begin{pmatrix} \sum_{k=1}^{M_1}(\mathbf{a}_{1,k} - q_{1,k}\mathbf{c}_{1,k} \cdot u(\mathbf{a}_{1,k} + x_2(t) - \mathbf{c}_{1,k})) \\ (1+M_2)E_1(t) + \sum_{k=1}^{M_2}(\mathbf{a}_{2,k} - q_{2,k}\mathbf{c}_{2,k} \cdot u(\mathbf{a}_{2,k} + x_1(t+1) - \mathbf{c}_{2,k})) \\ (1+M_3)E_2(t) + \sum_{k=1}^{M_3}(\mathbf{a}_{3,k} - q_{3,k}\mathbf{c}_{3,k} \cdot u(\mathbf{a}_{3,k} + x_2(t+1) - \mathbf{c}_{3,k})) \\ (1+M_4)E_3(t) + \sum_{k=1}^{M_4}(\mathbf{a}_{4,k} - q_{4,k}\mathbf{c}_{4,k} \cdot u(\mathbf{a}_{4,k} + x_3(t+1) - \mathbf{c}_{4,k})) \\ (1+M_5)E_4(t) + \sum_{k=1}^{M_5}(\mathbf{a}_{5,k} - q_{5,k}\mathbf{c}_{5,k} \cdot u(\mathbf{a}_{5,k} + x_4(t+1) - \mathbf{c}_{5,k})) \\ (1+M_6)E_5(t) + \sum_{k=1}^{M_6}(\mathbf{a}_{6,k} - q_{6,k}\mathbf{c}_{6,k} \cdot u(\mathbf{a}_{6,k} + x_5(t+1) - \mathbf{c}_{6,k})) \\ (1+M_7)E_6(t) + \sum_{k=1}^{M_7}(\mathbf{a}_{7,k} - q_{7,k}\mathbf{c}_{7,k} \cdot u(\mathbf{a}_{7,k} + x_6(t+1) - \mathbf{c}_{7,k})) \\ (1+M_8)E_7(t) + \sum_{k=1}^{M_8}(\mathbf{a}_{8,k} - q_{8,k}\mathbf{c}_{8,k} \cdot u(\mathbf{a}_{8,k} + x_7(t+1) - \mathbf{c}_{8,k})) \end{pmatrix} \quad (2.43)$$

2.3.3.2 L'analyse de la périodicité

Pour différentes combinaisons de valeurs de vecteurs de paramètres de contrôle $a_{i,k}, b_{i,k}, c_{i,k}, d_{i,k}$, il est facile de vérifier que le système est périodique avec une période dans l'espace des phases qui correspond au plus petit commun multiple des périodes produites par les cycles de tous les points initiaux de cet espace des phases. Pour différentes combinaisons de vecteurs de paramètres de contrôle PWLCM 8D et 2D$(1 \leq s \leq 100)$, nous avons évalué différentes périodes des différents systèmes en considérant que le nombre de terme de perturbations maximales pour chaque paramètre d'etat du système vaut 4 ($M_i = 4_{1 \leq i \leq 8}$) et $n = 1$. Les résultats obtenus sont présentés sur la figure 2.13. Ils montrent que la période du système augmente avec sa dimension de sorte que la période maximale du PWLCM 8D est $\Pi_{8D\,PWLCM} = 24$, alors que du PWLCM 2D est $\Pi_{2D\,PWLCM} = 4$ pour une précision arithmétique monobit.

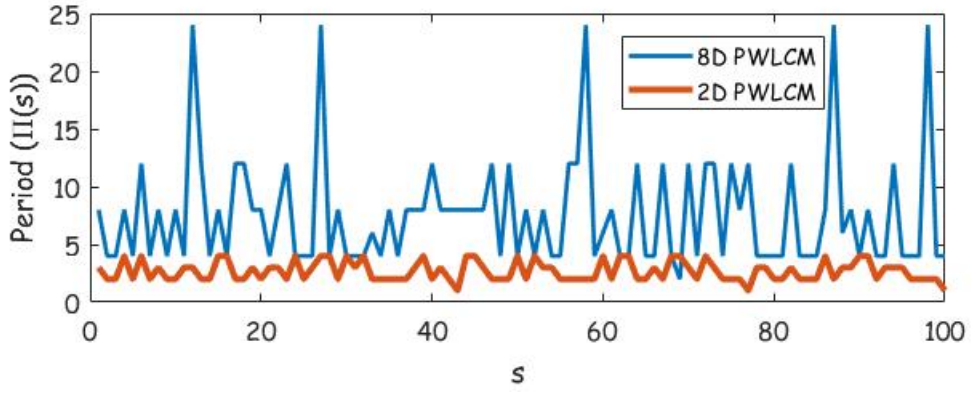


FIGURE 2.13 – Evaluation périodique pour 100 combinaison différents des paramètres de contrôle des systèmes dynamiques 8D et 2D, proposé avec $n = 1$

2.3.3.3 L'analyse des Exposants de lyapunov

Pour déterminer les exposants de Lyapunov de la récurrence obtenue en dimension 8, les valeurs propres de la matrice jacobienne J l'application 2.41 sont considérées. La matrice jacobienne correspondante est donnée :

$$J = \begin{pmatrix} 1 & J_{12} & 0 & 0 & 0 & 0 & 0 & 0 \\ J_{21} & J_{22} & 0 & 0 & 0 & 0 & 0 & 0 \\ J_{31} & J_{32} & 1 & 0 & 0 & 0 & 0 & 0 \\ J_{41} & J_{42} & J_{43} & 1 & 0 & 0 & 0 & 0 \\ J_{51} & J_{52} & J_{53} & J_{54} & 1 & 0 & 0 & 0 \\ J_{61} & J_{62} & J_{63} & J_{64} & J_{65} & 1 & 0 & 0 \\ J_{71} & J_{72} & J_{73} & J_{74} & J_{75} & J_{76} & 1 & 0 \\ J_{81} & J_{82} & J_{83} & J_{84} & J_{85} & J_{86} & J_{87} & 1 \end{pmatrix} \tag{2.44}$$

Avec :

$$\begin{aligned}
J_{1,2}(t) &= D_{1,2} - \sum_{k=1}^{M_1} \mathbf{c}_{1,k}\delta_{1,k}(x_2(t) - \tau_{x_2}^k); & J_{6,3}(t) &= J_{5,3}(t) \cdot J_{6,4}; \\
J_{2,1}(t) &= D_{2,1} - \sum_{k=1}^{M_2} \mathbf{c}_{2,k}\delta_{2,k}(x_1(t+1) - \tau_{x_1}^k); & J_{6,4}(t) &= J_{5,4}(t) \cdot J_{6,5}; \\
J_{7,6}(t) &= \left(D_{7,6} - \sum_{k=1}^{M_7} c_{7,k}\delta_{7,k}(x_6(t+1) - \tau_{x_7}^k)\right); & J_{2,2}(t) &= J_{1,2}(t) \cdot J_{2,1}(t) + 1; \\
J_{3,1}(t) &= J_{2,1}(t) \cdot \left((1+M_3) - \sum_{k=1}^{M_3} c_{3,k}\delta_{3,k}(x_2(t+1) - \tau_{x_3}^k)\right); & J_{7,1}(t) &= J_{6,1}(t) \cdot J_{7,6}(t); \\
J_{3,2}(t) &= J_{2,2}(t) \cdot \left((1+M_3) - \sum_{k=1}^{M_3} \mathbf{c}_{3,k}\delta_{3,k}(x_2(t+1) - \tau_{x_3}^k)\right); & J_{7,2}(t) &= J_{6,2}(t) \cdot J_{7,6}(t); \\
J_{4,3}(t) &= \left(D_{4,3} - \sum_{k=1}^{M_4} \mathbf{c}_{4,k}\delta_{4,k}(x_3(t+1) - \tau_{x_4}^k)\right); & J_{7,3}(t) &= J_{6,3}(t) \cdot J_{7,6}(t); \\
J_{4,1}(t) &= J_{3,1}(t) \cdot J_{4,3}(t); & J_{7,4}(t) &= J_{6,4}(t) \cdot J_{7,6}(t); \\
J_{4,2}(t) &= J_{3,2}(t) \cdot J_{4,3}(t); & J_{7,5}(t) &= J_{6,5}(t) \cdot J_{7,6}(t); \\
J_{5,4}(t) &= \left(D_{5,4} - \sum_{k=1}^{M_5} \mathbf{c}_{5,k}\delta_{5,k}(x_4(t+1) - \tau_{x_5}^k)\right); & J_{6,2}(t) &= J_{5,2}(t) \cdot J_{6,5}(t); \\
J_{5,1}(t) &= J_{4,1}(t) \cdot J_{5,4}(t); & J_{8,1}(t) &= J_{7,1}(t) \cdot J_{8,7}(t); \\
J_{5,2}(t) &= J_{4,2}(t) \cdot J_{5,4}(t); & J_{8,2}(t) &= J_{7,2}(t) \cdot J_{8,7}(t); \\
J_{5,3}(t) &= J_{4,3}(t) \cdot J_{5,4}(t); & J_{8,3}(t) &= J_{7,3}(t) \cdot J_{8,7}(t); \\
J_{6,5}(t) &= \left(D_{6,5} - \sum_{k=1}^{M_6} \mathbf{c}_{6,k}\delta_{6,k}(x_5(t+1) - \tau_{x_6}^k)\right); & J_{8,4}(t) &= J_{7,4}(t) \cdot J_{8,7}(t);
\end{aligned}$$

$$J_{6,1}(t) = J_{5,1}(t) \cdot J_{6,5}(t); \qquad J_{8,5}(t) = J_{7,5}(t) \cdot J_{8,7}(t);$$
$$J_{8,7}(t) = \left(D_{8,7} - \sum_{k=1}^{M_8} \mathbf{c}_{8,k} \delta_{8,k} (x_7(t+1) - \tau_{x_8}^k) \right); \quad J_{8,6}(t) = J_{7,6}(t) \cdot J_{8,7}(t).$$

avec : $\tau_{i,k} = c_{i,k} - a_{i,k}$;

et

$$\delta_{i,k}(t) = \begin{cases} 1, & \text{if } t = 0 \\ 0, & \text{else if} \end{cases} \tag{2.45}$$

Sachant que le $\det(J) = 1$, nous pouvons tirer les conclusions suivantes la PWLCM en dimension 8 que nous avons pu obtenir.

— C'est un système dynamique **conservatif** ;

— il est **inversible** ;

— La somme de ses Exposants de Lyapunov ($\lambda_i(t)_{1 \leq i \leq 8}$) est nulle impliquant que le plus grand esposant de lyapunov est positif dont, c'est un système dynamique **chaotique** pour $n = 0$.

Toutes ces caractéristiques sont indépendamment du choix des paramètres $\mathbf{a}_{i,k}$ et $\mathbf{c}_{i,k}$. Selon le choix de ces paramètres, il peut être difficile de déterminer les états stationnaires du système, quand ils existent.

En calculant les valeurs propres de J, nous avons trouvé $\lambda_3 = \lambda_4 = \lambda_5 = \lambda_6 = \lambda_7 = \lambda_8 = 1$ et les deux autres valeurs propres ont pour expression :

$$\Lambda_1(t) = 1 + \tfrac{1}{2} \left(J_{1,2}(t) \cdot J_{2,1}(t) \right) \left(1 + \sqrt{1 + \tfrac{4}{J_{1,2}(t) \cdot J_{2,1}(t)}} \right) \tag{2.46}$$

$$\Lambda_2(t) = 1 + \tfrac{1}{2} \left(J_{1,2}(t) \cdot J_{2,1}(t) \right) \left(1 - \sqrt{1 + \tfrac{4}{J_{1,2}(t) \cdot J_{2,1}(t)}} \right) \tag{2.47}$$

Les exposants de Lyapunov correspondant aux valeurs propres $\lambda_{3,4,5,6,7,8}(t)$ sont nuls. La valeur propre $\lambda_1(t) \geq 1$ (correspondant à un exposant de Lyapunov positif) et $0 > \lambda_2(t) > 1$(correspondant à un exposant de Lyapunov négatif). L'état stationnaire du système dépend du paramètre perturbateur et reste formellement difficile à déterminer . La **Fig.2.14** présente le comportement de l'exposant de Lyapunov pour un paramètre arbitraire et diverses conditions initiales $(x_1(0), x_2(0), x_3(0), x_4(0), x_5(0), x_6(0), x_7(0), x_8(0))$. Les exposants de Lyapunov ont été évalués après 150 itérations.

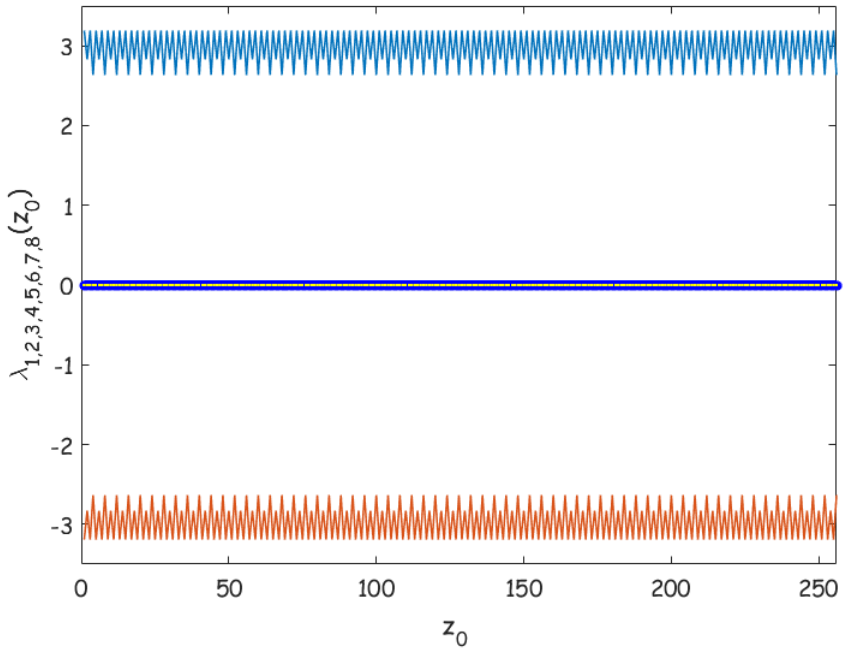


FIGURE 2.14 – Représentation graphique des 8 Exposants de Lyapunov de la PWLCM 8D ($\Lambda_{1,2,3,4,5,6,7,8}(z0)$) en fonction des conditions initiales $z0 = x_1(0) + x_2(0) \cdot 2 + x_3(0) \cdot 2^2 + x_4(0) \cdot 2^3 + x_5(0) \cdot 2^4 + x_6(0) \cdot 2^5 + x_7(0) \cdot 2^6 + x_8(0) \cdot 2^7$, pour $n = 1$ et un vecteur de paramètres de contrôles aléatoires $\mathbf{a}_{i,k}$ et $\mathbf{c}_{i,k}$

.

2.4 Conclusion

Le présent chapitre avait pour objectif, de présenter les différents outils et la méthodologie que nous avons adopté pour développer notre générateur de nombre pseudo-aléatoire. Tout au long de notre développement, nous avons d'abord fait la présentation de quelques méthodes analytiques utilisées dans la littérature pour qualifier un système chaotique ou une séquence aléatoire. Parmi ces méthodes, il y avait entre autres les Exposants de Lyapunov, l'entropie, les tests de corrélation et les tests statistiques de NIST. Ensuite partant, de la récurrence du Chat d'Arnold nous avons proposé un nouveau système dynamique nommé la récurrence du Chat d'Arnold linéaire par morceau, "piece-wise linear cat map (PWLCM)". L'analyse de ce système a montré qu'il présente une dynamique beaucoup plus complexe que la récurrence d'Arnold classique. Pour réduire la complexité du système obtenue en dimension 2 sur 4 bits, nous avons présenté une généralisation de la PWLCM monobit en dimension 8.

CHAPITRE 3

RÉSULTATS ET DISCUSSION

3.1 Introduction

Divers domaines de la recherche et de la télécommunication nécessitent la génération des nombres pseudo-aléatoires. Toute proposition d'un générateur de nombres pseudo-aléatoires chaotiques évolue suivant une loi déterministe qui repose sur le choix du type de systèmes chaotiques, des conditions initiales et des paramètres de contrôles. Nous avons présenté dans le chapitre précédent les différentes récurrences chaotiques sur lesquelles repose l'algorithme du générateur de nombres pseudo-aléatoires que nous proposons dans cette thèse. Dans ce chapitre, nous allons réaliser la mise en œuvre sur FPGA des différents systèmes dynamiques que nous avons proposé. À partir des résultats de l'implémentation, nous ferons le choix du système idéal pour la conception de notre GNPA. Par la suite, une description de l'algorithme de mise en réseau du système dynamique choisi est effectuée et implémentée sous forme de GNPA. Cette implémentation sera réaliser sur la plateforme de prototypage FPGA Zynq xc7z020clg400-1 dont dispose le laboratoire d'électronique de l'université de Yaoundé 1.

3.2 L'implémentation des différents PWLCM sur FPGA

l'implémentation électronique est la meilleure solution pour réaliser un cryptosystème puissant. Elle offre plus de performances et sécurité grâce à l'exécution d'un algorithme. En conséquence, deux choix sont proposés : un circuit intégré à l'application spécifique, "Application-specific integrated circuit,(ASIC)" et un réseau de portes logiques programmables (FPGA). Le premier choix est le plus cher, le second est une solution prometteuse. Les FPGA permettent au concepteur de créer une implémentation de circuit personnalisée d'un algorithme à l'aide d'un composant standard composé d'éléments logiques programmables de base. Cette technique offre des avantages significatifs en matière de coût par rapport à un effort de développement ASIC et offre le même niveau de performances dans la plupart des cas. Un autre avantage du FPGA par rapport à l'ASIC est sa capacité à être reconfiguré dynamiquement.

Les FPGAs (Field Programmable Gate Arrays en anglais) ou réseaux de portes logiques programmables sont une famille de composants programmables depuis un programme appelé "bitstream". Ce dernier n'est pas destiné à être exécuté par un microprocesseur, mais plutôt à configurer des portes logiques et les relier entre elles selon une logique d'interconnexion, pour répondre à un objectif bien déterminé. De part, la possibilité d'être reconfigurés dans leur totalité et leur haut degré de parallélismes et de flexibilités, les FPGAs sont devenus les plus populaires en matière d'implémentation et de prototypage des circuits numériques, pour un bon nombre d'applications, y compris les applications cryptographiques [82]. Cependant, pour réussir à implémenter un système sur FPGA de manière efficace, il est indispensable de bien connaître sa structure interne et ses limites du point de vue performance.

La connaissance de la structure interne du système que nous souhaitons implémenter passe d'abord par l'étape de synthèse. Dans cette partie, nous allons procéder à la synthèse des différents systèmes que nous avons proposé au chapitre précédent grâce à la plateforme de conception Vivado 2018.3. Les modules qui constituent ces systèmes tels que les additionneurs, et comparateurs, multiplexeurs et registres sont obtenus ou créés à l'aide du générateur IP-CORE développé pour la plate-forme. Les fichiers sources des différentes équations (**Eq.2.17**,**Eq.2.40**) sont construits à l'aide des modules requis et du langage matériel Verilog HDL utilisant une arithmétique de précision n-bits.

La synthèse des différentes PWLCM que nous avons proposée sur FPGA est réalisée à partir des modules suivants :

1. Le module de la **Fig.3.1** représente l'architecture numérique du terme additif que nous avons introduit dans la récurrence du chat d'Arnold pour modifier sa dynamique. Les variables d'état de chaque système implémenté possèdent un nombre k de ces modules de "perturbation". Il est constitué d'un additionneur n-bit, d'un multiplexeur et d'un opérateur modulo.

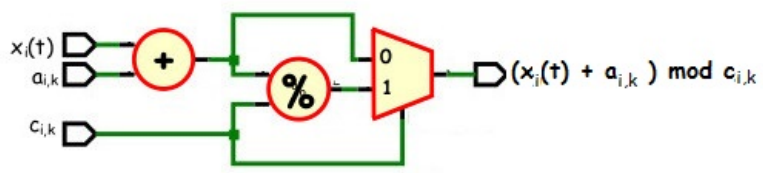


FIGURE 3.1 – Architecture des termes d'addition modulaire (P_k).

2. Le module de la **Fig.3.2** illustre les différentes opérations à réaliser pour obtenir la nouvelle valeur d'une variable d'état $x_i(t+1)$ à partir de celle produite par chaque terme de perturbation (P_K), de l'état précédent $x_i(t)$ et de la nouvelle valeur de la variable d'état qui la précède $x_{i-1}(t+1)$.

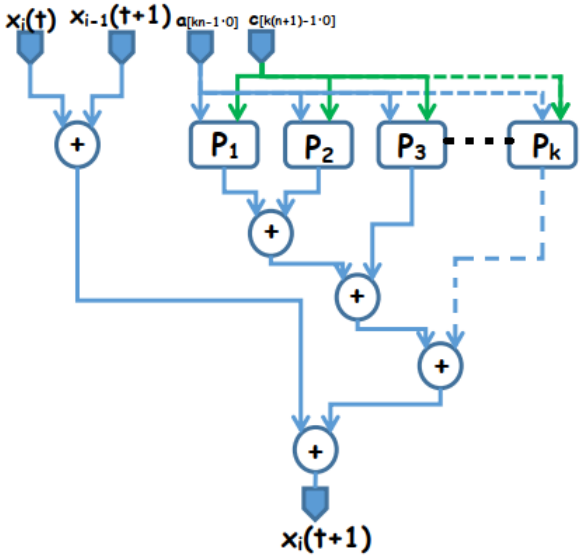


FIGURE 3.2 – Architecture générale du module d'évaluation de chaque variable d'état x_i.

3. L'association d'un nombre d d'architectures présentés à la **Fig.3.2** permet d'obtenir le schéma de la PWLCM de dimension d comme présenté à la **Fig.3.3**. Par exemple si $d = 4$ cela permettra d'obtenir la récurrence du chat D'Arnold linéaire par morceaux de dimension 4 sur n-bit.

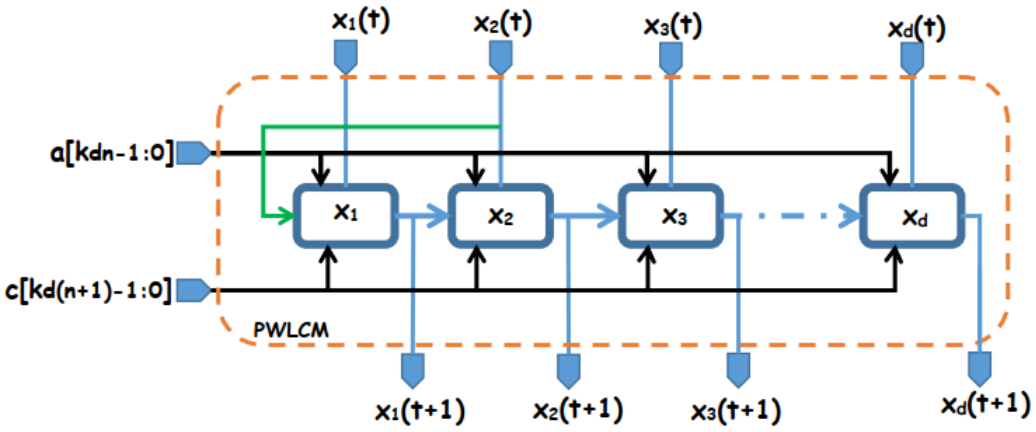


FIGURE 3.3 – Architecture générale de la PWLCM.

La **Fig.3.4** illustre l'architecture de mise en œuvre des différentes PWLCM implémentées sur FPGA. L'implémentation est réalisée de telle sorte qu'à chaque itération, ils produisent le même nombre de bits soit dn bits (d représente la dimension du système et n la précision de calcul).

La taille binaire des paramètres de contrôle est fonction de d, n et du nombre de termes additifs utilisés k lors de la conception d'une PWLCM spécifique, tel que : la longueur de bit du paramètre **a** vaut $k \cdot d \cdot n$ et celle de **c** correspond à $k \cdot d \cdot (n+1)$.

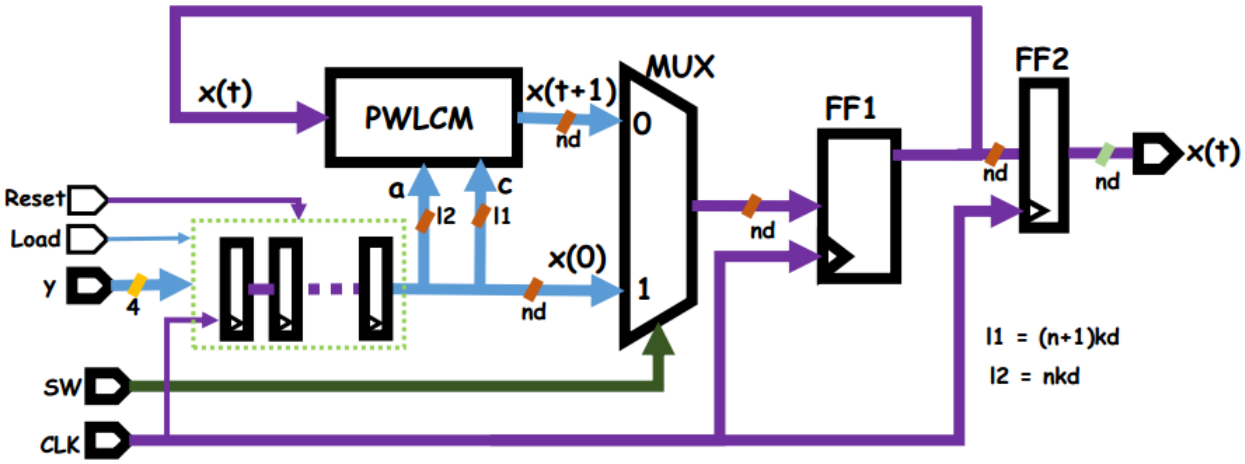


FIGURE 3.4 – Implémentation FPGA généralisée de PWLCM n-bit. La commande "LOAD" permet de fixer $x(0)$ comme condition initiale, a et c comme paramètres de contrôles dans le registre à $kd(2n+1)+nd$-bits ; "RESET" permet d'effacer le registre de $kd(2n+1)+nd$-bits ; et la commande "SW" permet de charger les conditions initiales dans les registres à nd-bits **FF1**, donc de démarrer le système.

Cette étape, nous a permis d'obtenir la quantité de ressources matérielles utilisées, ainsi que le débit des différentes architectures de PWLCM implémenté qui sont la PWLCM $2D$, $4D$ et $8D$. Les résultats obtenus sont inscrits dans le **Tab.3.1**. De ce tableau, nous constatons que la mise en œuvre de ces systèmes ne nécessite aucun module DSP, mais exclusivement des modules de base tels que tables de correspondance (LUT) et bascule (FF). Comparé à la PWLCM 2D, et la PWLCM 4D, la PWLCM monobit 8D se présente comme le système dynamique la mieux adapté pour la conception d'un GNPA sécurisé et de faibles complexités pour les systèmes IoTs. Car ce dernier ne consomme que très peu de ressources (4 LUT et 8 FF) que les deux autres et fonctionne à un débit élevé de $5,05$ Gbps pour une implémentation générant la même taille de données. En plus son espace de clé est le plus élevé soit 64 bits ce qui nécessite l'utilisation de 4 PWLCM pour obtenir un générateur avec un espace de clé de 256 qui représente la taille minimale de la clé pour garantir une sécurité. Par contre pour avoir une taille de clé minimale de 256 avec les deux autres systèmes nous devrions utiliser plus de 6 système dynamique ce qui entrainera une énorme consommation des ressources et par conséquence de l'énergie. C'est pour ces raisons que nous l'avons adopté pour implémenter notre générateur de nombres aléatoires.

TABLE 3.1 – Comparaison des ressources utilisées lors de la mise en œuvre de PWLCM $2D$ sur $4-$bits, PWLCM $4D$ sur $2-$bits et de PWLCM $8D$ sur $1-$bit, chaque système produit une sortie 8 bits.

	PWLCM		
	2D	4D	8D
n	4	2	1
k	4	4	4
d	2	4	8
c(bit)	40	48	**64**
a(bit)	32	32	**32**
IO	16	16	16
LUT	11	8	**4**
FF	12	12	**8**
Fréquence maximale(MHz)	363,90	410,5	**631,31**
débit (Gbps)	2,911	3,28	**5,05**

3.3 La mise en réseau de PWLCM monobit en dimension 8

Pour résoudre les problèmes de dégradation dynamique et faible taille des clé généralement rencontrés lors de la conception des GNPA, nous avons adopté une architecture de réseau dynamique aléatoire pour la structure interne du GNPA que nous proposons. Elle est composée de plusieurs PWLCM monobit $8D$ utilisées comme source d'entropie. Chaque PWLCM est définie comme une application booléenne $f : \mathbb{B}^8 \to \mathbb{B}^8$. Une technique de post-traitement basé sur l'opération logique "OU-EXCLUSIF" est également utilisée pour augmenter l'incertitude et le caractère aléatoire de chaque sortie de la PWLCM monobit. Elle est aussi utilisée comme lien d'interaction entre tous les PWLCM du réseau, favorisant ainsi un réseau de PWLCM robuste.

Initialement, le générateur est conçu de sorte que, la graine $X(0)$, l'état interne $X(t)$ et la sortie $X(t+1)$ sont tous exprimés avec le même nombre N de bits comme illustré dans **Fig.3.5** sans perte de données. Pour la suite, nous considérons que $N = 64$. Pour produire une nouvelle sortie $X(t+1)$ à partir d'une entrée donnée $X(t)$ au travers du réseau, Il faut premièrement que $X(t)$ soit tronqué en deux sous-blocs X_A^t et X_B^t. Chaque sous-bloc est d'abord divisé en blocs p de longueur binaire égale à 8. Pour un état X de taille 64 bit, $p = 4$ et on a donc $X_A^t = (x_a^t, x_b^t, x_c^t, x_d^t)$ où x_l^t correspond au sous bloc de X_A^t pour $l \in (a, b, c, d)$ et $X_B^t = (x_e^t, x_f^t, x_g^t, x_h^t)$ où x_m^t correspond au sous bloc de X_B^t avec $m \in (e, f, g, h)$.

Chaque bloc x_l^t est modifié séparément tel que **Fig.3.5(1)** :

$$x_a^{t+1} = PWLCM(x_a^t \oplus x^0)$$
$$x_b^{t+1} = PWLCM(x_b^t \oplus x_a^{t+1})$$
$$x_c^{t+1} = PWLCM(x_c^t \oplus x_b^{t+1})$$
$$x_d^{t+1} = PWLCM(x_d^t \oplus x_c^{t+1})$$

où tous les x_l^{t+1} sont ensuite concaténés, produisant le nouvel état interne X_A^{t+1} tel que $X_A^{t+1} = (x_a^{t+1}, x_b^{t+1}, x_c^{t+1}, x_d^{t+1})$. Les valeurs des paramètres de contrôles $\{\mathbf{a}_{i,k}\}$ et $\{\mathbf{c}_{i,k}\}$ de chaque PWLCM 8D sont obtenues respectivement en utilisant X_B^t et la clé du GNPA de taille $d{\cdot}(n{+}1){\cdot}k{\cdot}e$. Les variables d et k, représentent respectivement la dimension et le nombre de termes d'addition modulaire utilisé. La variable e est le nombre d'octets du sous-bloc X_B^t de taille $N/2$. Chaque $\mathbf{a}_{i,k}$ a une longueur de n bits et chaque $\mathbf{c}_{i,k}$ une taille de bit de $n+1$ bits. Le signe $\oplus$ représente l'opérateur logique "OU-EXCLUSIF"

Chaque bloc x_m^t est modifié séparément tel que :

$$x_e^{t+1} = PWLCM(x_e^t \oplus x^0)$$
$$x_f^{t+1} = PWLCM(x_f^t \oplus x_e^{t+1})$$
$$x_g^{t+1} = PWLCM(x_g^t \oplus x_f^{t+1})$$
$$x_h^{t+1} = PWLCM(x_h^t \oplus x_g^{t+1})$$

où tous les x_m^{t+1} sont ensuite concaténés, produisant le nouvel état interne X_B^{t+1} tel que $X_B^{t+1} = (x_e^{t+1}, x_f^{t+1}, x_g^{t+1}, x_h^{t+1})$. Les valeurs des paramètres de contrôles $\{\mathbf{a}_{i,k}\}$ et $\{\mathbf{c}_{i,k}\}$ de chaque PWLCM 8D sont obtenues respectivement en utilisant X_A^{t+1} et $iKey$. Le vecteur de clé $iKey$ est obtenu en inversant l'ordre binaire de la clé du GNPA Key. Par conséquent, le bit $j - me$ de Key devient $(d \cdot (n+1) \cdot k \cdot e - 1) - j$ ème bit dans $iKey$, $0 \leq j < d \cdot (n+1) \cdot k \cdot e$.

Enfin, les X_A^{t+1} et X_B^{t+1} obtenus sont concaténés sur N bits pour produire la nouvelle sortie $X(t+1)$ (**Fig.3.5(2)**).

3.4 L'implémentation FPGA du GNPA-PWLCM

Bien que les systèmes dynamiques proposés dans le chapitre précédent possèdes des propriétés aléatoires, mais à cause du problème de dégradation dynamique causer par une implémentation monobit, ils ne peuvent directement être exploités en tant que générateurs de nombres pseudo-aléatoires. Puisque, les séquences qu'ils produisent ne passe pas les tests statistiques de NIST avec

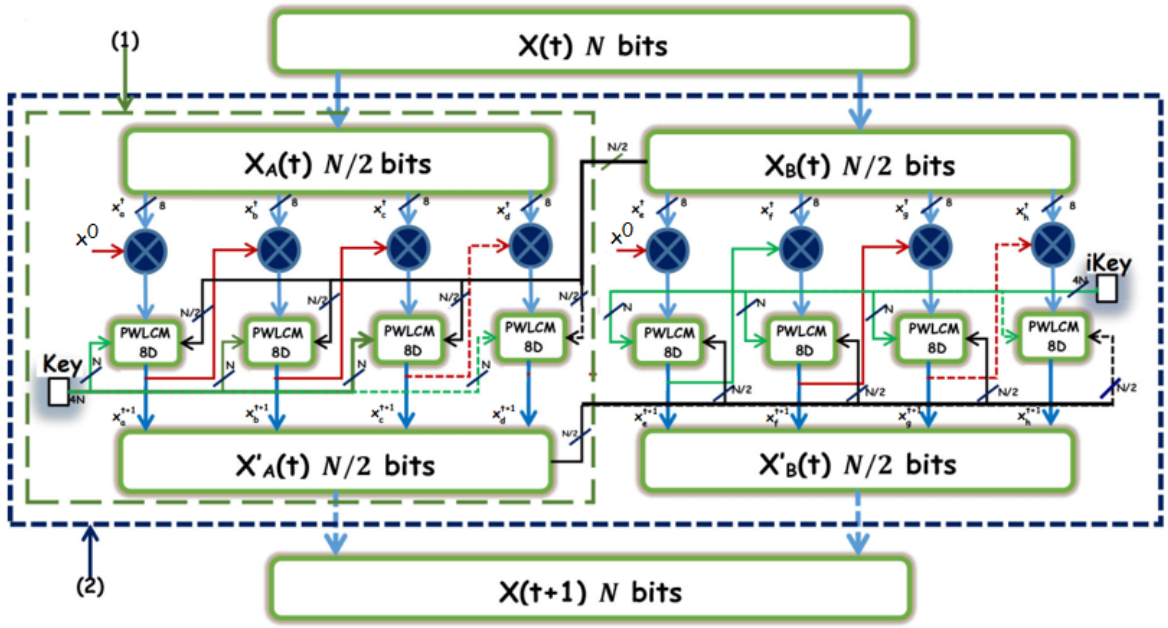


FIGURE 3.5 – Le réseau de PWLCM monobit en dimension 8 : La structure interne du PRNG-PWLCM pour $N = 64$

succès. Pour résoudre ce problème, la solution que nous proposons dans cette thèse consiste à combiner sous forme d'un réseau dynamique, plusieurs PWLCM monobit obtenu en dimension 8 pour créer un générateur de nombres pseudo-aléatoires plus robuste et ultra-rapide. L'algorithme de ce générateur a été décrit dans la section précédent, cette partie étant réservée à la description de son l'implémentation FPGA.

Les étapes de synthèse et de mise en œuvre sont les prochaines étapes après l'élaboration de l'algorithme de fonctionnement. Nous avons réalisé notre générateur sur la plateforme d'évaluation FPGA Zynq xc7z020clg400-1. Cette plateforme possède une fréquence de 125 MHz. Les différents schémas d'architectures du générateur de nombres pseudo-aléatoires proposés, issus de la phase de synthèse, sont visibles sur la **Fig.3.6**. Notons que toutes les opérations internes effectuées au sein du module GNPA-PWLCM sont soumises aux contrôles des différents signaux d'entrée/sortie décrits au tableau 3.2.

Le **Tab.3.3** présente, l'estimation des ressources FPGA utilisées par l'architecture du PRNG que nous avons proposé. A cause des différentes contraintes imposées par la plateforme d'implémentation. La synthèse a été réalisée en considérant $R(t)$ comme étant la variable aléatoire externe produite à chaque front montant d'horloge. Elle possède une taille binaire w telle que $w \leq N$ et $N = 64$. La mise en œuvre de cette architecture s'est faite avec succès sans violation de synchronisation à l'aide d'une période minimale d'horloge de $10\ ns$. Pour la réalisation de nos

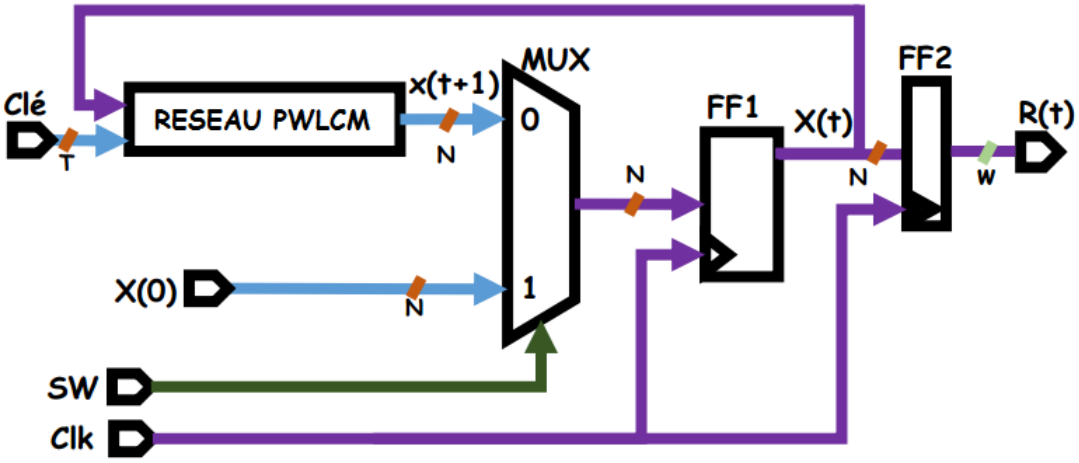


FIGURE 3.6 – La structure Général du GNPA-PWLCM N bits sur le FPGA

TABLE 3.2 – Description des signaux intervenant au module GNPA-PWLCM.

Nom	Direction	Description
Clk	Entrée	Signal d'horloge du système
SW	Entrée	Mise en marche des itérations
$R[w-1:0]$	Sortie	Flux des bits pseudo-aléatoires, dont N bits interne et w le nombre de bit externe, produits à chaque cycle d'horloge.tels que $w \leq N$
$X(0)[N-1:0]$	Entrée	Vecteur d'initialisation

tests et simulations numérique, nous avons considéré que : clé $= [K_1, K_2, K_3, K_4, K_5, K_6, K_7, K_8]$ et $X(0) = [K_9, K_{10}]$. Chaque K_i représente une variable binaire codée sur 32-bits avec $K_1 = 025555210, K_2 = 000065211, K_3 = 020365212, K_4 = 020365213, K_5 = 020365214, K_6 = 020365215, K_7 = 020365216, K_8 = 120365214, K_9 = 336254778, K_{10} = 985852146$

TABLE 3.3 – Les Caractéristiques du GNPA-PWLCM sur FPGA. Avec $N = 64$-bits interne, $w = 32$ et $w = 64$ bits en sortie.

Ressources	GNPA-PWLCM		Disponible
Utilisées	32	64	
Nombre de LUT	88	90	53200
Nombre de registre (FF)	64	65	106400
IO	34	66	200
LUTRAM	0	0	17400
BUFG	1	1	32
DSP	0	0	220
Fréquence maximale (MHz)	195.503	177,809	...
Débit (Gbps)	6,256	11,37	...

3.5 L'analyse de l'aléa du générateur de nombres pseudo-aléatoires proposé

L'utilisation des GNPAs la plus importante se situe au niveau de la conception des crypto-systèmes numériques intégrés, qui sont utilisés dans des schémas de cryptage de données, tel que le cryptage d'image haute définition dans l'Internet des objets et le cryptage vidéo numérique dans l'armée. En ce sens, le GNPA-PWLCM proposé et implémenté sur FPGA a été soumis à une analyse de sécurité du point de vue de la cryptographie pour montrer sa capacité à garantir la sécurité dans de telles applications. L'analyse de sécurités comprend les tests statistiques de NIST, l'espace de clés, la sensibilité des clés, l'entropie, histogramme, portrait de phase et la corrélation.

3.5.1 Le test statistique de NIST

Dans la mise en œuvre matérielle, la dégradation du chaos provoquée par la précision des calculs affecte grandement le caractère aléatoire des GNPAs. Par exemple, le phénomène de courte période peut apparaître dans une simulation chaotique, ce qui entraîne une périodicité des séquences aléatoires et conduit finalement à l'échec du test de séquence aléatoire. À l'heure actuelle, la suite de tests NIST 800.22 est la norme de test de caractère aléatoire la plus couramment utilisée, qui utilise 15 méthodes de test pour évaluer un grand nombre de séquences aléatoires. Par conséquent, pour déterminer le caractère aléatoire du GNPA-PWLCM monobit, nous avons testé ses séquences aléatoires émises à l'aide de la suite de tests NIST 800.22. Ce test recommande que la longueur de la séquence à tester soit au minimum de $100 \cdot 10^6$ bits. Pour atteindre cette taille nous avons réalisé 1562500 et 3125000 itérations respectivement pour le GNPA-PWLCM 64 et 32 bits. Les résultats des tests des séquences aléatoires générées sont présentés dans le tableau 3.4. Ils montrent que les 15 tests ont réussi et que les séquences aléatoires ont un bon caractère aléatoire.

3.5.2 L'espace des clés de chiffrement

La taille de l'espace des clés détermine la sécurité d'un système de chiffrement. Un grand espace de clés peut améliorer la force de cryptage, protéger ainsi efficacement la confidentialité des informations et résister à l'analyse cryptographique. Les petits espaces de clés sont vulnérables aux attaques exhaustives, qui, en tant qu'attaque standard, peuvent être utilisées pour n'importe quel mot de passe. Ce type d'attaque implique généralement d'épuiser toutes les clés possibles en fonction de certaines politiques et règles jusqu'à ce que la bonne soit trouver. On pense généra-

TABLE 3.4 – Résultat des tests de NIST des séquences produites par le GNPA-PWLCM 32 et 64 bits.

	$\mathbb{Z}_{64}$		$\mathbb{Z}_{32}$	
Statistical test	P-value	Proportion %	P-value	Proportion
Frequency	0.911413	100/100	0.075719	100/100
Block-frequency	0.574903	98/100	0.334538	99/100
Cumulative-sums	0,935716	99/100	0.249284	100/100
Runs	0.224821	100/100	0.554420	96/100
Longest-runs	0.025193	99/100	0.834308	100/100
Rank	0.699313	99/100	0.048716	99/100
FFT	0.289667	99/100	0.867692	100/100
Non-overlapping-templates	0.911413	100/100	0.739918	99/100
Overlapping-templates	0.554420	100/100	0.719747	100/100
Universal	0.006196	100/100	0.401199	99/100
Approximate entropy	0.924076	100/100	0.304126	100/100
Random-excursions	0.534146	58/58	0.875539	65/65
Random-excursions variant	0.911413	57/58	0.756476	65/65
Serial	0.554420	98/100	0.816537	99/100
Linear-complexity	0.554420	100/100	0.574903	99/100

lement que la taille de l'espace clé doit être supérieure à 2^{256} pour garantir qu'il résistera à une attaque exhaustive [108].

Dans cet article, le GNPA proposé utilise 8 PWLCM monobit de dimension 8 . La PWLCM 8D a de nombreux paramètres, ce qui implique un espace clé relativement grand. Sur la base de l'arithmétique monobit et en utilisant seulement 4 termes de perturbation pour chaque variable d'état, la clé est composée du vecteur initial de 64 bits, de la taille des paramètres de contrçôles d'une taille 256 bits répartie en $\{c_1, c_2, c_3, c_4\}$ de 64-bits chacun et de P_0 avec une taille binaire de 8-bits, ce qui fait un total de 328-bit. En d'autres termes, l'espace des clés obtenu par cette méthode est 2^{328}. Le générateur proposé peut donc bien résister à aux attaques exhaustive car il possède une taille de clé supérieure à 2^{256}.

3.5.3 Le portrait de phase

Le caractère aléatoire d'une séquence de nombres peut être analysé à partir du tracé de l'espace de phase. Les tracés de l'espace de phase des séquences générées par notre PRNG sont illustrés à la **Fig.3.7**. Nous définissons clé $= [K_1, K_2, K_3, K_4, K_5, K_6, K_7, K_8]$ et $X(0) = [K_9, K_{10}]$. Sachant que les 64 bits, produits à chaque itération par le générateur proposé n'est qu'une concaténation des 8-bits générés par chaque PWLCM monobit de dimension 8. Nous avons représenté les portraits des

phases en utilisant quelques couples des séquences de 8-bits générées par les différents PWLCM du générateur (x_a, x_b), (x_c, x_d), (x_e, x_f),(x_g, x_h). De cette figure, on observe que les données sont dispersées dans le plan, donc sont aléatoires.

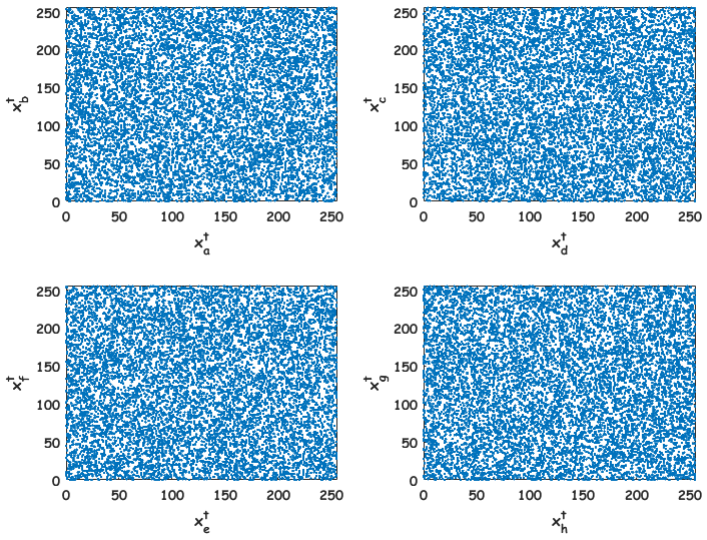


FIGURE 3.7 – Les portraits de phase entre (x_a, x_b) ;(x_c, x_d) ;(x_e, x_f) ;(x_g, x_h) ;

3.5.4 La sensibilité à la clé de chiffrement

Une caractéristique importante du système dynamique chaotique est qu'il est très sensible aux conditions et paramètres initiaux. Par conséquent, GNPA-PWLCM proposé dans cette thèse doit également avoir une bonne sensibilité de clé, car c'est une caractéristique appreciée en cryptographie. Pour ce test, nous allons effectué 10 itérations pour générer 80 valeurs numérique codée sur 8 bits. A chaque itération de notre générateur produit le mot binaire $X(t+1)$ de 64, formé en réalisant la concaténation de 8 mots binaires de 8 bits chacun.

— Pour la sensibilité à la clé, nous avons fixé le vecteur d'initialisation à $X(0) = [K_9, K_{10}]$. Les deux clés légèrement différentes que nous avons utilisé sont :

⋆ clé $= [K_1, K_2, K_3, K_4, K_5, K_6, K_7, K_8]$

⋆ clé $= [K_1, K_2, K_3, K_4, K_5, K_6, K_7, K_8^{'}]$ avec : $K_8 = 120365215$

le résultat obtenu est présenté à La **Fig.3.8(a)** où l'onde rouge est celle obtenue par avec clé $= [K_1, K_2, K_3, K_4, K_5, K_6, K_7, K_8]$ et l'onde bleue est celle obtenue en utilisant $[K_1, K_2, K_3, K_4, K_5, K_6, K_7, K_8^{'}]$ comme clé.

— Pour la sensibilité au vecteur d'initialisation, nous avons généré deux sequences produites par eux vecteurs initiales$(X(0)^{'}, X(0)^{''})$ légèrement différents et de clé identique clé $= [K_1, K_2, K_3, K_4, K_5, K_6, K_7, K_8^{]}$.

⋆ $X(0)^{'} = [K_9, K_{10}]$

⋆ $X(0)^{"} = [K_9, K_{10}^{'}]$ avec : $K_{10} = 985852147$

le résultat obtenu est présenté sur la **Fig.3.8(b)** où l'onde rouge est celle obtenue par le $X(0)^{'}$ et la vague bleue est celle obtenue par le $X(0)^{''}$

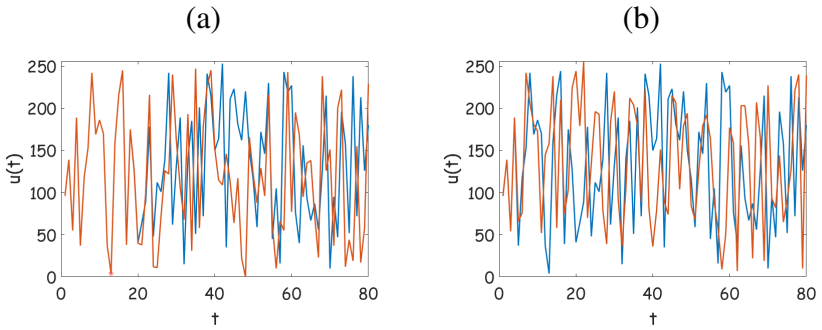


FIGURE 3.8 – Les formes d'onde traduisant la sensibilité du générateur : (a) aux conditions initiales , (b) à la clé du générateur .

La figure 3.8 montre que la moindre modification sur la clé de chiffrement ou sur le vecteur d'initialisation produit deux orbites complètement différentes. Cela signifie que le GNPA-PWLCM est clairement conforme aux caractéristiques d'un bon crypto-système.

Pour étudier la sensinilité du générateur aux paramètres de contrôle nous avons aussi utilisé le taux de changement de bit r. Plus les résultats de simulation sont proches de 50%, plus le GNPA est sensible à la valeur initiale. Lorsque de deux séquences pseudo-aléatoires S_1 et S_2 de longueur L sont générées avec des conditions initiales proches, le taux de changement de bit correspondant peut être défini comme :

$$r = \frac{\sum_{j=1}^{L} S_1(j) - S_2(j)}{L} \cdot 100 \tag{3.1}$$

Où $S_1(j)$ et $S_2(j)$ sont respectivement les j- ème bits des sequences binaires pseudo-aléatoires S_1 et S_2. Le taux de changement de bit r est calculé avec le vecteur initial de valeur $X(0)^{'}$ et la clé initiale $[K_1, K_2, K_3, K_4, K_5, K_6, K_7, K_8]$ du GNPA, et le résultat obtenu est montré à la **Fig. 3.9**. Cette figure montre les différents taux de changement de bit entre la séquence pseudo-aléatoire générée par la clé d'origine et les 256 séquences pseudo-aléatoires générées par les 256 clés obtenues en changeant un seul bit dans la clé initiale. On peut voir que lorsque la clé initiale du

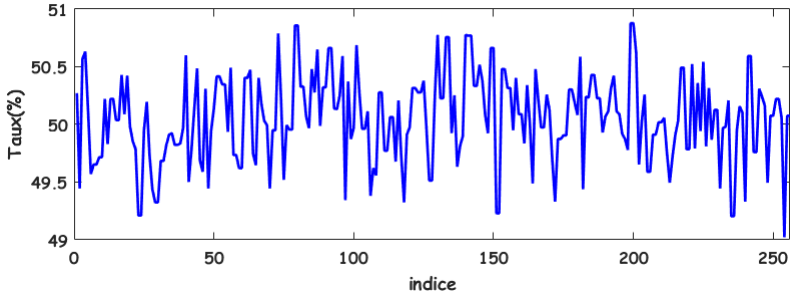


FIGURE 3.9 – Le Taux de changement séquence

GNPA subit un changement d'un bit, le taux de changement de bit de la séquence pseudo-aléatoire est très proche de l'idéal 50%, ce qui montre que le GNPA-PWLCM proposé est très sensible à ses paramètres .

3.5.5 L'entropie des informations

L'entropie de l'information est utilisée pour mesurer l'incertitude du flux binaire aléatoire généré par un GNPA. Il permet de déterminer l'imprédictibilité de certains messages. Lorsque le flux binaire est diffusé aléatoirement, l'entropie est la plus grande. Il est très important dans l'analyse de sécurité. Une entropie élevée signifie un générateur pseudo-aléatoire robust, tandis qu'une entropie faible signifie un générateur pseudo-aléatoire faible avec une certaine prédictibilité. L'entropie $H(S)$ d'une séquence S peut être estimée à l'aide de l'expression suivante :

$$H(S) = \sum_{i=1}^{2^s} p(S_i) \log_2 p(S_i) \tag{3.2}$$

Où $s \in \mathbb{N}^*$ est le nombre de bits de chaque élément de la séquence S, 2^s représente le nombre de symboles possibles dans la séquence, S_i peut avoire 2^s symboles différents dans la séquence S, $p(S_i)$ représente la probabilité de distribution de l'élément S_i dans la séquence S. Si S a 2^s éléments possibles, l'entropie devrait être $0 \leq H(S) \leq s$. La figure 3.10 montre l'entropie de 50 séquences aléatoires. chaque séquence possède 256 symboles doit $s = 8$ obtenue en réalisant la truncaténation du bloc de 64 bits produit à chaque itération en 8 blocs de 8 bits. Nous avons obtenu comme entropie moyenne 7.9991 proche de la valeur idéale qui est 8. Par conséquent, la séquence est imprédictible donc aléatoire.

fréquence

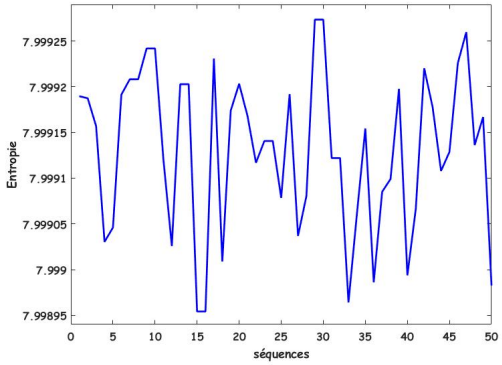


FIGURE 3.10 – Entropie de l'information de 50 séquences.

3.5.6 L'histogramme

Un histogramme est un outil couramment utilisé pour représenter les caractéristiques clés de la distribution des données de manière pratique. En d'autre terme, il permet de vérifier si tous les symboles de l'espace de phase sont représentés dans les séquences aléatoires avec des probabilités presque identiques. Nous avons évalué l'histogramme de la distribution des données des séquences x_a^t, x_b^t, x_c^t, x_d^t, x_e^t, x_f^t, x_g^t après 20000 itérations. La **Fig.3.11** montre que les distributions des séquences sont assez uniformes, confirmant ainsi son caractère d'équiprobabilité. Elle n'a pas été réalisé sur la séquence de 64 bits à cause du faite que le nombre minimale qui devrais être effectuer est de 2^{64}, ce qui représente un temps d'itération énorme.

3.5.7 La corrélation

Dans cette partie nous allons utilisé le coefficient de corrélation pour poursuivre le test de simulation. Le cœfficient de corrélation comme nous l'avons mentionné dans le chapitre 2, mesure la relation statistique entre les deux séquences pseudo-aléatoires. Si une séquence générée est aléatoire, le graphe d'autocorrélation doit être très proche de zéros. Pour deux séquences $X = f(x_0; x_1; ...; x_{L-1})$ et $Y = f(y_0; y_1; ...; y_{L-1})$, la corrélation entre les deux séquences peut être exprimée comme :

$$C_{xy} = \frac{\sum_{i=1}^{L}(x_i - \bar{x}) \cdot (y_i - \bar{y})}{\sqrt{\sum_{i=1}^{L}(x_i - \bar{x})^2} \cdot \sqrt{\sum_{i=1}^{L-1}(y_i - \bar{y})^2}} \tag{3.3}$$

Ou $\bar{x} = \sum_{i=0}^{L-1} \frac{x_i}{L}$, $\bar{y} = \sum_{i=0}^{L-1} \frac{y_i}{L}$ et L est la longueur de la séquence et $C_{XY} \in [-1; 1]$. Après le

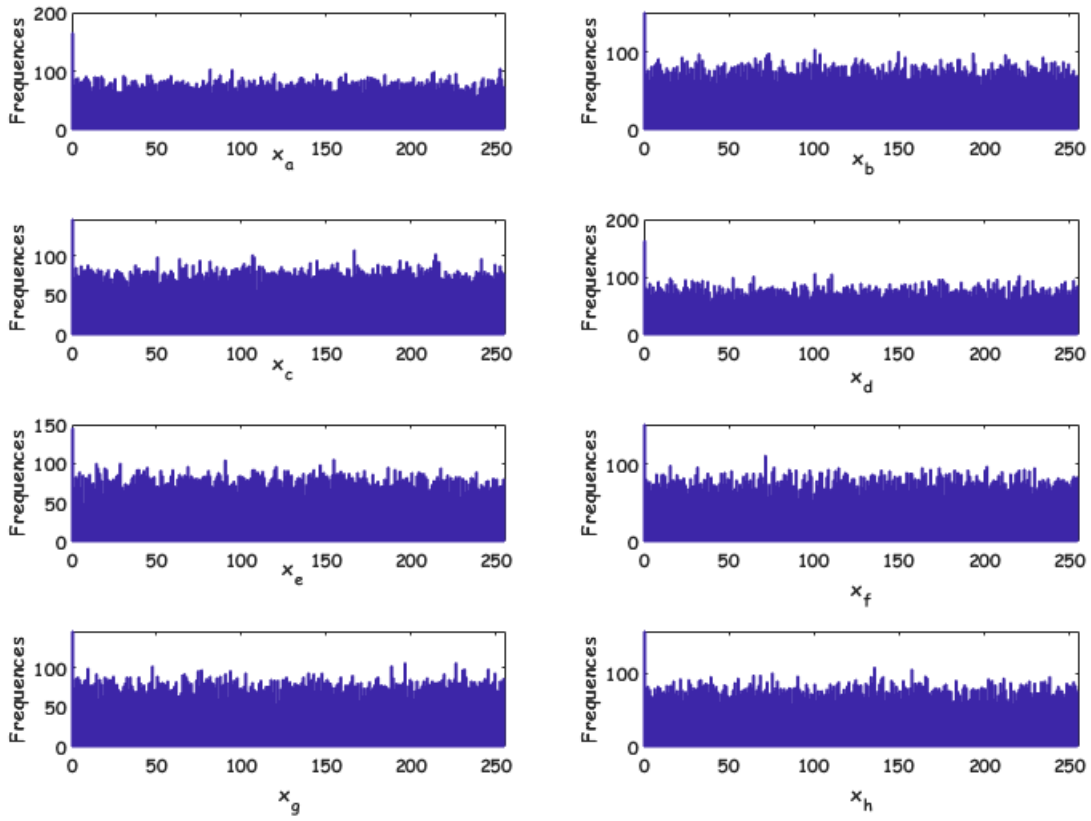


FIGURE 3.11 – Distribution des valeurs dans les séquences aléatoires x_a, x_b, x_c, x_d, x_e, x_f, x_g.

calcul du coefficient de corrélation, nous avons obtenu $C_{XY} = 8.1558 \cdot 10^{-4}$. Ce résultat nous permet de confirmer qu'il n'y a pas de corrélation entre X et Y, par conséquent, la sensibilité du système chaotique aux petits changements de paramètres est élevée.

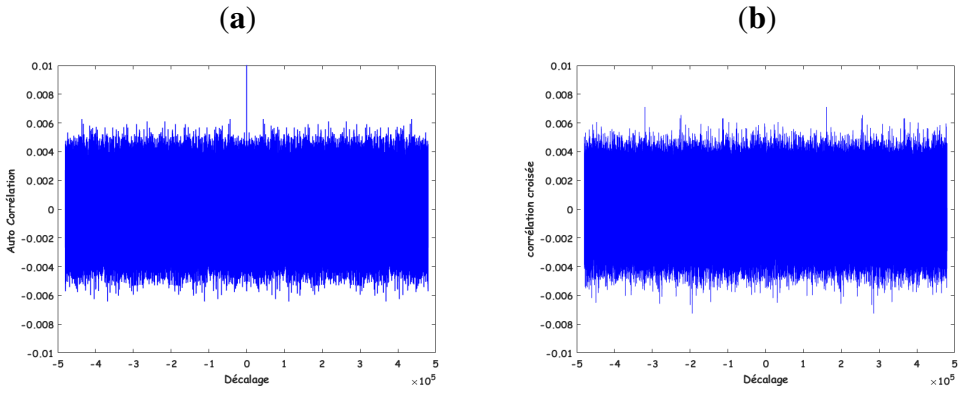


FIGURE 3.12 – Analyse de corrélation des séquences pour GNPA-PWLCM :(**a**) Auto-corrélation, (**b**) Cross-corrélation.

La **Fig.3.12** montre le graphe de Cross-corrélation et d'auto-corrélation entre les séquences pseudo-aléatoires générées par des clés sélectionnées aléatoirement. Les résultats de ce test

montrent que nous obtenons la valeur de corrélation proche de 0. Ce qui implique que les séquences pseudo-aléatoires produites par le générateur que nous proposons avec des paramètres presque identiques ne sont pas corrélées.

3.6 Le procédé de validation du générateur

3.6.1 La validation des résultats théoriques

Une simulation de la pré-synthèse du circuit de l'algorithme du GNPA proposé a été réalisée pour s'assurer de sa validité et vérifié que son fonctionnement est conforme aux résultats obtenus par simulation numérique sur MATLAB, avant de procéder à son implémentation sur le composant FPGA. Nous avons réalisé Cette simulation de pré-synthèse en concevant un modèles tesbench de notre générateur en Verilog, celui-ci est directement invoqué par le simulateur Xilinx intégré à l'outil de développement Vivado.

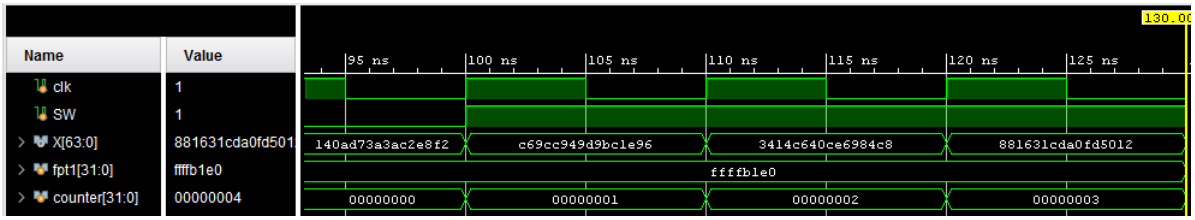


FIGURE 3.13 – Résultats de la simulation GNPA-PWLCM sur Vivado.

```
Command Window
>> hexValuer
hexValuer =
  4×16 char array
    '140AD73A3AC2E8F2' 'C69CC949D9BC1E96' '3414C640CE6984C8' '881631CDA0FD5012'
```

FIGURE 3.14 – Résultats de la simulation GNPA-PWLCM sur MATLAB.

Les résultats de simulation du GNPA-PWLCM mis en œuvre sur la plateforme Vivado 2018.3 à l'aide du langage Verilog HDL proposé dans cette section sont présentés à la **Fig.3.13**. Et les résultats de la mise en œuvre du logiciel MATLAB sont présentés à la **Fig.3.14**, et on peut voir que les résultats de la mise en œuvre matérielle sont complètement cohérents avec les résultats de la mise en œuvre logicielle. Ce qui traduit le bon fonctionnement de notre générateur de nombre pseudo-aléatoire.

3.6.2 La validation des résultats expérimentaux

Après l'étape de synthèse logique d'un circuit, la mise en œuvre de son algorithme consiste à adapter le circuit logique synthétisé aux ressources matérielles disponibles sur le FPGA ciblé. Pour cela, la démarche de conception se déroule en quatre étapes qui sont :

1. **Traduction** (*Translate*) : opération de fusion entre les Netlists (listes de nœuds, c'est-à-dire d'interconnexions et de composants logiques) résultant de la synthèse logique avec le fichier de contraintes spécifié ;
2. **Cartographie** (*Mapping*) : elle consiste à planifier la conception à l'intérieur des ressources disponibles du FPGA ciblé (qui peut être partiellement occupée par d'autres fonctions déjà intégrées) ;
3. **Placement et Routage** (*Placement and routage*) : place les composants élémentaires et les connectés physiquement, en respectant les contraintes précisées lors de la synthèse, afin d'obtenir un fichier de configuration ;
4. **Génération du fichier binaire** (*Generate programming file*) :A cette étape, un fichier binaire est produit pour la configuration physique du FPGA ciblé.

Donc, La mise en œuvre du générateur de nombres pseudo-aléatoires se termine par l'étape de génération du fichier binaire appelé "bitstream", pour la configuration physique de notre carte de prototypage FPGA. Ce fichier binaire qui va être transféré sur la carte FPGA et permettra de configurer le contenu de chaque LUT et de chaque interconnexion du FPGA pour que celui-ci implémente l'architecture de la **Fig3.6** que vous avons obtenu lors de la phase de synthèse. Tout ceci à partir du ficher de contrainte de carte de prototypage FPGA. C'est à partir du fichier de contrainte de la carte que nous avons sélectionné les entrées, les sorties, et les boutons de la carte qui seront utile pour le bon fonctionnement du générateur. les sorties de la carte sont représentées par les Pmod Header (JA, JB, JC, JD, JE) représentés à la **Fig3.15**. Chaque Pmod permet d'extraire 8 bits du générateur.

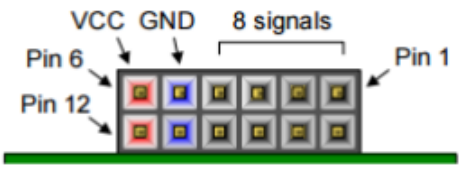


FIGURE 3.15 – Diagramme du Pmod Header.

Suivant le faite que le nombre de Pmod de la carte ne nous donne pas la possibilité d'extraire tous les 64 bits produit par notre générateur à chaque signal d'horloge, nous avons configuré le Pmod JB pour extraire les 8 bits produit par $x_a(t)$ et le Pmod JD pour l'extraction des 8 bits produit par $x_b(t)$.

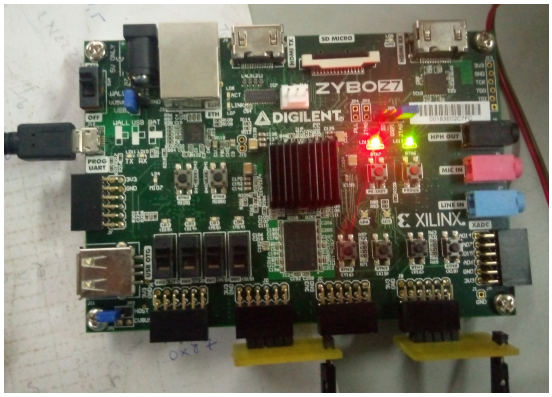


FIGURE 3.16 – Carte de prototypage FPGA ZYBO 7020.

Le "bitstream" obtenu après implémentation du code verilog est transféré sur le circuit FPGA via le câble USB. Une fois le circuit configuré, nous avons changé la valeur de l'interrupteur correspondant à "SW" pour vérifier le bon fonctionnement de notre générateur.

Le montage expérimental du laboratoire que nous avons utilisé, illustré à la **Fig.3.17**, permet de valider le PRNG basé sur la PWLCM que nous avons proposé.

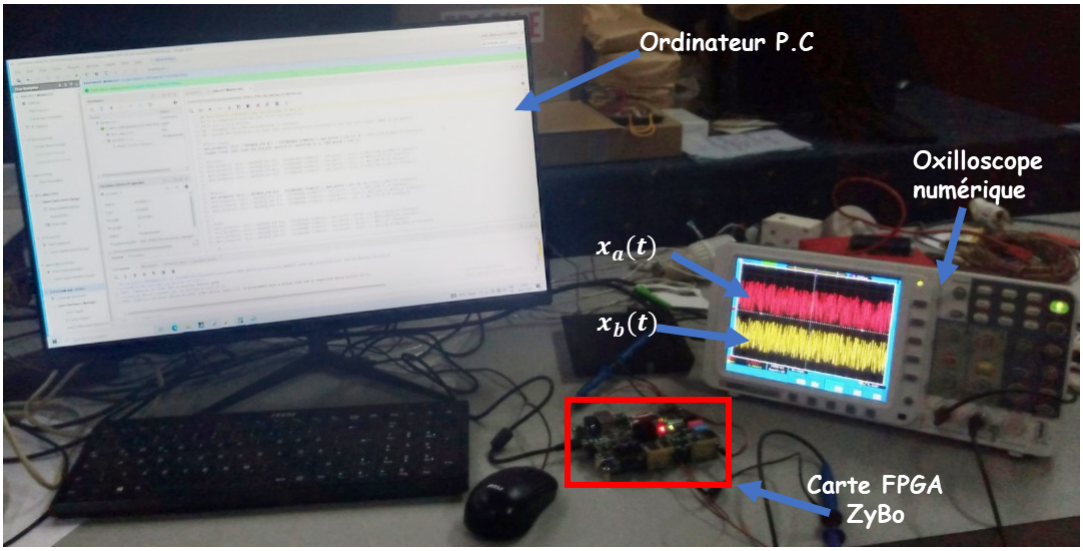


FIGURE 3.17 – Dispositif expérimentale.

Pour observer nos valeurs nous avons utilisé le Pmod B pour les valeur $x_a(t)$ et le Pmod D pour les valeurs de $x_b(t)$. A chaque observation, les 8 bits générés par chaque PWLCM du PRNG proposé sont capturés en temps réel par un convertisseur numérique-analogique. celui-ci

transforme les valeurs numériques codées sur 8 bits en des valeurs réel comprissent entre 0 et 3.3 volt (3.3V représente la tension de référence de la carte de prototypage, il correspond à 255 qui la plus grande valeur numérique codée sur 8 bits). Et elles sont affichées sur un oxilloscope numérique de marque OWON Smart OS7102V. La **Fig.3.18** montre les formes d'onde en temps réel du PRNG proposé et capturé sur l'oxilloscope où l'onde rouge représente le résultat de la conversion des valeurs numérique de $x_a(t)$ et la courbe en jaune celle de $x_b(t)$.

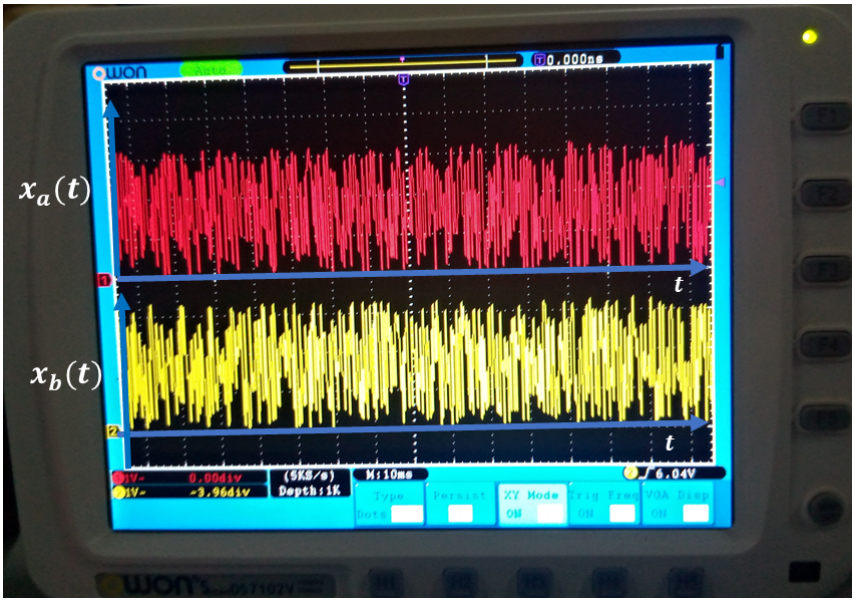


FIGURE 3.18 – L'évolution temporelle des sorties $x_a(t)$ et $x_b(t)$ capturée sur l'oxilloscope numérique.

La **Fig.3.19** montre le portait de phase obtenu en temps réel à partir du couple $(x_a(t), x_b(t))$ capturé sur l'oxilloscope. Ceci permet la validation de nos résultats.

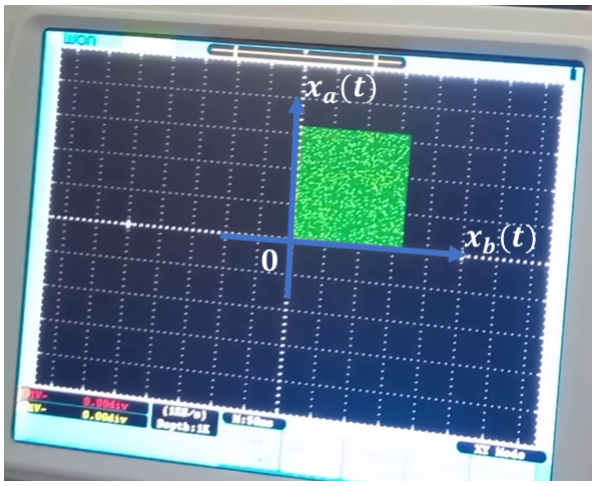


FIGURE 3.19 – Portrait de phase formé par les séquences x_a et x_b.

3.7 La comparaison du GNPA proposé avec la littérature

Ces dernières années, des GNPA basés sur les systèmes chaotiques ont été proposés dans la littérature. Bien qu'ils réussissent les tests statistiques, certains d'entre eux ont une architecture complexe pour être facilement appliqués dans l'ingénierie embarquée. Le tableau 3.5 montre le résumé de la comparaison des caractéristiques du générateur que nous avons proposé a ceux récemment publiée dans la littérature ([109–111]). Tous ces générateurs ont été implémentés sur FPGA et la comparaison est faites sur la base du nombre de ressources matériels d'utilisées (surface), de la performances (débit et fréquence de fonctionnement maximal). On peut voir d'après celui-ci, que l'architecture matérielle du générateur que nous proposons présente des performances beaucoup plus meilleur comparées aux autres. En matière de complexité, il est de loin le moins complexe avec seulement 90 tables de correspondance (LUT) et 65 bascules (FF) ce qui implique qu'il occupe moins d'espace et qu'il sera facile à embarquer. Sa faible complexité impacte directement sur sa vitesse et fait de lui le plus rapide avec un débit de 11,37 Gbit/s utile pour un traitement de données en temps réel.

TABLE 3.5 – Comparaison de la conception proposée avec d'autres dans la littérature.

ref	Type de GNPA	Source d'entropie	carte FPGA	LUT	FF	F_{max} (MHz)	débit (Mbps)
[109]	PUF-CPRNG 16-bits	PUF-Chaos	Virtex−7	631	1014	52	832
[110]	PRNG 16-bits	Piece-wise Linear chaotic map	Zynq 7000	4215	578	81	1296
[34]	PRNG-FWMHS 48-bits	HFWMS $5D$	Zynq XC7Z020	20517	2605	$138,331$	$15,37$
[111]	PRNG 64-bits	l'oscillateur du réseau de neurones de Hopfield	Zynq XC7Z020	37977	47195	$109,337$	$16,20$
PRNG proposé	GNPA-PWLCM 64−bits	PWLCM $8D$	Zynq XC7Z020	**90**	**65**	**177,809**	**11370**

Pour compléter la comparaison de notre générateur avec ceux de la littérature. Nous avons implémenté sur un même dispositif notre générateur avec celui présenté dans [78]. Dans cet article les auteurs proposent une méthode pour générer des séquences pseudo-chaotiques unidimensionnelles basée sur la récurrence du chat d'Arnold conventionnelle définie sur l'anneau entier $\mathbb{Z}_{2^n}$ ($n = 32$ et 64 bits). Les générateurs qu'ils proposent sont conçus à l'aide du langage Verilog HDL et sont implémentés à la fois dans le dispositif FPGA Xilinx Virtex-5 XC5VLX50T-1FFG1136 (ML505) et Altéra Cyclone Puce FPGA IV EP4CE115F29C7 (DE2-115). Etant donné que nous

ne disposons pas ces dispositifs FPGA, nous avons implémenté les générateurs qu'ils proposent sur la plateforme d'évaluation FPGA Zynq xc7z020clg400-1 que nous disposons. Ceci dans le but de disposer d'un même terrain de comparaison. L'architecture FPGA que nous avons adoptée pour cela est présentée à la **Fig3.6** Il est identique à celui qu'ils ont utilisé dans leur article. Les résultats obtenus après implémentation ont été inscrit dans le Tableau 3.6. Nous constatons que le générateur de [78] implémenté sur notre dispositif de prototypage fait usage de 10 module DSP ce qui explique les différences de LUT et FF obtenu par rapport à ceux qu'ils ont présenté dans leur article. Les résultats que nous avons obtenu de leur générateur montre, l'utilisation des modules DSP donc le rôle est de réaliser les opérations complexe présent dans la syntaxe vérilog du générateur dans le but de limité le nombre de ressource (LUT, FF) et d'augmenter le débit. Malgré cela le générateur que nous proposons est le plus rapide et le moins complexe des générateurs de nombre pseudo-aléatoire basé sur la récurrence du chat d'Arnold.

TABLE 3.6 – Les caractéristiques FPGA des générateurs [78] et GNPA-PWLCM .

FPGA utilisé	Virtex-5	Zynq xc7z020clg400-1	
Méthode	[78]	[78]	GNPA-PWLCM
LUT	2261	223	90
FF	192	179	65
DSP	0	10	0
Fréquence(MHz)	94,55	170.444	172,74
Nombre de bit par itération	64	64	64
Débit (Gbps)	6,05144	10,908	11,370

La méthode de conception des GNPA moins complexe et rapide que nous proposons permet de réaliser l'équilibre entre le faible coût de ressource, un débit élevé et une réussite aux différents tests statistiques. Cette méthode, permet de résoudre le problème de dégradations dynamiques des systèmes chaotiques lors de leur implémentation dans des dispositifs numériques comme c'est souvent le cas.

3.8 Conclusion

Dans ce chapitre le but principal était de créer un générateur chaotique parfaitement aléatoire, de faible complexité et ultra-rapide. Pour cela nous avons combiné 8 systèmes chaotiques identiques multidimensionnels (PWLCM) réalisant des opérations internes sur 1 bit que nous avons choisi en raison de leurs faibles complexités. La conception que nous avons faite a été établie en utilisant le langage Vérilog et l'analyse de l'aléa des séquences aléatoires générées à partir des tests statistiques

a présenté de bon résultat. Par la suite, une comparaison des performances et du coût matériel de notre générateur avec la littérature nous a permis d'affirmer qu'il fait partir des générateurs de nombres pseudo-aléatoires adapté pour créer une dynamique aléatoire dans des dispositifs de faible complexité.

CONCLUSION GÉNÉRALE

A - Résumé de la recherche

Les études menées dans le cadre de ce travail de thèse s'articulent autour de l'implémentation d'une récurrence du chat d'Arnold (RCA) comme générateur de nombre pseudo-aléatoire (GNPA), outil de base pour le chiffrement de l'information. Nous avons tenté dans cette thèse, d'apporter certaines solutions aux problèmes tels que la dégradation dynamique, les courtes périodes, causés par la représentation binaire des systèmes dynamiques lors de leurs implémentations dans des dispositifs numériques. Étant donné que ces problèmes rendent les séquences aléatoires générées prédictible, causant ainsi la vulnérabilité des systèmes de cryptographie aux différentes attaques informatiques. Cette thèse est divisée en trois chapitres. Le premier chapitre passe en revue un grand nombre de GNPA utilisant comme source d'entropie des systèmes linéaires pour la conception des GNPA linéaires. Ces générateurs sont certes rapides, mais produisent des séquences aléatoires qui sont statistiquement faibles. Les GNPA dites modernes sont conçus en utilisant comme source d'entropie des systèmes chaotiques. Ces systèmes sont de bon candidat pour la conception des GNPAs grâce à leur propriété interne qui sont l'ergodicité, la mixité et la sensibilité liée à la positivité des exposants de Lyapunov. Par contre leur implémentation dans des dispositifs numériques entraîne un problème de dégradation dynamique dû à leur représentation binaire. Pour résoudre ce problème, différentes solutions sont proposées dans la littérature. Ces solutions améliorent certes la qualité statistique de la séquence mais ne tient pas compte du coût de conception et du débit. Ce chapitre se termine en donnant quelques notions sur les systèmes dynamiques. Le chapitre 2, présente les fondements mathématiques qui sont liés à la conception du GNPA que nous proposons. Il débute par une mise au point des différents outils tels que l'exposant de lyapunov, l'entropie, la sensibilité aux conditions initiales et la corrélation qui permettent de caractériser le degré d'aléa d'une séquence aléatoire. Par la suite nous décrivons la méthodologie que nous avons utilisée pour apporter une solution au problème posé. Elle repose sur la conception de nouveaux systèmes dynamiques, utilisant la récurrence du chat d'Arnold comme système de base. Il a la particularité de présenter un caractère aléatoire peu importe la valeur de ces paramètres de

contrôle, ce qui n'est pas le cas pour la plupart des systèmes chaotiques. Dans le but d'augmenter ses périodes nous l'avons modifié en introduisant un terme additif de l'addition modulaire. Cela nous a fait passer pour seulement 4 bits de précision, d'une période de 12 pour la RCA classique à $4.8 \cdot 10^{12}$ pour la RCA modifié que nous avons nommé la PWLCM. Enfin, pour augmenter le débit et réduire la complexité du système obtenu nous avons réduit tous les opérations à 1 bit et une extension du système obtenu en dimension 8 a été necessaire pour augmenter le niveau d'aléa et le nombre de paramètres de contrôles. Le troisième chapitre, concerne la proposition d'un nouveau algorithme de générateur de nombres pseudo-aléatoires à base de la PWLCM monobit en dimension 8. Cette contribution porte sur plusieurs aspects, à savoir le choix des systèmes chaotiques convenables à la conception des GNPA rapide et de faible complexité, la représentation binaire adéquate et le procédé d'extraction des nombres pseudo-aléatoires. Tout d'abord, une étude comparative des résultats obtenus de l'implémentation FPGA des différents systèmes chaotiques proposés a été présentée dans la première section du chapitre, en vue de sélectionner le plus adapté à notre algorithme. Ensuite, nous avons détaillé la structure interne de notre générateur de nombres pseudo-aléatoires conçu sur la mise en réseau de 8 PWLCM monobit en dimension 8, et implémenté sur le dispositif FPGA du laboratoire d'électronique de l'université de Yaoundé 1. Enfin, nous avons par une analyse de sécurité, pu confirmer les bonnes performances de notre algorithme qui sont bien adaptées aux applications en temps réel. Comparé aux récents générateurs de la littérature, notre algorithme dévoile des performances compétitives, avec un bon compromis entre le coût du matérielle, le débit et de bonne propriété statistique. Le dernier point abordé correspond à la validation de l'algorithme du GNPA proposé par une implémentation numérique sur un composant FPGA ZYBO 7020.

B - Perspective des travaux future

Les travaux de recherche développés dans le cadre de cette thèse apportent des contributions prometteuses en matière d'exploitation des systèmes chaotiques dans la conception des GNPA, et laissent entrevoir, par ailleurs, quelques perspectives et élargissements aussi bien sur le plan théorique que pratique :

— Du point de vue théorique, le système sur lequel nous avons porté notre attention est la récurrence du chat d'Arnold. Cependant des études plus approfondies peuvent encore être réaliser ceci dans le but d'obtenir des systèmes dynamique beaucoup plus complexe que la PWLCM.

— L'architecture du GNPA que nous avons proposé présente une solution originale en ce qui concerne le traitement en temps réel pour des applications sur tout type de dispositif numérique. Néanmoins, il sera très utile de le combiner avec d'autres mécanismes de cryptographie au sein des protocoles de sécurité.

— D'un point de vue pratique, l'expérimentation de l'algorithme du GNPA proposé sur FPGA constitue une première version qui peut subir différentes optimisations, par l'augmentation du débit de génération de nombres pseudo-aléatoires d'une part, et la réduction des ressources occupées d'autre part, notamment pour répondre aux exigences des systèmes de chiffrement à bas coût ;

— Les expériences, résultats et connaissances acquises au cours de ce travail nous permettront d'explorer de nouvelles perspectives pour les générateurs de nombres pseudo-aléatoires basés sur la récurrence du chat D'Arnold linéaire par morceaux (PWLCM).

ANNEXE 1 : ENVIRONNEMENT D'IMPLÉMENTATION MATÉRIEL FPGA

I- Description de carte de prototypage ZYBO 7020

Le Zybo Z7 est un logiciel embarqué riche en fonctionnalités et prêt à l'emploi. C'est une carte de développement de circuits numériques construit autour de la famille Xilinx Zynq-7000. La famille Zynq est basée sur l'architecture Xilinx All Programmable System-on-Chip (AP SoC) [83], qui intègre étroitement un processeur ARM Cortex-A9 double cœur avec la logique Xilinx 7-series. Le Zybo Z7 entoure le Zynq d'un riche ensemble de périphériques multimédia et de connectivité pour créer un formidable ordinateur monocarte, avant même de considérer la flexibilité et la puissance ajoutées par le FPGA. Nous avons utilisé dans notre cas la carte Xilinx Zybo (xc7z020clg400-1). Cette gamme de FPGA offre un environnement de conception adapté au prototypage d'applications variées, dont celles des systèmes numériques à usage général et des systèmes embarqués. Elle intégre des fonctionnalités vidéo du Zybo Z7, avec une entrée HDMI, une sortie HDMI et une bande passante DDR3L élevée et des connexions de matériel supplémentaire facilitée par les connecteurs Pmod du Zybo Z7, permettant d'accéder au catalogue de Digilent de plus de 70 cartes périphériques Pmod, y compris des contrôleurs de moteur, des capteurs, des écrans, etc.

Les caractéristiques du ZYNQ-7 ZC702 sont les suivantes :

— 53200 slices LUT (éléments logiques)

— 106400 bascules (Filp Flops) ;

— 140 bloc de RAM rapide 32-Kb ;

— 220 slices DSP ;

— 125 MHz vitesses d'horloge.

— Processeur Cortex-A9 double cœur 667MHz ;

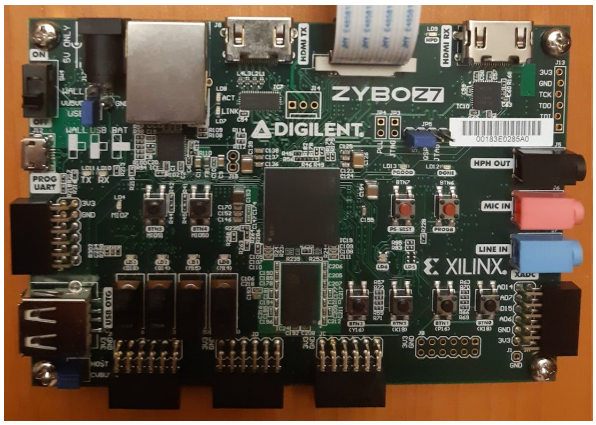


FIGURE 3.20 – Dispositif de prototypage FPGA : Zybo Z7

— Contrôleur de mémoire DDR3L avec 8 canaux DMA et 4 ports esclaves AXI3 hautes performances ;

— Contrôleurs périphériques haut débit : Ethernet 1G, USB 2.0, SDIO ;

— Contrôleurs périphériques à faible bande passante : SPI, UART, PEUT, I2C ;

— Programmable à partir de JTAG, Quad-SPI flash etcarte Micro SD ;

— Logique programmable équivalente à Artix-7 FPGA.

II- Définition du Langage verilog HDL

Verilog est un langage de description matérielle, "(Hardware description language, (HDL))" utilisé pour définir la structure et/ou le comportement des circuits numériques. A la différence du VHDL, Verilog est aujourd'hui le plus largement utilisé dans l'industrie. Le grand nombre de paramètre de contrôle et la dimension de la PWLCM et sa dimension que nous allons utiliser pour la conception du générateur de nombres pseudo-aléatoires proposé nous impose l'utilisation de l'approche textuelle basée sur un langage de description matériel (Verilog), liée davantage aux processus algorithmiques. L'avantage d'une telle approche réside dans son caractère exécutable : une spécification décrite en verilog HDL peut être vérifiée par simulation, avant que le design détaillé ne soit établi. De plus, les outils de conception assistée par ordinateur (CAO) permettent de passer directement d'une description fonctionnelle en verilog à un schéma de porte logique qui a révolutionné les méthodes de conception de circuits numériques, ASIC ou FPGA.

III- Outil de CAO : Vivado 2018.3

Pour la mise en œuvre de notre GNPA, nous avons utilisé le navigateur de projet Vivado 2018.3. Cet outil, téléchargeable sur le site Xilinx, constitue un environnement de conception extrêmement performant, qui intègre tous les mécanismes nécessaires à la réalisation d'un circuit numérique, allant de la description comportementale en code Verilog à la génération du schéma correspondant en porte logique. La Fig.3.22 représente l'interface de programation du logiciel vivado.

FIGURE 3.21 – Logiciel vivado 2018.3

;

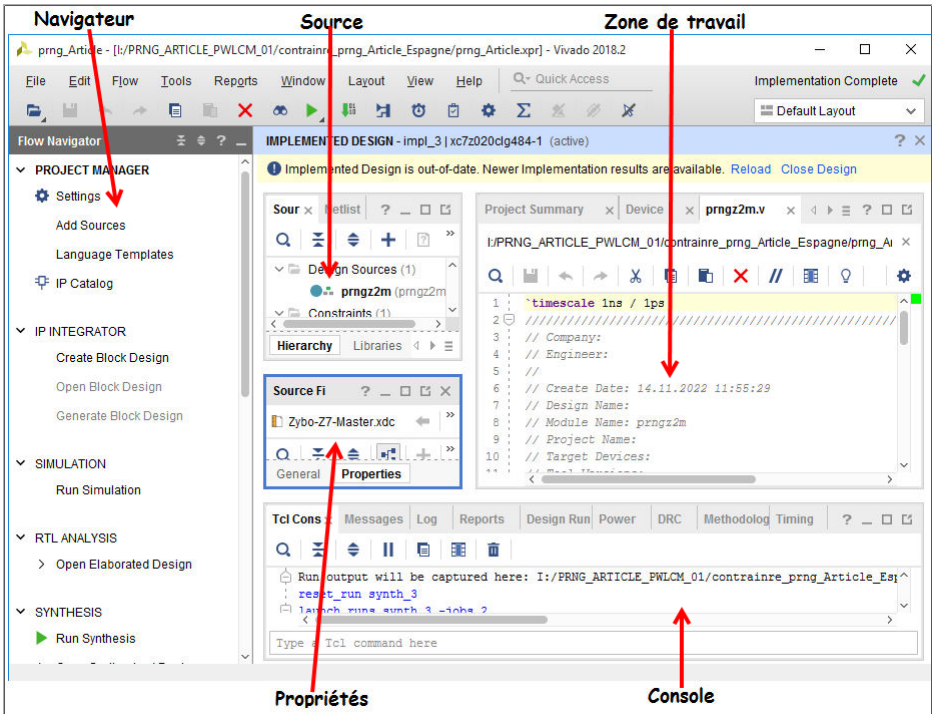


FIGURE 3.22 – Interface du logiciel vivado 2018.3

;

IV- Analyse des performances d'un circuit synthétisé

Les paramètres représentatifs dont dépendent les performances des implémentations de systèmes embarqués sur FPGA, tel que les limites temporelles des éléments séquentiels, la valeur

minimale de la période d'horloge et les broches d'entrée/sortie physiques à utiliser sur le FPGA ciblés, sont des contraintes strictes qui agissent fortement sur les outils d'implémentations. D'où l'intérêt d'associer à un fichier de contraintes toutes les spécifications liées au fonctionnement de l'algorithme du circuit que l'on souhaite implémenter. Ainsi, nous avons vérifié la conformité du circuit implémenté aux contraintes préétablies suite au placement et au routage. Cette étape permet d'évaluer les performances réelles de algorithme du circuit grâce à l'estimation des paramètres déterminants suivants :

— **La surface :** Une mise en oeuvre éfficace cherche à réduire au maximum le taux d'occupation de surface d'un FPGA. Elle s'exprime en silice qui, dans le cas d'un Xilinx Zynq-7 ZC702, contient $53,200$ LUT (*Look-Up Table* ou table de correspondance) et $106,400$ bascules (Flip-Flops) etc. Il est toutefois intéressant de noter que pour un algorithme de chiffrement, trop peu de logique peut entraîner une très grande fragilité de l'algorithme.

— **La fréquence maximale de fonctionnement :** L'estimation correcte de la fréquence de fonctionnement d'un système est liée à l'étape de placement/routage et aux contraintes imposées par le concepteur. Par conséquent, il est essentiel de contrer la conception par les limites de temps ainsi que la valeur minimale de la période d'horloge, afin d'établir une optimisation en termes de vitesse de fonctionnement. La fréquence de fonctionnement maximale est calculée à l'aide de l'équation 3.4.

$$F_{max} = \frac{1}{T - WNS} \tag{3.4}$$

où T représente la période d'horloge minimale en ns, et WNS (*worst negative slacks en anglais*) est le pire écart négatif et s'exprime en ns.

— **Le débit en sortie :** C'est le paramètre le plus important dans l'évaluation des cryptosystèmes, puisqu'il nous renseigne sur la capacité qu'un algorithme à chiffrer un message, et détermine par conséquent ses domaines d'applications privilégiés. Il s'exprime en Mbps (Mégabits par seconde) et se calcule comme suit :

$$d = \frac{N \times F_{max}}{Latence} \tag{3.5}$$

Où N est le nombre de bits produits à chaque cycle d'horloge.

ANNEXE 2 : ENVIRONNEMENT DE SIMULATION NUMÉRIQUE MATLAB

MATLAB (MATtrix LABoratory) est un environnement de calcul numérique de haut niveau conçu par la société américaine Mathworks. Il est particulièrement efficace pour les calculs matriciels, car sa structure de données internes est basée sur des matrices. Par exemple, il possède également d'excellentes capacités graphiques pour visualiser des objets mathématiques complexes. Son fonctionnement repose sur un langage de programmation interprété qui permet un développement très rapide. Les langages compilés tels que $C++$ et FORTRAN conviennent aux applications nécessitant plus de temps de calcul. Les simulations numériques qui sont présentées dans cette thèse ont été realisées à l'aide de MATLAB $2018b$ sous un processeur Intel $i5-4590$ CPU à $3,30$ GHz avec 4 Go de SDRAM DDR3.

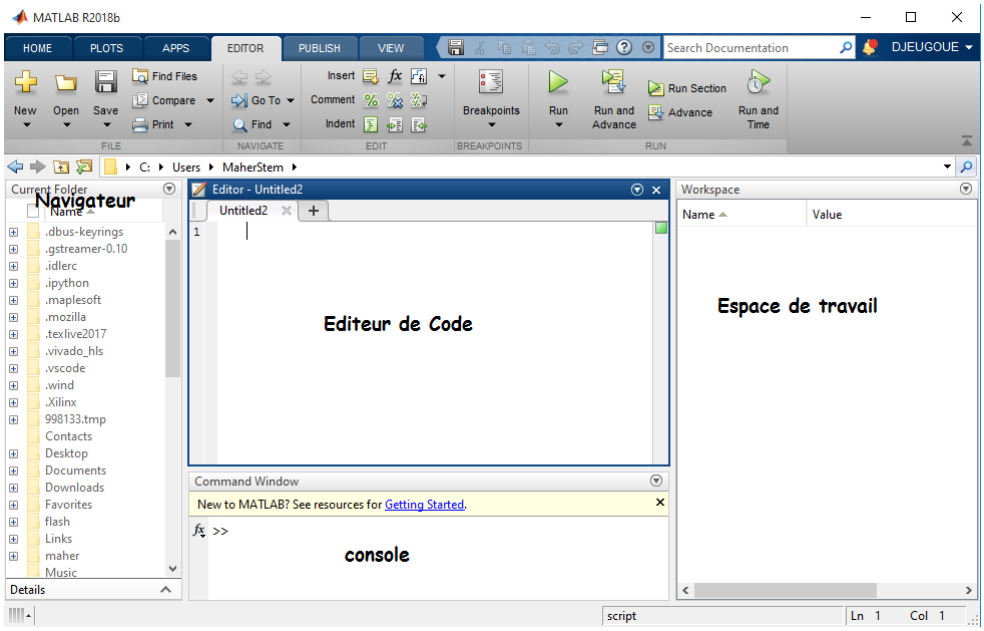


FIGURE 3.23 – Interface du logiciel MATLAB

Bibliographie

[1] Panda, Amit Kumar and Ray, Kailash Chandra, "A coupled variable input LCG method and its VLSI architecture for pseudorandom bit generation", *IEEE Transactions on Instrumentation and Measurement*. vol. 69, pp. 1011-1019. 2019.

[2] Bakiri, Mohammed and Guyeux, Christophe and Couchot, Jean-François and Marangio, Luigi and Galatolo, Stefano, "A hardware and secure pseudorandom generator for constrained devices", *IEEE Transactions on Industrial Informatics*, vol.14, pp.3754-3765, 2018.

[3] Pandey, Jai Gopal and Goel, Tarun and Karmakar, Abhijit. "A high-performance and area-efficient VLSI architecture for the PRESENT lightweight cipher", *2018 31st International Conference on VLSI Design and 2018 17th International Conference on Embedded Systems*, pp.392-397, 2018.

[4] Da Xu, Li and He, Wu and Li, Shancang, "Internet of things in industries : A survey", *IEEE Transactions on industrial informatics*, vol.10, pp.2233-2243, 2014.

[5] Li, Shancang and Tryfonas, Theo and Li, Honglei, "The Internet of Things : a security point of view",*Internet Research*, vol.26, pp.337-359, 2016.

[6] Lamkuche, Hemraj Shobharam and Pramod, Dhanya "CSL : FPGA implementation of lightweight block cipher for power-constrained devices" *International Journal of Information and Computer Security*, vol.12, pp.349-377, 2020.

[7] Knuth, Donald, "Deciphering a linear congruential encryption", *IEEE Transactions on Information Theory*, vol.31, pp.49-52, 1985.

[8] Pasalic, Enes. "On guess and determine cryptanalysis of LFSR-based stream ciphers", *IEEE Transactions on Information Theory*, vol.55, pp.3398-3406, 2009.

[9] Matsumoto, Makoto and Nishimura, Takuji ; "Mersenne twister : a 623-dimensionally equidistributed uniform pseudo-random number generator", *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, vol.8, pp.3-30, 1998.

[10] Marsaglia, George and Tsay, Liang-Huei, "Matrices and the structure of random number sequences", *Linear algebra and its applications*, vol.67, pp.147-156, 1985.

[11] López, Amalia Beatriz Orúe and Encinas, Luis Hernández and Muñoz, Agustín Martín and Vitini, "Fausto Montoya. A lightweight pseudorandom number generator for securing the Internet of Things", *IEEE access*, vol.5, pp.27800-27806, 2017.

[12] Frieze, Alan M and Hastad, Johan and Kannan, Ravi and Lagarias, Jeffrey C and Shamir, Adi, "Reconstructing truncated integer variables satisfying linear congruences" *SIAM Journal on Computing*, vol. 17, pp.262-280, 1988.

[13] Stern, Jacques. "Secret linear congruential generators are not cryptographically secure", *28th Annual Symposium on Foundations of Computer Science*, pp.421-426, 1987.

[14] Bakiri, Mohammed and Guyeux, Christophe and Couchot, Jean-François and Oudjida, Abdelkrim Kamel, "Survey on hardware implementation of random number generators on FPGA : Theory and experimental analyses", *Computer Science Review*, vol.27, pp.135-153, 2018.

[15] Jessa, Mieczyslaw and Walentynowicz, Marcin, "Statistical properties of number sequences generated by 1D chaotic maps considered as a potential source of pseudorandom number sequences ICECS 2001" *8th IEEE International Conference on Electronics, Circuits and Systems*, vol.1, pp.449-455, 2001.

[16] Addabbo, Tommaso and Alioto, Massimo and Fort, Ada and Rocchi, Santina and Vignoli, Valerio, "Low-hardware complexity prbgs based on a piecewise-linear chaotic map",*IEEE Transactions on Circuits and Systems II : Express Briefs*, vol.53, pp.329-333, 2006.

[17] Addabbo, Tommaso and Alioto, Massimo and Fort, Ada and Pasini, Antonio and Rocchi, Santina and Vignoli, Valerio, "A class of maximum-period nonlinear congruential generators derived from the Rényi chaotic map", *IEEE Transactions on Circuits and Systems I : Regular Papers*, vol.54, pp.816-828, 2007.

[18] Zhou, Yicong and Hua, Zhongyun and Pun, Chi-Man and Chen, CL Philip, "Cascade chaotic system with applications", *IEEE transactions on cybernetics*, vol.45, pp.2001-2012, 2014.

[19] Wong, Kwok-Wo and Lin, Qiuzhen and Chen, Jianyong, "Simultaneous arithmetic coding and encryption using chaotic maps", *IEEE Transactions on Circuits and Systems II : Express Briefs*, vol.57, pp.146-150, 2010.

[20] Zhou, Yicong and Bao, Long and Chen, CL Philip, "A new 1D chaotic system for image encryption". *Signal processing*, vol.97, pp.172-182, 2014.

[21] Arroyo, David and Rhouma, Rhouma and Alvarez, Gonzalo and Li, Shujun and Fernandez, Veronica, "On the security of a new image encryption scheme based on chaotic map lattices", *Chaos : An Interdisciplinary Journal of Nonlinear Science*, vol.18, pp.1-8, 2008

[22] Wu, Xiaogang and Hu, Hanping and Zhang, Baoliang, "Parameter estimation only from the symbolic sequences generated by chaos system" *Chaos, Solitons & Fractals*, vol.22, pp.359-366, 2004.

[23] Gao, Tiegang and Chen, Zengqiang. "A new image encryption algorithm based on hyper-chaos", *Physics letters A*, vol.372, pp.394-400, 2008.

[24] Arnold, Vladimir Igorevich and Avez, André, "Ergodic problems of classical mechanics", *Mathematical morphose serie*, vol.9, pp.1-286, 1968.

[25] Kanso, Ali and Ghebleh, Mohammad and Alazemi, Abdullah, "Efficient image encryption scheme based on 4-Dimensional Chaotic Maps", *Informatica*, vol.31, pp.793-820, 2020.

[26] Zhang, Yinxing and Hua, Zhongyun and Bao, Han and Huang, Hejiao and Zhou, Yicong, "An n-Dimensional Chaotic System Generation Method Using Parametric Pascal Matrix, *IEEE Transactions on Industrial Informatics*, vol.18, pp.8434-8444, 2022.

[27] Gnyamsi Nkuigwa, Gaetan Gildas and Djeugoue Nzeuga, Hermann and Fouda, JS and Sabat, Samrat L and Koepf, Wolfram, "An extendable key space integer image-cipher using 4-bit piece-wise linear cat map", *Multimedia Tools and Applications*, vol.1, pp.1-23, 2022.

[28] Zareai, Delavar and Balafar, Mohammadali and Feizi Derakhshi, Mohammad Reza. "A new Grayscale image encryption algorithm composed of logistic mapping, Arnold cat, and image blocking", *Multimedia Tools and Applications*, vol.80, pp.18317-18344, 2021.

[29] Li, Chengqing and Tan, Kai and Feng, Bingbing and Lu, Jinhu, "The Graph Structure of the Generalized Discrete Arnold's Cat Map", *IEEE Transactions on Computers*, vol.71, pp.364-377, 2021.

[30] Kumar, Nishant and Jain, Sourabh, "A Review of Performance Comparison on Chaos Based Image Encryption Technique", *International Journal of Innovative Science and Research Technology*, vol.7, pp.2456-2165, 2022.

[31] Dana, Itzhack and Chernov, Vladislav E, "Chaotic diffusion on periodic orbits : The perturbed Arnold cat map", *Phys. Rev*, vol.67, pp.1-7, 2003.

[32] Hermann, Djeugoue and Gildas, Gnyamsi Gaetan and Fouda, Jean Sire Armand Eyebe and Koepf, Wolfram, "On the implementation of large period piece-wise linear Arnold cat map", *Multimedia Tools and Applications*, vol.1, pp.1-18, 2022.

[33] Miguel Garcia-Bosque and Adrián Pérez-Resa and Carlos Sánchez-Azqueta and Concepción Aldea and Santiago Celma. "Chaos-Based Bitwise Dynamical Pseudorandom Number Generator On FPGA", *IEEE Transactions on Instrumentation and Measurement*, vol.68, pp.291-293, 2019.

[34] Yu, Fei and Li, Lixiang and He, Binyong and Liu, Li and Qian, Shuai and Zhang, Zinan and Shen, Hui and Cai, Shuo and Li, Yi, "Pseudorandom number generator based on a 5D hyperchaotic four-wing memristive system and its FPGA implementation", *European Physical Journal : Special Topics*. vol.230, pp.1763-1772, 2021.

[35] Nalla Anandakumar and N. and Sanadhya and Somitra Kumar and Hashmi and Mohammad S, "FPGA-Based True Random Number Generation Using Programmable Delays in Oscillator-Rings", *IEEE Transactions on Circuits and Systems II : Express Briefs*, vol.67, pp.570-574, 2020

[36] Cicek, Ihsan and Pusane, Ali Emre and Dundar, Gunhan, "An Integrated Dual Entropy Core True Random Number Generator", *IEEE Transactions on Circuits and Systems II : Express Briefs*, vol.64, pp.329-333, 2017.

[37] Johnson, Anju P. and Chakraborty, Rajat Subhra and Mukhopadyay, Debdeep, "An Improved DCM-Based Tunable True Random Number Generator for Xilinx FPGA" *IEEE Transactions on Circuits and Systems II : Express Briefs*, vol.64, pp.452-456, 2017

[38] Bucci, M. and Germani, L. and Luzzi, R. and Tommasino, P. and Trifiletti, A. and Varanonuovo, M, "A high speed truly IC random number source for smart card microcontrollers", *9th International Conference on Electronics, Circuits and Systems*, vol.1, pp.239-242, 2002.

[39] Holman, W.T. and Connelly, J.A. and Dowlatabadi, A.B. "An integrated analog/digital random noise source", *IEEE Transactions on Circuits and Systems I : Fundamental Theory and Applications*, vol.44, pp.521-528, 1997.

[40] L'Ecuyer, Pierre, "Uniform Random Number Generators : A Review", *Proceedings of the 29th Conference on Winter Simulation*, vol.8, pp.127-134, 1997.

[41] Seda Arslan Tuncer and Turgay Kaya, "True Random Number Generation from Bioelectrical and Physical Signals", *Computational and Mathematical Methods in Medicine*, vol.2018, pp.1-11, 2018.

[42] Bakiri, Mohammed, "Hardware implementation of a pseudo random number generator based on chaotic iteration", *Université Bourgogne Franche-Comté*, vol.1, pp.1-140, 2018.

[43] Oudjida, AK and Benamrouche, D and Liem, M, "Front-end ip development : Basic know-how", *International Conference on Design & Technology of Integrated Systems in Nanoscale Era*, pp.60-63, 2007.

[44] Freeman, Ross H and Hsieh, Hung-Cheng, "Distributed memory architecture for a configurable logic array and method for using distributed memory", *United States Patent*, pp.1-19, 1994,

[45] F. Hussain, M. M. Iqbal, H. Parvez, M. Rashid, "Exploring FPGA Logic Block Architecture for Reduced Configuration Memory," *Advances in Electrical and Computer Engineering*. vol.22, pp.15-24, 2022.

[46] Bakiri, Mohammed and Couchot, Jean-François and Guyeux, Christophe, "CIPRNG : A VLSI Family of Chaotic Iterations Post-Processings for $\mathbb{F}_2$ -Linear Pseudorandom Number Generation Based on Zynq MPSoC", *IEEE Transactions on Circuits and Systems I : Regular Papers*, vol. 65, pp.1628-1641, 2018.

[47] L'Ecuyer, P. and Panneton, F, "Fast random number generators based on linear recurrences modulo 2 : overview and comparison", *Proceedings of the Winter Simulation Conference*. pp. 1-11. 2005

[48] Fishman, George S. and Moore, III, Louis R. "An Exhaustive Analysis of Multiplicative Congruential Random Number Generators with Modulus $2^{31}-1$", *SIAM J. Sci. Stat. Comput*, vol.7, pp.24-45, 1986.

[49] Banks, Simon and Beadling, Philip and Ferencz, Andras, "FPGA Implementation of Pseudo Random Number Generators for Monte Carlo Methods in Quantitative Finance", *2008 International Conference on Reconfigurable Computing and FPGAs*, pp.271-276, 2008.

[50] Marsaglia, George. "Xorshift RNGs". *Journal of Statistical Software.*, vol.8, pp.1-6, 2003.

[51] George Marsaglia and Arif Zaman, "A New Class of Random Number Generators", *The Annals of Applied Probability*, vol.1, pp.462-480, 1991.

[52] Katti, Raj S. and Srinivasan, Sudarshan K, "Efficient hardware implementation of a new pseudo-random bit sequence generator", *IEEE International Symposium on Circuits and Systems*, pp.1393-1396, 2009.

[53] Robert Tausworthe, "Random Numbers Generated by Linear Recurrence Modulo Two", *Mathematics of Computation*, vol.19, pp.201-209, 1965.

[54] Erkek, Esra and Tuncer, Taner. "The implementation of ASG and SG Random Number Generators" .*2013 International Conference on System Science and Engineering (ICSSE)*, pp.363-367, 2013.

[55] Thomas, David Barrie and Luk, Wayne, "FPGA-optimised high-quality uniform random number generators", *Proceedings of the 16th international ACM/SIGDA symposium on Field programmable gate arrays*, pp.235-244, 2008.

[56] Thomas, David B and Luk, Wayne, "Fpga-optimised uniform random number generators using luts and shift registers", *2010 International Conference on Field Programmable Logic and Applications*, pp.77-82, 2010.

[57] Thomas, David B and Luk, Wayne. "The LUT-SR family of uniform random number generators for FPGA architectures", *IEEE transactions on very large scale integration (vlsi) systems*, vol.21, pp.761-770. 2012.

[58] Matsumoto, Makoto and Kurita, Yoshiharu, "Twisted GFSR generators", *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, vol.2, pp.179-194, 1992.

[59] Banks, Simon and Beadling, Philip and Ferencz, Andras, "FPGA implementation of pseudo random number generators for Monte Carlo methods in quantitative finance", *International Conference on Reconfigurable Computing and FPGAs*, pp.271-276, 2008.

[60] , Dalal, Ishaan L and Harwayne-Gidansky, Jared and Stefan, Deian, "On the fast generation of long-period pseudorandom number sequences", *IEEE Long Island Systems, Applications and Technology Conference*, pp.1-9, 2008.

[61] , Li, Yuan and Jiang, Jiang and Cheng, Hanqiang and Zhang, Minxuan and Wei, Shaojun, "An efficient hardware random number generator based on the mt method", *2012 IEEE 12th International Conference on Computer and Information Technology*, pp.1011-1015, 2012.

[62] . Wu, Shengfei and Jiang, Jiang and Fu, Yuzhuo, "Hardware architecture for the parallel generation of long-period random numbers using mt method", *CCF National Conference on Computer Engineering and Technology*, pp.8-15, 2012.

[63] Echeverría, Pedro and López-Vallejo, Marisa. "High performance FPGA-oriented mersenne twister uniform random number generator". *Journal of Signal Processing Systems*. vol. 71. pp. 105-109. 2013.

[64] Elwakil, Ahmed S and Kennedy, Michael Peter. "Construction of classes of circuit-independent chaotic oscillators using passive-only nonlinear devices". *IEEE Transactions*

on Circuits and Systems I : Fundamental Theory and Applications. vol. 48. pp. 289-307. 2001.

[65] Elwakil, Ahmed S and Kennedy, Michael Peter, "Chaotic oscillator configuration using a frequency dependent negative resistor", *International journal of circuit theory and applications*. vol.28, pp.69-76, 2000.

[66] , RL Devaney, "An Introduction to Chaotic Dynamical Systems", *CRC press*, vol.43, pp.152-153, 1989

[67] , ROSLAN, Ummu'Atiqah Mohd, SALLEH, Zabidin, and KILIÇMAN, "A. Solving Zhou chaotic system using fourth-order Runge-Kutta method". *World Applied Sciences Journal*, vol.21, pp.939-944, 2013.

[68] HESTHAVEN, Jan S, "On the analysis and construction of perfectly matched layers for the linearized Euler equations", *Journal of computational Physics*, vol.142, pp.129-147, 1998.

[69] Hammer, Preston C, "The midpoint method of numerical integration", *Mathematics Magazine*. vol.31, pp.193-195, 1958.

[70] , Mansingka, Abhinav S and Barakat, Mohamed L and Zidan, M Affan and Radwan, Ahmed G and Salama, Khaled N. "Fibonacci-based hardware post-processing for non-autonomous signum hyperchaotic system". *International Conference on IT Convergence and Security*, pp.1-4, 2013.

[71] Yu, Fei and Li, Lixiang and He, Binyong and Liu, Li and Qian, Shuai and Zhang, Zinan and Shen, Hui and Cai, Shuo and Li, Yi, "Pseudorandom number generator based on a 5D hyperchaotic four-wing memristive system and its FPGA implementation", *The European Physical Journal Special Topics*, vol.230, pp.1763-1772, 2021.

[72] , May, Robert M, "Simple mathematical models with very complicated dynamics", *Nature*, vol.261, pp.459-467, 1976.

[73] , Hénon, Michel, "A two-dimensional mapping with a strange attractor", *The theory of chaotic attractors*, pp.94-102, 2004.

[74] Dabal, Pawel and Pelka, Ryszard, "A chaos-based pseudo-random bit generator implemented in FPGA device", *IEEE International Symposium on Design and Diagnostics of Electronic Circuits and Systems*, pp.151-154, 2011.

[75] Padgett, Wayne T and Anderson, David V, "Fixed-point signal processing", *Synthesis Lectures on Signal Processing*, vol.4. pp.1-133, 2009.

[76] , Giard, Pascal and Kaddoum, Georges and Gagnon, François and Thibeault, Claude. "FPGA implementation and evaluation of discrete-time chaotic generators circuits", *IECON 2012-38th Annual Conference on IEEE Industrial Electronics Society*. pp.3221-3224, 2012.

[77] Geisel, T and Fairen, V. "Statistical properties of chaos in Chebyshev maps". *Physics Letters A*, vol.105, pp.263-266, 1984.

[78] Souza, Carlos EC and Moreno, Davi and Chaves, Daniel PB and Pimentel, Cecilio. "Pseudo-Chaotic Sequences Generated by the Discrete Arnold's Map Over $\mathbf{Z}_{2^m}$: Period Analysis and FPGA Implementation". *IEEE Transactions on Instrumentation and Measurement*. vol. 71, pp. 1-10, 2022.

[79] , Souza, Carlos EC and Chaves, Daniel PB and Pimentel, Cecilio. "One-Dimensional Pseudo-Chaotic Sequences Based on the Discrete Arnold's Cat Map Over $\mathbf{Z}_{3^m}$". *IEEE Transactions on Circuits and Systems II : Express Briefs*, vol. 68, pp. 491-495, 2020.

[80] /71 Robert, F. "Discrete Iterations, A Metric Study". Translated by Rokne J. *Springer-Verlag, Berlin*, vol.6, 1986

[81] , Mohammed, Bakiri and Couchot, Jean-Francois and Guyeux, Christophe. "FPGA Implementation of F2-Linear Pseudorandom Number Generators based on Zynq MPSoC : A Chaotic Iterations Post Processing Case Study", *International Conference on Security and Cryptography*, vol. 2, pp. 302-309, 2016.

[82] , Association for Computing Machinery. ACM transactions on embedded computing systems : TECS. *ACM Press*. 2002.

[83] Rajagopalan, Vidya and Boppana, Vamsi and Dutta, Sandeep and Taylor, Brad and Wittig, Ralph, "Xilinx Zynq-7000 EPP : An extensible processing platform family". *2011 IEEE Hot Chips 23 Symposium (HCS)*, pp. 1-24, 2011.

[84] Wolf, Alan and Swift, Jack B and Swinney, Harry L and Vastano, John A, "Determining Lyapunov exponents from a time series", *Physica D : nonlinear phenomena*. vol.16, pp.285-317, 1985.

[85] Hua, Zhongyun and Chen, Yongyong and Bao, Han and Zhou, Yicong, "Two-dimensional parametric polynomial chaotic system", *IEEE Transactions on Systems, Man, and Cybernetics : Systems*. vol.25, pp.4402-4414, 2021.

[86] , Barash, L and Shchur, LN, "Periodic orbits of the ensemble of Sinai-Arnold cat maps and pseudorandom number generation". *Physical Review E*, vol.73, pp.1-15. 2006

[87] , Avaroğlu, Erdinc. "Pseudorandom number generator based on Arnold cat map and statistical analysis". *Turkish Journal of Electrical Engineering and Computer Sciences*, vol.25, pp.633-643, 2017.

[88] Chen, Fei and Wong, Kwok Wo and Liao, Xiaofeng and Xiang, Tao, "Period distribution of the generalized discrete arnold cat map for $N = 2^e$", *IEEE Transactions on Information Theory*, vol.59, pp.3249-3255, 2013.

[89] J P Keating and F Mezzadri. "Pseudo-symmetries of Anosov maps and spectral statistics", *Nonlinearity*, vol.13, pp.747-775, 2000.

[90] Wu, Yue and Hua, Zhongyun and Zhou, Yicong. "n-dimensional discrete cat map generation using Laplace Expansions". *IEEE transactions on cybernetics*, vol.46, pp.2622-2633, 2015.

[91] Tong, Xiaojun and Cui, Minggen. "Image encryption with compound chaotic sequence cipher shifting dynamically". *Image and Vision Computing*, vol.26, pp.843-850, 2008.

[92] Ford, Joseph and Mantica, Giorgio and Ristow, Gerald H. "The Arnol'd cat : Failure of the correspondence principle". *Physica D : Nonlinear Phenomena*, vol.50, pp.493-520, 1991.

[93] Chen, Fei and Wong, Kwok-Wo and Liao, Xiaofeng and Xiang, Tao, "Period Distribution of Generalized Discrete Arnold Cat Map for $N = p^e$", *IEEE Transactions on Information Theory*, vol.58, pp.445-452, 2012.

[94] , Dyson, Freeman J and Falk, Harold, "Period of a discrete cat mapping", *The American Mathematical Monthly*, vol.99, pp.603-614, 1992.

[95] Kaneko, Kunihiko, "Coupled map lattice", *Chaos, Order, and Patterns*, pp.237-247, 1993.

[96] Grassi, Giuseppe and Mascolo, Saverio. "A systematic procedure for synchronizing hyperchaos via observer design". *Journal of Circuits, Systems, and Computers*. vol.11, pp.1-16, 2002.

[97] . Nance, Joe, "Periods of the discretized Arnold Cat map and its extension to n dimensions", *arXiv preprint arXiv*, pp.1111-2984, 2011.

[98] Just, Wolfram. "Bifurcations in globally coupled map lattice". *Journal of statistical physics*. vol.79, pp. 429-449, 1995.

[99] Kwok, HS and Tang, KS. "A fast image encryption scheme based on high-dimensional chaotic map". *Chaos, Solitons and Fractals*, vol. 32, pp. 1518-1529, 2007.

[100] Falcioni, Massimo and Palatella, Luigi and Pigolotti, Simone and Vulpiani, Angelo. "Properties making a chaotic system a good pseudo random number generator". *Physical Review E*, vol. 72. pp.1-10. 2005.

[101] , Li, Shu-Jun. "Analyses and new designs of digital chaotic cipher", *Xi'an Jiaotong University*, pp. 1-207. 2003.

[102] Martínez-Ñonthe, JA and Castañeda-Solís, A and Díaz-Méndez, A and Cruz-Irisson, Miguel and Vazquez-Medina, Ruben, "Chaotic block cryptosystem using high precision approaches to tent map", *Microelectronic engineering*, vol.90, pp.168-172, 2012.

[103] Wheeler, Daniel D and Matthews, Robert AJ. "Supercomputer investigations of a chaotic encryption algorithm", *Cryptologia*, vol.15, pp.140-152, 1991.

[104] , Hua, Zhongyun and Zhou, Yicong and Bao, Bocheng. "Two-dimensional sine chaotification system with hardware implementation". *IEEE Transactions on Industrial Informatics*, vol.16, pp.887-897, 2019.

[105] Alawida, Moatsum and Samsudin, Azman and Teh, Je Sen and Alkhawaldeh, Rami S, "A new hybrid digital chaotic system with applications in image encryption", *Signal Processing*, vol.160, pp. 45-58, 2019.

[106] Hua, Zhongyun and Zhou, Binghang and Zhou, Yicong. "Sine-transform-based chaotic system with FPGA implementation". *IEEE Transactions on Industrial Electronics*. vol.65. pp, 2557-2566, 2017.

[107] Hua, Zhongyun and Zhou, Binghang and Zhou, Yicong. "Sine chaotification model for enhancing chaos and its hardware implementation". *IEEE Transactions on Industrial Electronics*, vol.66, pp. 1273-1284, 2018.

[108] Fouda, J S Armand Eyebe and Koepf, Wolfram, "An 8-bit precision cipher for fast image encryption", *Multimedia Tools and Applications*, vol.1, pp.1-20, 2022.

[109] Yu, Fei and Li, Lixiang and He, Binyong and Liu, Li and Qian, Shuai and Huang, Yuanyuan and Cai, Shuo and Song, Yun and Tang, Qiang and Wan, Qiuzhen and others, "Design and FPGA implementation of a pseudorandom number generator based on a four-wing memristive hyperchaotic system and Bernoulli map". *IEEE Access*, vol.7, pp.181884-181898, 2019.

[110] Yu, Fei and Li, Lixiang and He, Binyong and Liu, Li and Qian, Shuai and Huang, Yuanyuan and Cai, Shuo and Song, Yun and Tang, Qiang and Wan, Qiuzhen and others, "Hardware

architecture of a digital piecewise linear chaotic map with perturbation for pseudorandom number generation", *AEU-International Journal of Electronics and Communications*, vol.147, pp.1-10. 2022.

[111] Yu, Fei and Zhang, Zinan and Shen, Hui and Huang, Yuanyuan and Cai, Shuo and Jin, Jie and Du, Sichun. "Design and FPGA implementation of a pseudo-random number generator based on a Hopfield neural network under electromagnetic radiation". *Frontiers in Physics*, vol.9, pp.1-15, 2021.

[112] D. M. Maranhao and C. P. Prado. "Evolution of chaos in the Matsumoto-Chua circuit : a symbolic dynamics approach", *Brazilian Journal of Physics*, vol.35 ; pp.1678-4448, 2013.

LISTE DES PUBLICATIONS

Cette thèse a donné lieu à la soumission et/ou à la publication des articles suivants.

— **Hermann Djeugoue**, Gildas Gnyamsi Gaetan, Fouda Jean Sire Armand Eyebe, Koepf Wolfram ; **"On the implementation of large period piece-wise linear Arnold cat map"** ; *Multimedia Tools and Applications*, vol.1, pp.1-18, 2022

Multimedia Tools and Applications
https://doi.org/10.1007/s11042-022-13175-6

On the implementation of large period piece-wise linear Arnold cat map

Djeugoue Hermann[1] · Gnyamsi Gaetan Gildas[1] · Jean Sire Armand Eyebe Fouda[1,2] · Wolfram Koepf[2]

Received: 27 April 2021 / Revised: 17 February 2022 / Accepted: 11 April 2022

Abstract
This paper presents a piece-wise linear cat map (PWLCM) obtained by perturbing the conventional quantized Arnold cat map (QACM) with a nonlinear term. The effect of the nonlinear term on the dynamics of the QACM is investigated. We show that the eigenvalues, hence the Lyapunov exponents of the PWLCM depend on the initial conditions, which is not the case for the QACM. As a result, the proposed PWLCM is a generalized form of the QACM, whose the period exponentially increases with respect to the precision, thus taking as value 1.09×10^{513} for only 10-bit precision; while that of the corresponding QACM is only 768. The nonlinear term increases the sensitivity of the system to the initial conditions, which contributes to increase its period, hence to enhance its complexity. An electronic implementation of both the QACM and the PWLCM in the case of 4-bit precision using Multisim is presented. The proposed architecture of both the QACM and the PWLCM are implemented using Verilog and prototyped on the Zynq 7020 FPGA board. For 4-bit precision, the FPGA implementation performs 1.072 Gbps throughput at 134 MHz maximum frequency. We verified that experimental and simulation behaviors of the proposed system perfectly match, thus confirming the effectiveness of the proposed electronic circuit for exhibiting the expected dynamics in real-time.

Keywords Dynamical system · Random number · Circuit theory · Digital circuits

1 Introduction

Random number generators are used in cryptography, in games of chance, in all computer algebra systems or other programming languages and in numerical simulations, to cite a

✉ Wolfram Koepf
koepf@mathematik.uni-kassel.de

Jean Sire Armand Eyebe Fouda
efoudajsa@yahoo.fr

[1] Départment de Physique, Université de Yaoundé I, Yaoundé, Cameroun

[2] Institute of Mathematics, University of Kassel, Kassel, Germany

few. In order to achieve fast generation of random numbers, researchers have focused their attention to chaotic systems that can be experimentally implemented [5, 15, 21]. Chaotic systems are sensitive to initial conditions and present mixing properties that are necessary for a good pseudo-random number generator [12]. Since then, various chaotic systems generating a large variety of complex dynamics have been modeled, but they are still suffering from lack of reliable methods for their implementation [3, 24–26, 32]. Some of them are realized with analogue circuits, while many others are more and more implemented using numerical targets like the Field Programmable Gate Array (FPGA) [2, 16, 19, 26–28].

Chaotic systems have been shown efficient and convenient for modern cryptography, although most experimental and commercial systems are using random number generators (RNGs) based on modular algebra operations, as the digitization of chaotic orbits sometimes may reduce the key sensitivity [20, 26]. Such RNGs present the advantage to make modular calculations in a finite field (the Galois field GF(2^n) for example), hence to be easily implemented on hardware with a finite number of basic electronic logic gates without loss of precision. An example is the RNG implemented by the $RAND$ function in the MATLAB software to generate uniformly distributed random numbers. Another example is the Arnold cat map (ACM) which is often used for its ergodic and mixing properties [1, 30]. In practical applications, the original cat map is generalized and discretized in the phase space to obtain the quantized ACM (QACM).

Arnold's cat map is known to be chaotic, area-preserving, ergodic and mixing, and invertible [8, 10, 17]. It has a unique hyperbolic fixed point and the linear transformation defining it is hyperbolic (it presents irrational eigenvalues, one with an absolute value greater than 1 and the other one less than 1). Its quantized version forms short limit cycles whose lengths depend on the modulo value, although it preserves the properties of its continuous analogue [4, 11]. The other properties of this interesting map can be found in the literature [4, 8, 10, 11, 17]. The ACM is used in cryptography, in digital tattoo applications, in watermarking and for random number generation to cite a few [8, 9, 23]. The QACM is particularly used for image scrambling due to its periodic nature [7]. It has been also used for the implementation of public key ciphers [18], but the latter are not secure when dealing with QACM with weak periods [22].

Thus, the period of the QACM is an important parameter when using it in cryptography. It has been shown that the period of the QACM does not exceed $3m$, m being the modulo value. In the case of n-bit precision ($m = 2^n$) which is convenient for digital applications, the period of the QACM is only equal to $3{\cdot}2^{n-2}$, $n \geq 2$, which is effectively smaller than $3m$. Therefore, for the security level of ciphers including the QACM to be enhanced, its period needs to be increased. In order to overcome such a limitation of the period, the ACM ($n = 0$) is preferred to the QACM in many applications. Some improvements including the increase of the dimension of the map have been proposed. Guoscheng et al. [14] proposed to extend the conventional 2D Arnold's cat map into a 3D map by introducing six control parameters. Although the resulting map allows to increase the key space for data encryption, the authors did not investigate its period distribution, as well as the impact of the introduced control parameters on dynamics of the system. To prevent the degradation of chaotic sequences into periodic ones due to the finite computer precision, they just applied a slight perturbation on the 3D map output without formally investigating its impact. In [33], the authors combined the 2D cat map with an affine cipher to enhance the security level of their proposed cipher. In order to efficiently apply Arnold's cat map to data encryption, it is important to directly investigate and increase its period in the discrete phase space, hence to significantly increase the period of the QACM, even for small data precision.

In this paper, we suggest to introduce a nonlinear perturbation to the linear QACM for its period to increase. In order to preserve the simplicity of the QACM, we consider as nonlinear element another modular algebra based module, such that the final system is a piece-wise linear cat map (PWLCM). We thus investigate the dynamics of the so-called PWLCM and verify that in the case of 4-bit precision, its period is more than 10^{11} times greater than that of the conventional QACM. An equivalent electronic circuit is proposed in the case of 4-bit precision to further confirm the simplicity of the proposed system that can be used for the generation of pseudo-random numbers. Furthermore, an equivalent implementation on Zynq 7020 FPGA board is presented in order to confirm the effectiveness of the proposed architecture.

The rest of the paper is organized as follows: in Section 2 the modeling system is presented, Section 3 is devoted to the results analysis, Section 4 shows the electronic implementation of the PWLCM while in Section 5 some concluding remarks are given.

2 The modeling system

2.1 The conventional quantized Arnold cat map

The Arnold cat map has been widely described and investigated in the literature. The evolution of the 2D system behavior, where the two variables (position and momentum) are completely depending on each other is modeled as follows [17]:

$$\begin{cases} x(t+1) = x(t) + \alpha y(t) \\ y(t+1) = \beta x(t+1) + y(t) \end{cases} \mod m. \tag{1}$$

The system in (1) can be rewritten using matrix representation as

$$\mathbf{x}(t+1) = A\mathbf{x}(t) \mod m \tag{2}$$

where

$$A = \begin{pmatrix} 1 & \alpha \\ \beta & \alpha \cdot \beta + 1 \end{pmatrix},$$

$(\alpha, \beta) \in \mathbb{N}_{\geq 1}^2$, and $\mathbf{x} = (x, y)^T$; $(\cdot)^T$ is the transpose of $(\cdot)$. This discrete time system is continuous in the phase space for $(x, y) \in [0, 1)^2$ and $m = 1$. The QACM is obtained for $(x, y) \in [0, m)^2$ with $m \in \mathbb{N}_{>1}$. It is periodic and its period depends both on m and the parity of α and β. For $\alpha = \beta = 1$ and $m = 2^n$, the period behaves like

$$\Pi_n = 2 \cdot \Pi_{n-1},\ n > 2 \tag{3}$$

with $\Pi_1 = \Pi_2 = 3$ for the minimal period, as shown in [8, 11].

2.2 The piece-wise linear cat map

In order to increase the period of the QACM, we introduce a nonlinear perturbation term to the conventional QACM. The modified system is written as

$$\mathbf{x}(t+1) = A\mathbf{x}(t) + \mathbf{x}_c(t) \mod m, \tag{4}$$

where

$$\mathbf{x}_c(t) = \begin{pmatrix} \sum_{i=1}^{M} \Big(a_i + y(t)\Big) \bmod c_i \\ \sum_{j=1}^{N} \Big(b_j + x(t+1)\Big) \bmod d_j \end{pmatrix} \tag{5}$$

with $(i, j) \in \mathbb{N}^2$, c_i and d_j two natural numbers such that $0 \leq c_i < m + a_i$ and $0 \leq d_j < m + b_j$, $0 \leq a_i, b_j < m$ if $(c_i, d_j) = (0, 0)$; $0 \leq a_i < c_i$ if $c_i \neq 0$ and, $0 \leq b_j < d_j$ if $d_j \neq 0$. Let us consider $\mathbf{x}_c(t) = \Big(x_c(t), y_c(t)\Big)^T$ with

$$x_c(t) = \sum_{i=1}^{M} x_{c,i}(t), \tag{6}$$

$$y_c(t) = \sum_{j=1}^{N} y_{c,j}(t). \tag{7}$$

and

$$x_{c,i}(t) = \Big(a_i + y(t)\Big) \bmod c_i, \tag{8}$$

$$y_{c,j}(t) = \Big(b_j + x(t+1)\Big) \bmod d_j, \tag{9}$$

Equations (8)-(9) can be written as piece-wise linear functions such that:

$$x_{c,i}(t) = \begin{cases} a_i + y(t), & \text{if } a_i + y(t) < c_i; \\ a_i + y(t) - q_i c_i, & \text{otherwise,} \end{cases} \tag{10}$$

where $q_i = \lfloor \frac{a_i + y(t)}{c_i} \rfloor$; and

$$y_{c,j}(t) = \begin{cases} b_j + x(t+1), & \text{if } b_j + x(t+1) < d_j; \\ b_j + x(t+1) - q_j d_j, & \text{otherwise,} \end{cases} \tag{11}$$

$q_j = \lfloor \frac{b_j + x(t+1)}{d_j} \rfloor$. By developing the whole set of equations, we obtain the PWLCM defined as:

$$\mathbf{x}(t+1) = B\mathbf{x}(t) + C(t) \mod m \tag{12}$$

where

$$B = \begin{pmatrix} 1 & \alpha' \\ \beta' & \alpha'\beta' + 1 \end{pmatrix}, \tag{13}$$

with $\alpha' = M + \alpha$, $\beta' = N + \beta$ and,

$$C(t) = \begin{pmatrix} \sum_{i=1}^{M} \Big(a_i - q_i c_i \cdot u\Big(a_i + y(t) - c_i\Big)\Big) \\ \beta' \sum_{i=1}^{M} \Big(a_i - q_i c_i \cdot u\Big(a_i + y(t) - c_i\Big)\Big) + \sum_{j=1}^{N} \Big(b_j - q_j d_j \cdot u\Big(b_j + x(t+1) - d_j\Big)\Big) \end{pmatrix}. \tag{14}$$

$u(t)$ is the Heaviside function defined as

$$u(t) = \begin{cases} 0, & \text{if } t < 0; \\ 1, & \text{otherwise.} \end{cases} \tag{15}$$

The PWLCM thus obtained presents a conservative linear term $B\mathbf{x}(\mathbf{t})$ ($\det(B) = 1$) that exhibits the same behavior as a QACM, and a nonlinear term $C(t)$ that contributes to

increase the period of the linear term by modifying its trajectory for a given initial condition. a_i and b_j, c_i and d_j are defined as perturbation parameters.

2.3 Stability analysis of the PWLCM

In this section, we investigate the stability of the PWLCM. While considering the system in (12), we deduce the following Jacobian matrix:

$$J = \begin{pmatrix} 1 & \alpha' - \sum_{i=1}^{M} c_i \delta\left(y(t) - \tau_y^i\right) \\ \beta' - \sum_{j=1}^{N} d_j \delta\left(x(t+1) - \tau_x^j\right) & 1 + \left(\alpha' - \sum_{i=1}^{M} c_i \delta\left(y(t) - \tau_y^i\right)\right)\left(\beta' - \sum_{j=1}^{N} d_j \delta\left(x(t+1) - \tau_x^j\right)\right) \end{pmatrix} \tag{16}$$

where $\tau_x^j = d_j - b_j$ and $\tau_y^i = c_i - a_i$. Indeed, $q_i = 1$ when $\delta\left(y(t) - \tau_y^i\right) = 1$ and $q_j = 1$ when $\delta\left(x(t+1) - \tau_x^j\right) = 1$. From the above Jacobian matrix we deduce the eigenvalues

$$\Lambda_1(t) = 1 + \frac{1}{2}\left(\alpha' - \sum_{i=1}^{M} c_i \delta\left(y(t) - \tau_y^i\right)\right)\left(\beta' - \sum_{j=1}^{N} d_j \delta\left(x(t+1) - \tau_x^j\right)\right) \left(1 + \sqrt{1 + \frac{4}{\left(\alpha' - \sum_{i=1}^{M} c_i \delta\left(y(t) - \tau_y^i\right)\right)\left(\beta' - \sum_{j=1}^{N} d_j \delta\left(x(t+1) - \tau_x^j\right)\right)}}\right) \tag{17}$$

and

$$\Lambda_2(t) = 1 + \frac{1}{2}\left(\alpha' - \sum_{i=1}^{M} c_i \delta\left(y(t) - \tau_y^i\right)\right)\left(\beta' - \sum_{j=1}^{N} d_j \delta\left(x(t+1) - \tau_x^j\right)\right) \left(1 - \sqrt{1 + \frac{4}{\left(\alpha' - \sum_{i=1}^{M} c_i \delta\left(y(t) - \tau_y^i\right)\right)\left(\beta' - \sum_{j=1}^{N} d_j \delta\left(x(t+1) - \tau_x^j\right)\right)}}\right) \tag{18}$$

The determinant of J is equal to 1, which implies that the sum of the Lyapunov exponents $\lambda_1(t)$ and $\lambda_2(t)$ is equal to 0, hence $\Lambda_1(t) \geq 1$ and $0 \leq \Lambda_2(t) \leq 1$. The PWLCM is thus a conservative system as the generating ACM, independently to the choice of the parameters a_i, b_i, c_i and d_i. Depending on the choice of these parameters, it can be difficult to determine the steady states of the system, whenever they exist. For $a_i = b_j = 0, \forall i,\ j, (x = 0, y = 0)$ is the single steady state of the system. Given that $\Lambda_{1,2} > 0$, all the existing steady states of the PWLCM are unstable.

2.4 The reverse PWLCM

The generalized inverse PWLCM is obtained by determining $x(t)$ and $y(t)$ from (4) as:

$$\begin{cases} x(t) = x(t+1) - y(t) - \sum_{i=1}^{M}(y(t) + a_i) \bmod c_i \\ y(t) = x(t+1) - y(t+1) - \sum_{j=1}^{N}(x(t+1) + b_j) \bmod d_j. \end{cases} \bmod 2^n \tag{19}$$

Given that $x(t+1)$ and y(t+1) are the initial condition for the reverse system, $x(t)$ and $y(t)$ are the iterates that are obtained from the initial conditions. By reversing the time evolution, (19) can be rewritten as:

$$\begin{cases} y(t+1) = x(t) - y(t) - \sum_{j=1}^{N}(x(t)+b_j) \bmod d_j \\ x(t+1) = x(t) - y(t+1) - \sum_{i=1}^{M}(y(t+1)+a_i) \bmod c_i. \end{cases} \bmod 2^n \qquad (20)$$

Equation (20) is the reverse PWLCM.

3 Results and discussion

In this section, we investigate the dynamics of the PWLCM period and largest Lyapunov exponent with respect to the system control parameters and initial conditions. For a given precision, the period of the PWLCM is equivalent to the least common multiple (lcm) of the individual periods of the set of initial conditions [6, 13, 27, 29, 31]. Indeed, each possible initial condition of the phase space generates its own dynamics whose period is determined. The number of initial conditions is directly related to the precision n. For $n = 1$ for example, there are four possible initial conditions that are $(0, 0),\ (0, 1),\ (1, 0)$ and $(1, 1)$.

3.1 Sensitivity to initial conditions

3.1.1 Sensitivity of the period

We evaluate the period of the system for various precisions $n = 2$ to $n = 8$. Table 1 illustrates the matrix $T(x_0, y_0)$ of individual periods for the different initial conditions of the PWLCM in the case of $n = 2$. We set for this example $\alpha = \beta = 1$, $M = N = 2$, $c_1 = 0$, $c_2 = 3$, $d_1 = 3$, $d_2 = 5$, $a_1 = 1$, $a_2 = 1$, $b_1 = 0$ and $b_2 = 2$. The period of the PWLCM in that case is equal to $\Pi = \text{lcm}(\{T(x_0, y_0)\}) = 105$, $0 \leq x_0, y_0 \leq 2^n - 1$; while the corresponding period for the QACM is 3. In order to estimate the impact of the nonlinear term on the QACM, we evaluated the period of the QACM described by matrix B, instead of matrix A. We obtained as period $\Pi_B = 3$ while $\Pi_A = 3$. It appears that by adding the modulus terms, the system described by T_A is both linearly and nonlinearly modified, thus leading to a new QACM that is perturbed by the nonlinear term $C(t)$. The linear modification leads to the conventional QACM, whereas the nonlinear modification leads to a completely different system with a large period that does not necessarily respect the relation in (3).

Table 1 also shows that the system above described presents a single steady state that is $(x_0 = 2,\ y_0 = 3)$. While evaluating the impact of the perturbation on the QACM, we observed that for some combinations of parameters a_i, b_i c_i and d_i, the PWLCM may present or not multiple steady states. For $\alpha = \beta = 1$, $M = N = 2$, $c_1 = c_2 = 0$,

Table 1 Matrix of the individual period of different initial conditions of the PWLCM for $n = 2, \alpha = \beta = 1, M = N = 2, c_1 = 0, c_2 = 3, d_1 = 3, d_2 = 5, a_1 = 1, a_2 = 1, b_1 = 0$ and $b_2 = 2$

$x_0 \backslash y_0$	0	1	2	3
0	7	5	7	5
1	7	3	5	7
2	7	3	5	1
3	3	5	7	7

Table 2 Matrix of the individual largest Lyapunov exponent of different initial conditions of the PWLCM for $n = 2, \alpha = \beta = 1$, $M = N = 2, c_1 = 0, c_2 = 3$, $d_1 = 3, d_2 = 5, a_1 = 1, a_2 = 1$, $b_1 = 0$ and $b_2 = 2$; $\lambda_0(3, 3) = 2.3895$

$x_0 \backslash y_0$	0	1	2	3
0	1.6777	1.2718	1.6778	1.2719
1	1.6778	2.4554	1.2721	1.6778
2	1.6778	2.4554	1.2720	2.3895
3	2.4554	1.2719	1.6777	1.6777

$d_1 = d_2 = 0, a_1 = 0, a_2 = 0$ and $b_1 = b_2 = 0$ for example, there is a single steady state, that is $(x_0 = 0,\ y_0 = 1)$ while the period of the system is $\Pi = 3$. By just changing the value of c_2 as $c_2 = 3$, the system now presents two steady states that are $(x_0 = 0, y_0 = 1)$ and $(x_0 = 0, y_0 = 2)$ and a period $\Pi = 20$.

3.1.2 Sensitivity of the Lyapunov exponent

We also evaluate the Lyapunov exponents of the above system for each initial condition with the above three parameter settings and compare them to the largest Lyapunov exponent of the corresponding QACM, that is

$$\lambda_0(\alpha', \beta') = \log\left(1 + \frac{1}{2}\alpha'\beta' + \frac{1}{2}\sqrt{\alpha'^2\beta'^2 + 4\alpha'\beta'}\right) \tag{21}$$

We considered 20 000 iterations of the PWLCM to evaluate the Lyapunov exponents.

For the first parameter setting, Table 2 shows the corresponding largest Lyapunov exponents of the PWLCM that are to be compared to $\lambda_0(3, 3) = 2.3895$. It appears from this table that setting $c_i \neq 0$ and $d_j \neq 0$ contributes to reduce the Lyapunov exponent, while it contributes to increase the period of the system.

The second parameter setting leads to Table 3, with values that are to be compared to $\lambda_0(3, 3) = 2.3895$. We can confirm that the Lyapunov exponent does not depend on a_i and b_i in the case $c_i = d_j = 0$.

The third parameter setting leads to Table 4, with values that are also to be compared to $\lambda_0(3, 3) = 2.3895$. This table as compared to Table 3 shows that setting $c_i \neq 0$ or $d_j \neq 0$ reduces the value of the Lyapunov exponent, while increasing the period of the PWLCM.

The combination of the observations made from these three tables implies that there is no direct relationship between the Lyapunov exponent and the period of the system. Furthermore, a large Lyapunov exponent in that case does not induce a high complexity of the PWLCM, as it corresponds to the smallest period. However, we observe that the period of the system increases with the diversity or variability of the Lyapunov exponent. Indeed, a high sensitivity of the Lyapunov exponent to the initial conditions contributes to increase the period of the system. Figure 1 shows the behavior of the Lyapunov exponent in the case of $n = 4$ for the above three parameter settings. In the first case (setting 1), there are

Table 3 Matrix of the individual largest Lyapunov exponent of different initial conditions of the PWLCM for $n = 2, \alpha = \beta = 1$, $M = N = 2, c_1 = c_2 = 0$, $d_1 = d_2 = 0, a_1 = 0, a_2 = 1$, $b_1 = b_2 = 0$; $\lambda_0(3, 3) = 2.3895$

$x_0 \backslash y_0$	0	1	2	3
0	2.3895	2.3895	2.3895	2.3895
1	2.3895	2.3895	2.3895	2.3895
2	2.3895	2.3895	2.3895	2.3895
3	2.3895	2.3895	2.3895	2.3895

Table 4 Matrix of the individual largest Lyapunov exponent of different initial conditions of the PWLCM for $n = 2, \alpha = \beta = 1$, $M = N = 2, c_1 = 0, c_2 = 3$, $d_1 = d_2 = 0, a_1 = 0, a_2 = 1$, $b_1 = b_2 = 0$; $\lambda_0(3, 3) = 2.3895$

$x_0 \backslash y_0$	0	1	2	3
0	2.3895	2.3895	0.0006	2.3895
1	1.8542	1.8542	1.8543	2.3895
2	1.8542	1.8542	1.8543	1.8542
3	2.3895	1.8542	1.8543	1.8542

187 distinct values of λ, five distinct period values ($T = \{1, 4, 5, 8, 233\}$), and the corresponding period of the PWLCM is $T_1 = 9320$; in the second case, a single value of λ and four distinct period values ($T = \{1, 3, 6, 12\}$) are obtained, thus leading to $T_2 = 12$; while in the third case (setting 3), there are 63 distinct values of λ and 9 distinct periods ($T = \{1, 2, 4, 8, 10, 12, 14, 18, 40\}$), which corresponds to $T_3 = 2520$. We can observe that in the case of a single Lyapunov exponent, the largest period value is a multiple of the other values, which contributes to reduce the period of the whole system.

Such an observation also implies a high sensitivity of the period to the precision n. Indeed, for a parameter setting with a high sensitivity of the Lyapunov exponent to the initial conditions, the diversity of the Lyapunov exponent values increases with n, as the number of initial conditions itself increases. While setting $n = 5$ for the above parameter settings, the periods become respectively $T_1 = 1.51 \cdot 10^{15}$, $T_2 = 24$ and $T_3 = 1.02 \cdot 10^{10}$. As shown in Fig. 2, the values of the largest Lyapunov exponent are within the same range (1.4, 2.5), but the number of distinct values has significantly increased, 450 for the first parameter setting

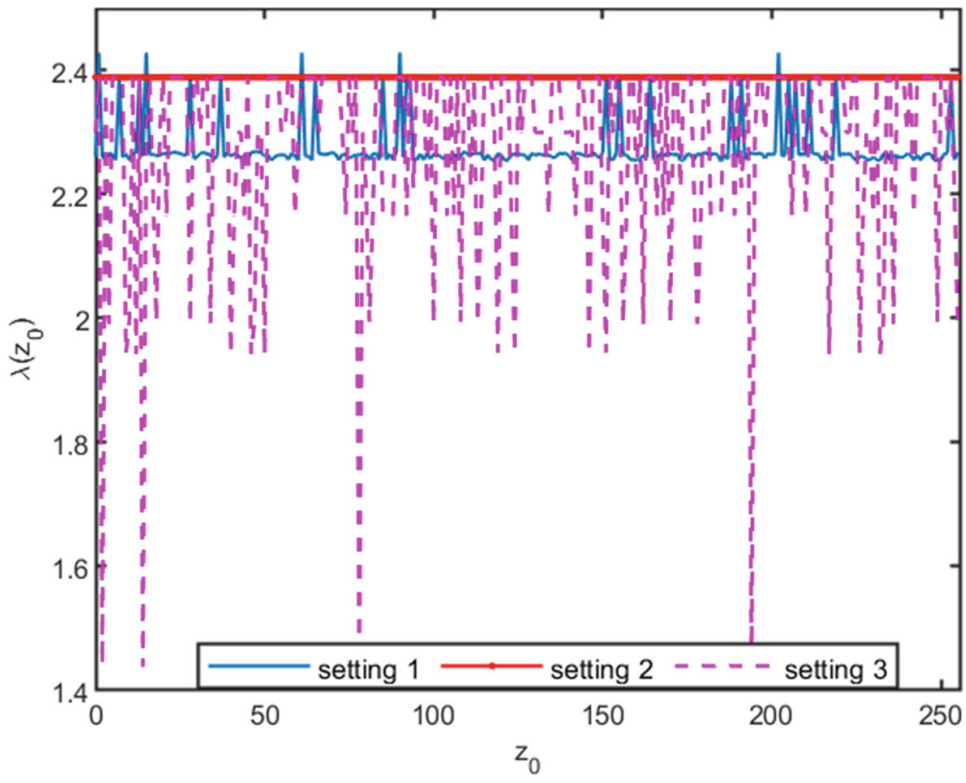


Fig. 1 Behavior of the Lyapunov exponent $\lambda(z_0)$ with respect to the initial condition $z_0 = 2^n x_0 + y_0$, for $n = 4$ and a_i, b_i, c_i and d_i set as in Table 2 (setting 1), Table 3 (setting 2) and Table 4 (setting 3). The corresponding periods are respectively $T_1 = 9320$, $T_2 = 12$ and $T_3 = 2520$.The Lyapunov exponents were evaluated after 1000 iterations of the PWLCM

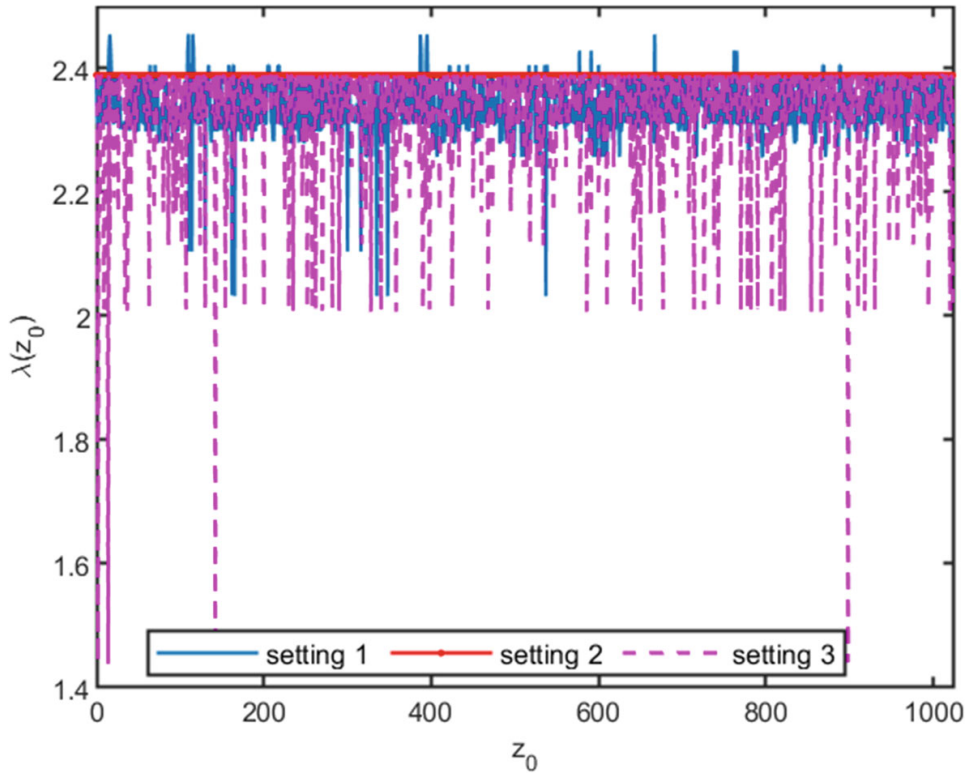


Fig. 2 Behavior of the Lyapunov exponent $\lambda(z_0)$ with respect to the initial condition $z_0 = 2^n x_0 + y_0$, for $n = 5$ and a_i, b_i, c_i and d_i set as in Table 2 (setting 1), Table 3 (setting 2) and Table 4 (setting 3). The corresponding periods are respectively $T_1 = 1.51 \cdot 10^{15}$, $T_2 = 24$ and $T_3 = 1.02 \cdot 10^{10}$. The Lyapunov exponents were evaluated after 1000 iterations of the PWLCM

(setting 1), and 163 for the third one (setting 3). There is no change for the second parameter setting as the corresponding Jacobian matrix does not depend on the initial conditions.

3.2 Sensitivity to control parameters a_i and b_i

For the analysis of the impact of a_i and b_i, we first set $M = N = 1$ and $n = 3$, $\alpha = \beta = 1$, $c_1 = 3$, $d_1 = 5$. The corresponding periods are summarized in Table 5 from where we can appreciate the sensitivity of the system to the parameters a_i and b_j. These periods are to be compared to $\Pi_A = 6$ that is the period of the equivalent QACM.

3.3 Sensitivity on control parameters c_i and d_i

Table 6 shows the dependence of the PWLCM period on the perturbation parameters c_1 and d_1, for $n = 2$, $M = N = 1$, $\alpha = \beta = 1$, $a_1 = 1$ and $b_1 = 3$. It appears from this table that the maximum period is obtained for $(c_1, d_1) = (1, 6)$. This table shows that large periods can be achieved even with a small number of bits ($n = 2$ for example), which is not possible

Table 5 Dependence of the PWLCM period Π_{PWLCM} on a_i and b_i for $n = 3$, $M = N = 1$, $\alpha = \beta = 1$, $c_1 = 3$, $d_1 = 5$ and different values of (a_1, b_1)

$a_1 \backslash b_1$	0	1	2	3	4
0	504	108	60	252	48
1	62	456	72	120	462
2	350	429	252	13566	12558

Table 6 Dependence of the PWLCM Π_{PWLCM} period on c_i and d_i for $n = 2$, $\alpha' = \beta' = 2$, $a_1 = 1$, $b_1 = 3$ and different values of (c_1, d_1)

$c_1 \backslash d_1$	0	1	2	3	4	5	6
0	2	4	4	16	2	6	4
1	4	3	12	70	4	105	12
2	4	12	6	16	4	14	8
3	6	28	14	8	6	36	14
4	2	4	4	16	2	6	4

with the conventional QACM. Indeed, the period depends on the choice of the parameters a_i, b_i, c_i and d_i.

The proposed PWLCM is conservative and depending on the parameter setting, it can be linear or nonlinear. As it includes all the properties of the QACM, it can be seen as a generalized form of the QACM that can exhibit large periods. The worst parameter setting of the PWLCM corresponds to a QACM. For the PWLCM to generate dynamics with large periods, its Jacobian matrix should depend on the initial conditions. We suggest a particular parameter setting depending on the parity of n that generates large periods such that:

$$c_i(\text{resp.}d_i) = \begin{cases} n+2, & \text{if } n = 2p+1; \\ 2^{n+1} - (n+1), & \text{if } n = 2p. \end{cases}, \; p \in \mathbb{N}_{\geq 1}, \tag{22}$$

and

$$a_i(\text{resp.}b_i) = \begin{cases} p, & \text{if } n = 2p+1; \\ 2^{n+1} - (2^n+1), & \text{if } n = 2p. \end{cases}, \; p \in \mathbb{N}_{\geq 1}. \tag{23}$$

We verified that the case $c_i = 0$ or $d_i = 0$ corresponds to the forced QACM in which the steady state (0,0) is modified and depends on a_i and b_i and the case $c_i = 1$ or $d_i = 1$ corresponds to the conventional QACM. The period of the system is large when only one dimension is perturbed with $d_i = 1$ and c_i as in (22), or $c_i = 1$ and d_i as in (22). In such a case, the Lyapunov exponent of the PWLCM is sensitive to the initial condition as it is the case for many chaotic systems. It can therefore generate rich and complex dynamics. An example periodic image shuffling using both 8-bit QACM and 8-bit PWLCM is shown in Fig. 3. The period of the QACM in that case is 192, while that of the PWLCM, with $a_1 = 0$, $a_2 = 0$, $c_1 = 0$, $c_2 = 11$ and $d_1 = d_2 = 1$, is 4.28×10^{114}. Applying the reverse system to the shuffled image with the same number of iterations allows to obtain the original image without needs of running the shuffling process on the whole period of the system.

In order to compare the mixing property of the PWLCM and QACM, we applied the NIST800-22 statistical test to a periodic $2^{13} \times 2^{13}$ image shuffled with both systems. The periodic image is obtained by repeating sequences of 8-bit encoded integers ranged from 0 to 255. Such a data set can be divided into 50 bitstreams of 10^6 length each. The NIST test was applied to the shuffled images, after 50 iterations. The PWLCM parameters were set as $a_1 = 0$, $a_2 = 0$, $c_1 = 0$, $c_2 = 11$ and $d_1 = d_2 = 1$, and the number of bits was $n = 13$ for both systems. The corresponding results are shown in Table 7, from where it appears that the PWLCM shuffled image is passing NIST test, whereas the QACM image is failing. The PWLCM thus better mixes the pixels of the image than the QACM, hence is suitable for image shuffling.

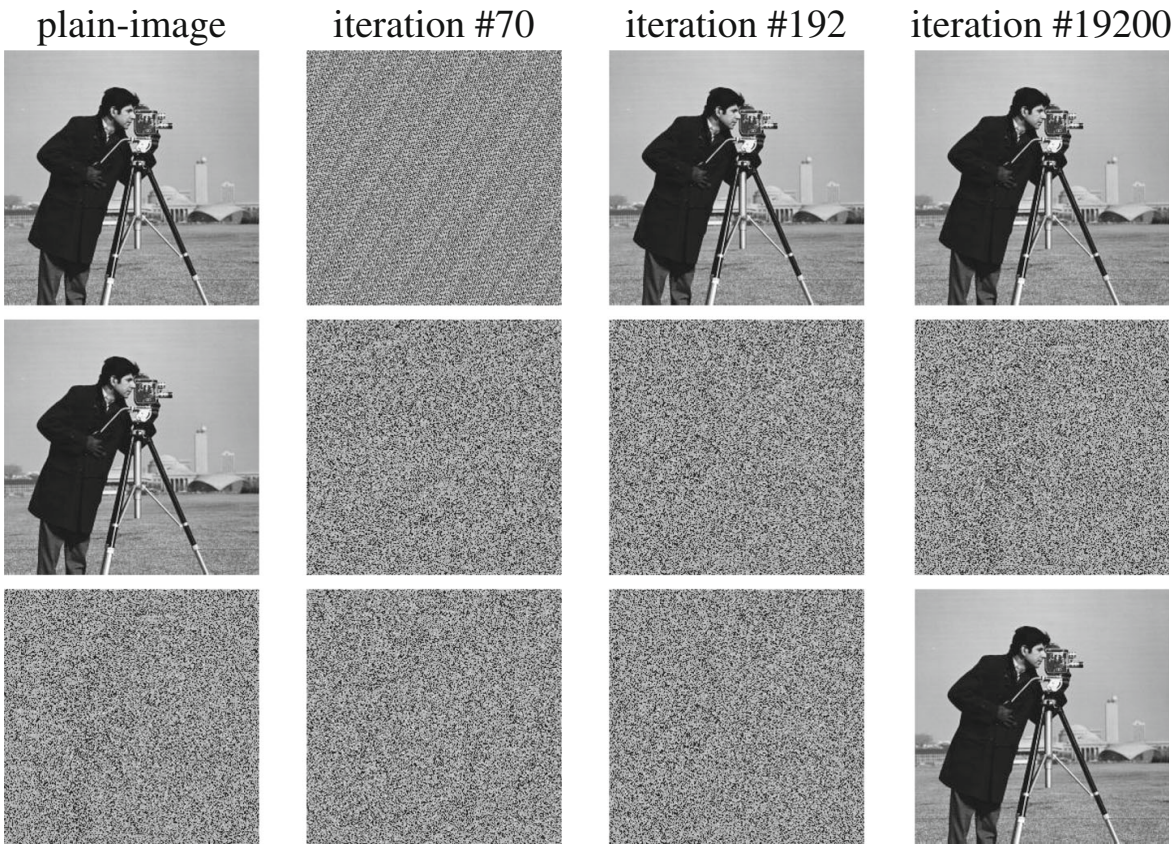


Fig. 3 Image shuffling using the QACM and PWLCM Transformation. The first line shows the results of the QACM, the second line depicts the PWLCM results, while the third line shows the reverse image obtained from the reverse PWLCM

Table 7 NIST 800-22 test results:

Sub-Tests	QACM		PWLCM	
	P-value	Proportion	P-value	Proportion
Frequency	0.0	7/50	0.3505	50/50
Block frequency	0.0	0/50	0.6993	49/50
Cumulative sums (forward)	0.0	0/50	0.3191	50/50
Cumulative sums (reverse)	0.0	0/50	0.1538	50/50
Runs	0.0	5/50	0.6163	49/50
Longest run	0.0	0/50	0.5749	49/50
Rank	0.0	0/50	0.1223	49/50
FFT	0.0	0/50	0.1719	48/50
Non overlapping	0.0	0/50	0.9114	50/50
Overlapping	0.0	0/50	0.6579	49/50
Universal	0.0	0/50	0.3191	50/50
Approximate entropy	0.0	0/50	0.4190	49/50
Random excursions	0.0	0/36	0.3505	36/36
Random excursions variant	0.0	0/36	0.8044	36/36
Serial	0.0	0/50	0.6163	50/50
Linear complexity	0.0	0/50	0.5749	50/50

4 Hardware implementation

In this section, we propose 2-dimensional (2D) electronic implementations of the conventional QACM and the proposed PWLCM. The implementation circuit includes exclusively basic electronic logic circuits such as adders, multiplexers, D-type flip-flops, and basic logic gates (AND, NOR, NAND, NOT, ...). We propose a hardware architecture which is simulated on both Multisim and Vivado HLx, and implemented on a Zynq 7020 FPGA board to confirm the effectiveness of the proposed architecture. The Multisim synthesis allows to optimize the FPGA architecture.

4.1 Multisim architecture

We first designed the circuit corresponding to the conventional 2-dimensional (2D) QACM with $\alpha = \beta = 1$. For such a circuit, we considered two stages, the one computing $x(t+1)$ and the one computing $y(t+1)$. Given that $y(t+1)$ depends on $x(t+1)$, we considered a delay time of half the clock period to sequentially determine $x(t+1)$, then $y(t+1)$. The corresponding electronic architecture is shown in Fig. 4.

Components U1 and U4 are multiplexers (74157N) that are used to set initial conditions x_0 and y_0, respectively. The SET input allows to load initial conditions when $SET = 1$. Once initial conditions have been set, the circuit starts oscillating ($SET = 0$), that is computing $x(t+1)$ on the leading edge of the clock pulse and $y(t+1)$ on the trailing edge of the clock signal. U3 and U6 are D-type flip-flops (74ALS273) that allow the circuit to sequentially change the output value as a clock pulse occurs (time increment). U2 and U5 are 4-bit adders (74283N) that are used to implement (1).

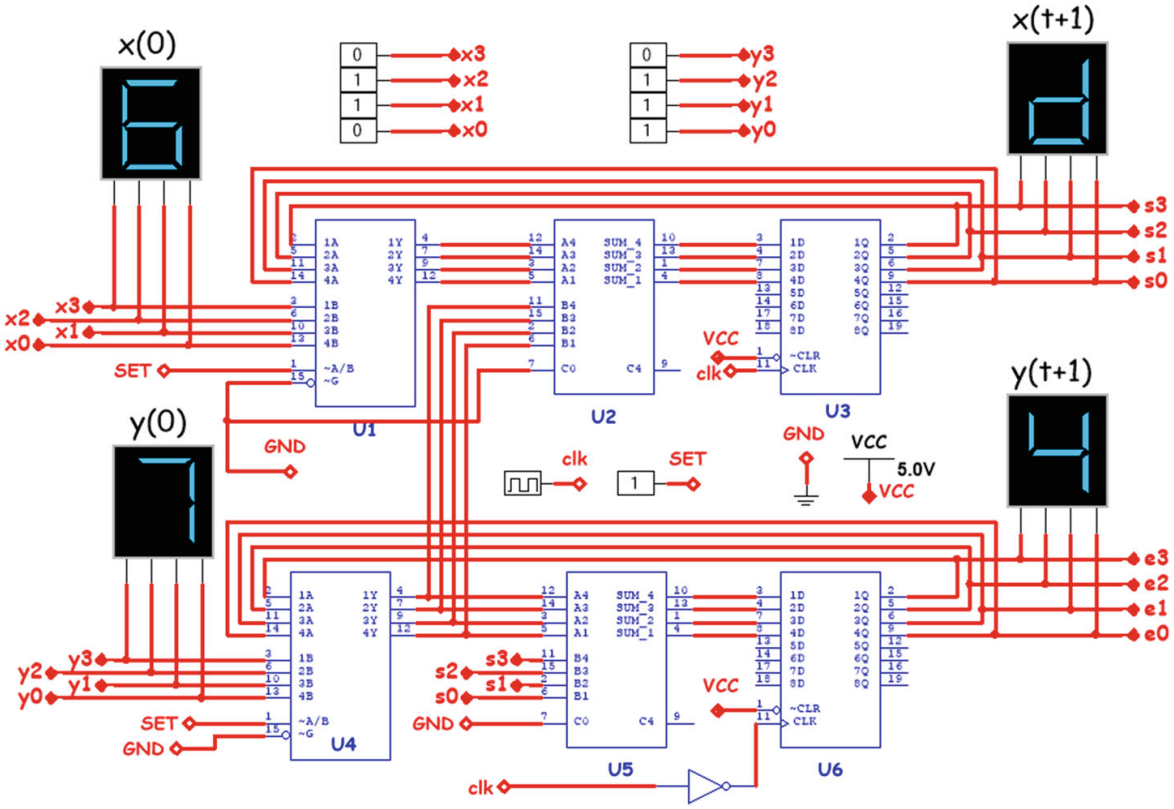


Fig. 4 Circuit of the conventional 2D QACM for $\alpha = \beta = 1$, $n = 4$. Values are displayed in hexadecimal representation

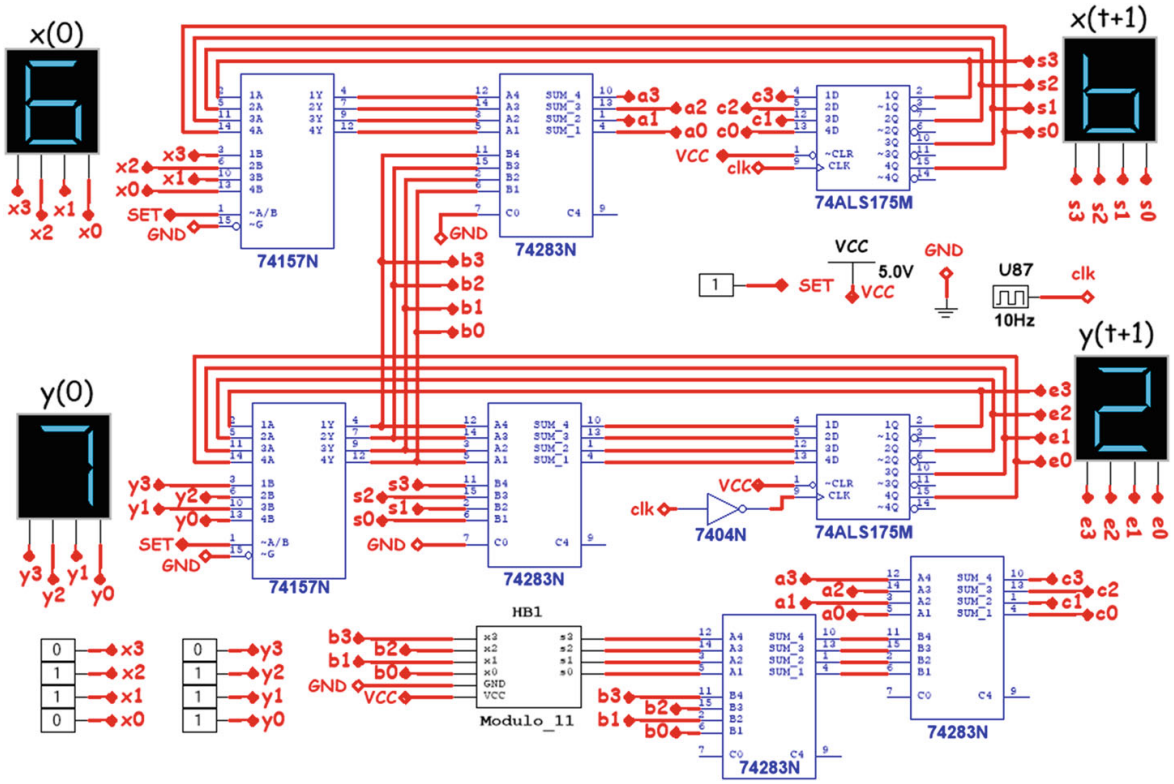


Fig. 5 Electronic implementation of 4-bit PWLCM with $\alpha' = 3$, $\beta' = 1$, $c_1 = 0$, $c_2 = 11$ and, $a_1 = a_2 = 0$. Values are displayed in hexadecimal representation

Now considering the perturbation term, we propose the schematic of the PWLCM whose equation is

$$\begin{cases} x(t+1) = x(t) + y(t) + \Big(a_1 + y(t)\Big) \mod c_1 + \Big(a_2 + y(t)\Big) \mod c_2 \\ y(t+1) = x(t+1) + y(t) \end{cases} \mod 2^n. \tag{24}$$

The circuit in Fig. 5 implements such a system for $\alpha' = 3$, $\beta' = 1$, $c_1 = 0$, $c_2 = 11$, $a_1 = a_2 = 0$ and, $n = 4$. The circuit implementing modulo 11 is shown in Fig. 6. The

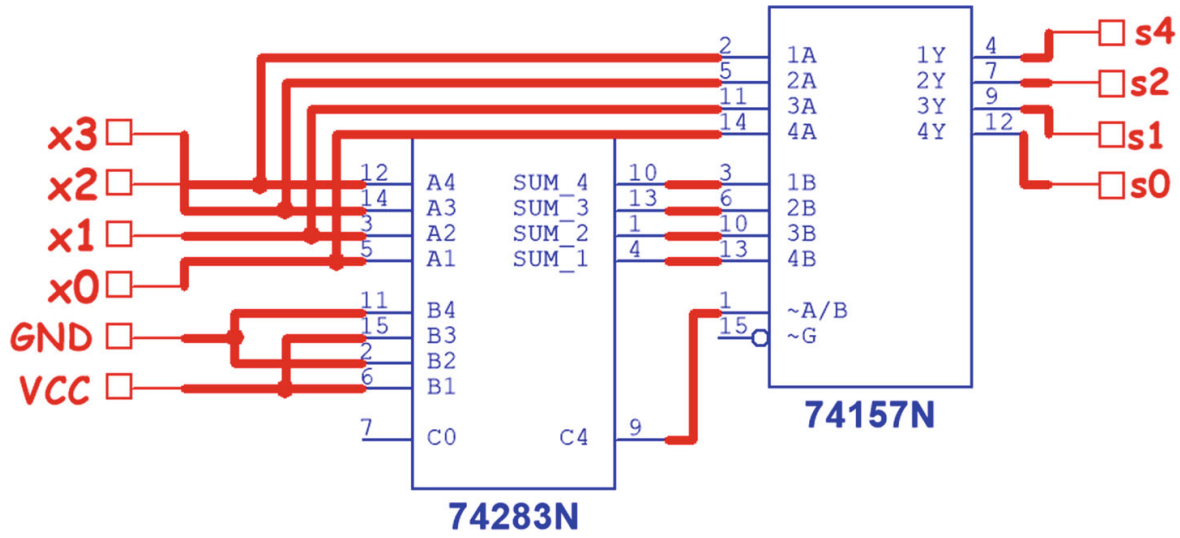


Fig. 6 Electronic implementation of modulo 11 (Modulo_11 module), 4-bit precision

Table 8 Dependence of the period Π on the precision n

n	Π_{QACM}	Π_{PWLCM}
2	3	6
3	6	6
4	12	4.81×10^{12}
5	24	1.06×10^{15}
6	48	1.03×10^{37}
7	96	5.49×10^{53}
8	192	4.28×10^{114}
9	384	1.96×10^{260}
10	768	1.09×10^{513}

period of the above PWLCM is $T = 4.8135 \times 10^{12}$, while that of the corresponding QACM is only $T = 12$. The period gain for this example is therefore $\gamma = 4^{11}$. We simulated this circuit using MULTISIM software as well as PROTEUS/ISIS software and verified that the dynamics of the electronic system perfectly matches with the MATLAB simulation. Based on the low complexity of the electronic circuit and the period gain brought by the insertion of the nonlinear term into the QACM, we concluded that the proposed system can be efficiently included in a pseudo-random number generator.

We analyzed its period for $2 \leq n \leq 10$ and compared it with that of the corresponding QACM. The overall results obtained are summarized in Table 8, from where we can confirm the efficiency of the nonlinear element for increasing the QACM period, thus giving an exponential growth of the period with respect to the precision n. Cases $n = 2$ and $n = 3$ correspond to the QACM with $\alpha = 3$ and $\beta = 1$, as $c_2 > 2^n$, while the periods of the cases $4 \leq n \leq 10$ are evaluated using the Maple software to avoid calculation errors in Matlab. An example of distribution of the orbit periods $T(x_0, y_0)$ with respect to the initial condition (x_0, y_0) for the case $n = 8$ is shown in Fig. 7. One can observe that there are some initial

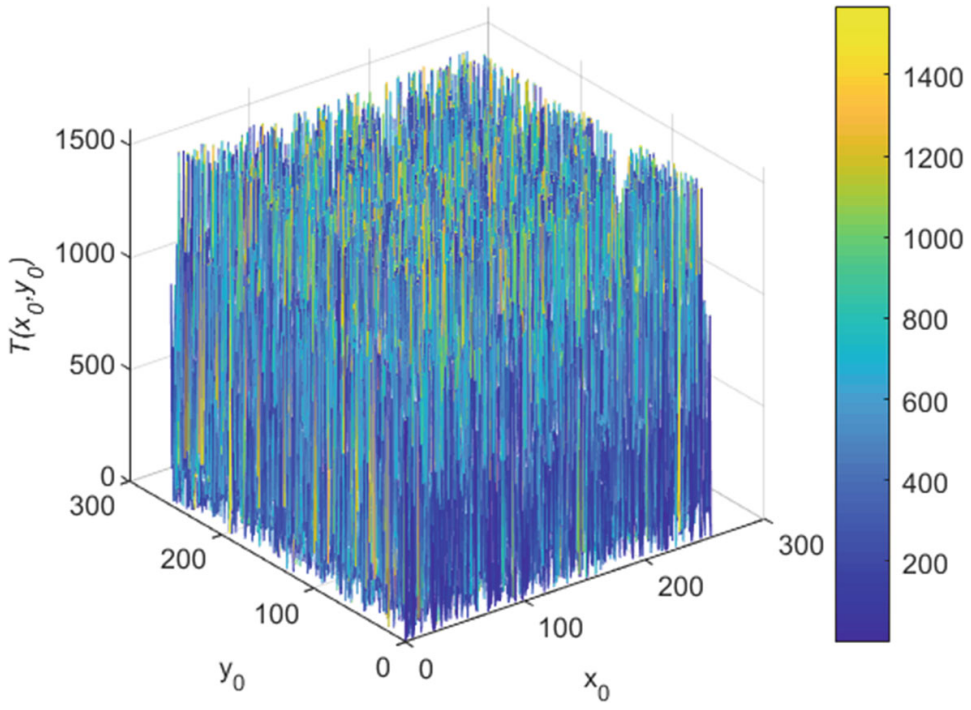


Fig. 7 Distribution of the PWLCM orbit period $T(x_0, y_0)$ with respect to the initial condition (x_0, y_0), case of $n = 8$. The other parameters are set as $\alpha' = 3$, $\beta' = 1$, $c_1 = 0$, $c_2 = 11$ and $a_1 = a_2 = 0$

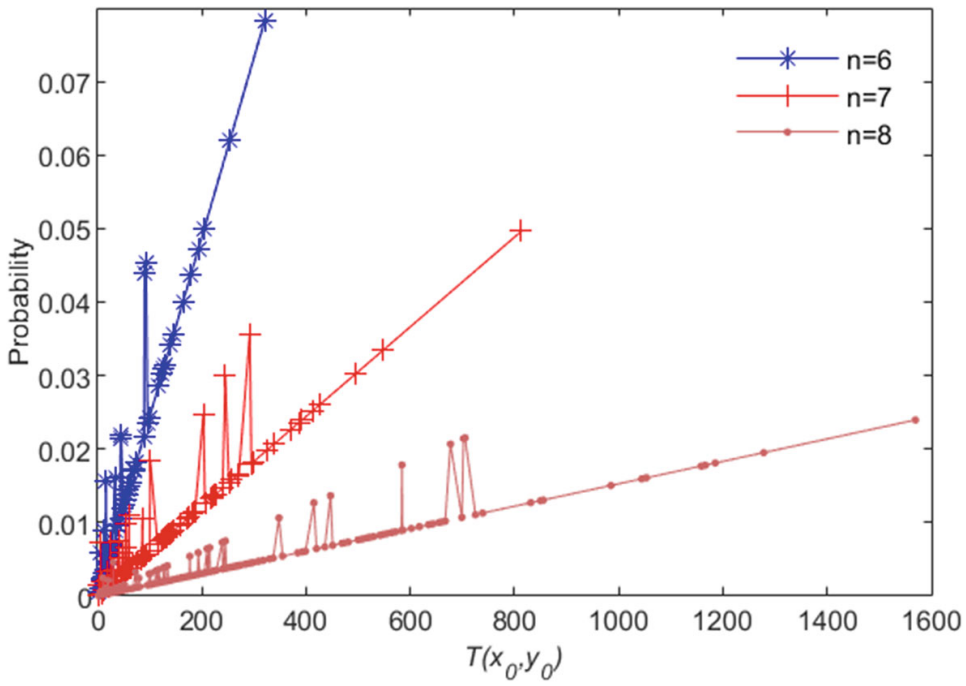


Fig. 8 Probability distribution of $T(x_0, y_0)$ of the PWLCM for $n = 6$, $n = 7$ and, $n = 8$; with $\alpha' = 3$, $\beta' = 1$, $c_1 = 0$, $c_2 = 11$ and $a_1 = a_2 = 0$

conditions for which the period is greater than the upper limit of the orbit periods of the QACM, that is $\pi = 2^n$. Figure 8 shows the probability distribution of $T(x_0, y_0)$ for $n = 6$, 7 and 8. It appears from this figure that the frequency or probability increases with the period: the highest probability corresponds to the largest period $T(x_0, y_0)$. Such a result is interesting as our goal is to obtain large periods for all the nontrivial points of the PWLCM.

4.2 FPGA implementation

In order to confirm the effectiveness of the above architecture simulated with the Multisim software, we used Vivado and implemented the system on Zynq 7020 FPGA board. The schematic of the implemented system is shown in Fig. 9, with $\alpha' = 3$, $\beta' = 1$, $c_1 = 0$,

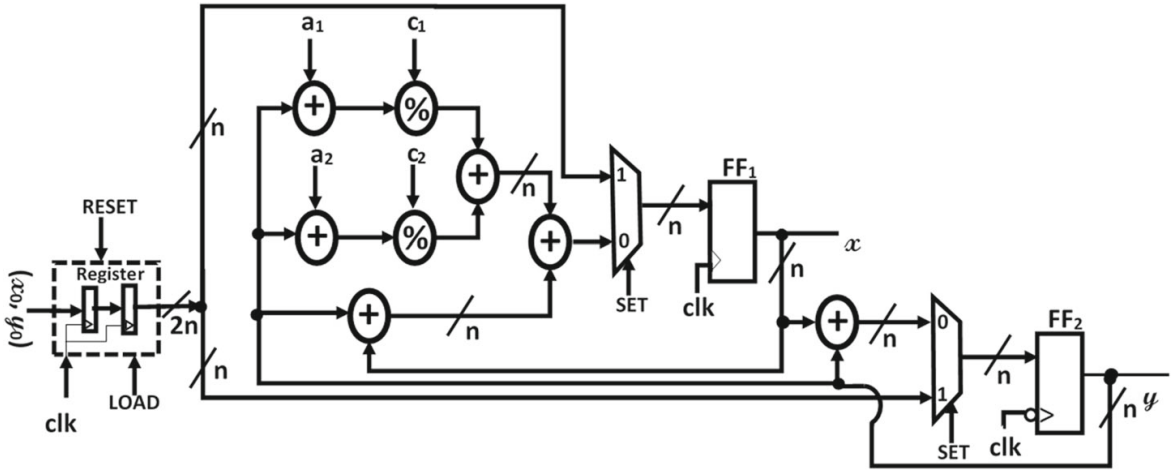


Fig. 9 Generalized FPGA implementation of n-bit PWLCM. The "LOAD" command allows to set x_0 and y_0 as initial conditions in the $2n$-bit register; "RESET" allows to clear the $2n$-bit register; and the "SET" command allows to load the initial conditions to n-bit registers FF_1 and FF_2, therefore to start the system

Table 9 Hardware resource utilization of the 4-bit PWLCM architecture

	Vivado HLx
Data format	Unsigned integer
Technology	Zynq 7020
LUT	16
FF	12
Fmax (MHz)	134
Throughput(Gbps)	1.072

$c_2 = 11$, $a_1 = a_2 = 0$ and, $n = 4$. The resource utilisation as well as the throughput of the proposed architecture are given in Table 9. From this table we observe that the implementation of the PWLCM does not require any DSP module, but exclusively basic modules such as look-up tables (LUT) and flip-flops (FF). We verified that the outputs of the FPGA and Multisim architectures perfectly match, thus confirming the effectiveness of the proposed architectures of the QACM and the PWLCM. For 4-bit precision, the FPGA implementation performs 1.072 Gbps throughput at 134 MHz maximum frequency. Such a high throughput shows that the proposed system can be easily combined with other basic gates such XOR gates or linear feedback shift registers (LSFR) for a real-time generation of pseudo-random numbers.

5 Conclusion

We presented in this paper the PWLCM obtained from a QACM that is nonlinearly perturbed. Depending on the parameter setting, the PWLCM exhibits large periods as compared to the equivalent QACM. The increase of the period enhance the complexity of the PWLCM, thereby is more suitable for image shuffling than the QACM. We evaluated the dependence of the period of the proposed system on the control parameters involved by the perturbing nonlinear term and observed that the worst parameter setting corresponds to a QACM. The PWLCM thus appears to be a generalized form of the QACM in which the sensitivity to the initial conditions has been improved for generating rich and complex dynamics. We showed that the Lyapunov exponent of the PWLCM is sensitive to the initial condition as it is the case for many chaotic systems, which justifies the large periods obtained. We noticed that the period of the PWLCM does not depend on the value of the largest Lyapunov exponent, but on its variability: a large Lyapunov exponent does not imply a large period, but a constant Lyapunov exponent implies a weak period. We also proposed an electronic implementation of both the QACM and the PWLCM. The two circuits are nearly identical, whatever confirms that introducing the perturbation term does not significantly modify the complexity of the QACM, despite the high period gain obtained. The effectiveness of the proposed architecture was confirmed by implementing the system on a Zynq 7020 FPGA board. Both the resource utilisation and the throughput of 1.072 Gbps at a maximum frequency of 134 MHz attest that the PWLCM can be easily combined with other basic gates to design a complete PRNG. The analysis of the model electronically implemented shows that the period exponentially increases with the precision n. In prospect, we intend to apply the PWLCM to data encryption in order to take advantage of its large periods, and to investigate its dynamics in the continuous phase space.

Acknowledgements This work was supported by the Alexander von Humboldt Foundation under Ref 3.4-CMR/1133622.

We thank all the reviewers for their valuable comments that allowed us to improve the content of this paper.

Funding Open Access funding enabled and organized by Projekt DEAL.

References

1. Arnold V, Avez A (1968) Ergodic problems of classical mechanics. New York:Benjamin
2. Bakiri M, Guyeux C, Couchot J-F, Marangio L, Galatolo S (2018) A hardware and secure pseudo-random generator for constrained devices. IEEE Trans Industr Inform IEEE T Ind Inform 14:3754–3765
3. Bakiri M, Couchot J-F, Guyeux C (2018) CIPRNG: A VLSI family of chaotic iterations post-processings for $\mathbb{F}_2$ -linear pseudorandom number generation based on zynq mpsoc. IEEE Trans Circuits Syst I Regul Pap 65:1628–1641
4. Bao J, Yang Q (2012) Period of the discrete Arnold cat map and general cat map. Nonlinear Dyn 70:1365–1375
5. Bonilla LL, Alvaro M, Carretero M (2016) Chaos-based true random number generators. M J Math Industry 7:1–17
6. Chen C, Ma H, aand HC, Meng Y, Ding Q (2015) FPGA Implementation of a UPT chaotic signal generator for image encryption. Pacific Science Review A: Natural Science and Engineering 17:97–102
7. Chen G, Mao Y, Chui C (2004) A symmetric image encryption scheme based on 3D chaotic cat maps. Chaos Solitons Fractals 21:749–761
8. Chen F, Wong K-W, Liao X, Xiang T (2012) Period distribution of generalized discrete Arnold cat map for $n = p^e$. IEEE Trans Inf Theory 58:445–452
9. Chen F, Wong K-W, Liao X, Xiang T (2013) Period distribution of generalized discrete Arnold cat map for $n = 2^e$. IEEE Trans Information Theory 59:3249–3255
10. Chen F, Wong K-W, Liao X, Xiang T (2014) Period distribution of generalized discrete Arnold cat map. Theor Comput Sci 552:13–25
11. Dyson FF, Falk H (1992) Period of a discrete cat mapping. Am Math Mon 99:603–614
12. Eckmann J-P, Ruelle D (1985) Ergodic theory of chaos. Rev Mod Phys 57:617–656
13. Fouda JSAE, Sabat S, multiplierless hyperchaotic system using coupled Duffing oscillators A (2015) Commun nonlinear sci. Numer Simulat 20:24–31
14. Gu G, Ling J (2014) A fast image encryption method by using chaotic 3d cat maps. Optik 125:4700–4705
15. Hu J, Gao JB, Tung WW (2009) The analysis of observed chaotic data in physical systems. Chaos 19:028506
16. Kalanadhabhatta S, Kumar D, Anumandla KK, Reddy SA, Acharyya A (2020) Puf-based secure chaotic random number generator design methodology. IEEE Trans Very Large Scale Integr (VLSI) Syst 28:1740–1744
17. Keating JP, Mezzadri F (2000) Pseudo-symmetries of Anosov map and spectral statistics. Nonlinearity 13:747–775
18. Kocarev L, Sterjev M, Fekete A, Vattay G (2004) Public-key encryption with chaos, chaos: Interdisciplinary. J Nonlinear Sci 14:1078–1082
19. Kumar Panda A, Chandra Ray K (2020) A coupled variable input LCG method and its VLSI architecture for pseudorandom bit generation. IEEE Trans Instrum Meas 69:1011–1019
20. Li C, Lin D, Feng B, Hao F (2018) Cryptanalysis of a chaotic image encryption algorithm based on information entropy. IEEE Access 6:75834–75842

21. Li W, Reidler I, Aviad Y, Huang Y, Song H, Zhang Y, Rosenbluth M, Kanter I (2013) Fast physical random-number generation based on room-temperature chaotic oscillations in weakly coupled superlattices. Phys Rev Lett 111:044102
22. Li C, Tan K, Feng B, Lu J (2017) The graph structure of the generalized discrete Arnold's cat map, arXiv:1712.07905, pp 1–15
23. Lou D, Sung C (2004) A steganographic scheme for secure communications based on the chaos and euler theorem. IEEE Trans Multimedia 6:501–509
24. Merah L, Lorenz P, Adda A-P (2021) A new and efficient scheme for improving the digitized chaotic systems from dynamical degradation. IEEE Access 9:88997–89008
25. Öztürk I, Kiliç R (2015) A novel method for producing pseudorandom numbers from differential equation-based chaotic systems. Nonlinear Dyn 80:1147–1157
26. Öztürk I, Kiliç R (2021) Utilizing true periodic orbits in chaos-based cryptography. Nonlinear Dyn 103:2805–2818
27. Rameshbabu R, Karthikeyan R, Balamurali R, Prasina A (2015) FPGA Implementation of adaptive complete synchronization methodology for novel chaotic systems, Middle-East. J Sci Res 23:36–44
28. Rezk AA, Madian AH, Radwan AG, Soliman AM (2020) Multiplierless chaotic pseudo random number generators. AEU-International Journal of Electronics and Communications 113:152947
29. Shah DK, Chaurasiya RB, Vyawahare VA, Pichhode K, Patil MD (2017) FPGA Implementation of fractional-order chaotic systems. Int J Electron Commun (AEU) 78:245–257
30. Souza CEC, Shavez DPB, Pimentel C (2018) One-dimensional nonlinear model for producing chaos. IEEE Transactions on Circuits and Systems I: Regular Papers 65:235–246
31. Wang D, Xu W, Xu J, Gu X, yang G (2019) Resonance responses in a two-degree-of-freedom viscoelastic oscillator under randomly disordered periodic excitations. Commun Nonlinear Sci Numer Simulat 68:302–318
32. Wang Y, Liu Z, Zhang LY, Pareschi F, Setti G, Chen G (2021) From chaos to pseudorandomness: A case study on the 2-d coupled map lattice. IEEE Trans Cybern, pp 1–11
33. Zhua H, Zhao C, Zhang X, Yang L (2014) An image encryption scheme using generalized Arnold map and affine cipher. Optik 125:6672–6677