الجمهورية الجزائرية الديمقراطية الشعبية
République Algérienne démocratique et populaire

وزارة التعليم العالي و البحث العلمي
Ministère de l'enseignement supérieur et de la recherche scientifique

جامعة سعد دحلب البليدة
Université SAAD DAHLAB de BLIDA

كلية التكنولوجيا
Faculté de Technologie

قسم الإلكترونيك
Département d'Électronique

Mémoire de Master

En Télécommunication
Spécialité : Systèmes des Télécommunications

Présenté par

EL FODIL Marouane

&

LAHFAIR Soumia

Étude et implémentation sur FPGA d'un générateur de nombres aléatoires Gaussiens avec la méthode Ziggurat

Proposé par : M.MAAMOUN Mountassar

Année Universitaire 2021-2022


بسم الله الرحمن الرحيم

Remerciements

Nous tenons à remercier en premier lieu ALLAH le tout puissant de nous avoir procuré la santé, la patience et la volonté pour réaliser ce modeste travail.

Nous exprimons nos reconnaissances et nos respects à Mr MAAMOUN Mountassar, notre promoteur, pour ses conseils pertinents, sa disponibilité, et son soutientout au long de la réalisation de ce mémoire.

Un grand remerciement aux membres du jury chacun par son propre nom, Mm Chagaga et Mr Dahmani pour l'honneur et l'intérêt qu'ils nous ont accordé en acceptant d'examiner et d'évaluer notre travail.

Nous tenons à exprimer notre reconnaissance et gratitude aux enseignants du département d'électronique de manière générale, et à tous ceux qui ont contribués de près ou de loin à l'élaboration de ce travail.

Sans oublier de remercier nos chère parents, qui étaient toujours à nos côtés et nous ont aidés et soutenu.

Merci à tous.

Je dédie ce modeste travail accompagné d'un profond amour,

Àcelle qui m'a arrosé de tendresse et d'espoirs, à la source d'amour incessible, qui s'est sacrifié pour mon bonheur et ma réussite, à ma chère mère Nassiba.

Àmon support dans la vie, qui a été toujours présent avec son soutien indéfectible, qui a consacré sa vie pour notre bien-être à mon cher père Mohamed.

À ma sœur Sara avec qui je partage les moments de joies et de peines.

À mes grands-parents, mes tantes, mes oncles, et tous mes cousins.
Àtous les membres de la famille EL FODIL et HARROU.

À tous mes enseignants et spécialement mon encadreur Mr MAAMOUN Mountassar.

Et à tous ceux qui m'aiment et qui croient en moi.

Marouane

""La terre qui a travaillé à labourer se réjouira le jour où elle sera récoltée""

Je dédie ce travail à :
Toute ma famille particulièrement à mes chers parents
Abd elKader et Bahia qui n'ont jamais arrêté de m'encourager.

À mes sœurs (Karima, Halima, Amel et Fella) qui ont toujours été soucieuses de mon sourire et de mon confort.

À mes neveux (Aya, Mohamed et Hiba) que je considère comme une source d'inspiration et que je vois en eux un beau souvenir de mon enfance.
Que Dieu les protège.

Àtous les membres de la famille LAHFAIR et REBAH.
À tous ceux qui m'ont aidé de près ou de loin pour pouvoir réaliser ce travail.

Soumia.

ملخص:

تلعب الأرقام العشوائية دور هام في القيام بالحسابات لأخذ عينات المترجعة مما يجعل هذه العملية حساسة و تعتمد بشكل كبير على الخورزميات مما يدفعنا الى تقديم حل دقيق لانجاز هذا النوع من التطبيقات في مشروع نهاية الدراسة هذا سنقوم بتنفيذ طريقة وهي الزقورة و التي تقوم على توليد أرقام عشوائية و اختيارها للمحاكاة متغير عشوائي له كثافة رتيبة بدقة و يسمح بتقليل الحسابات و الحصول على نتائج سريعة استنادا على جداول محسوبة مسبقا و هي أسرع الطرق للكثافات العادية و الأسية و غيرها من الكثافات المتناقضة و ذلك باستعمال برنامج Matlab و جهاز FPGA

الكلمات الدالة :

Matlab , FPGA , زقورة

Résumé :

Les nombres aléatoire jouent un role important dans la réalisation des calculs des retours d'échantillonnage , ce qui rend ce processus sensible et fortement dépendant des algorithmes ce qui nous amène à proposé une solution robuste et précise pour réaliser ce type d'applications. nous allons implimenter la méthode Ziggourat, qui est basée sur la génération de nombres aléatoires et leur sélection pour simuler une variable aléatoire qui a une densité strictement monotone et d'obtenir des résultats rapides basés sur des tableaux pré-calculés, et c'est le plus rapide méthode pour les densités normales et exponentielles et autres densités contrastées en utilisent le logiciel MATLAB et matériel FPGA .

Mots clés :

FPGA , MATLAB , Ziggourat

Abstract

Rnadom number play an important role in the realization of the calculations of the sampling returns , which makes this process sensitive and strongly dependent on the algorithms which leads us to propose a rebust and prcise solution to carry out this type of applications.we will implement the Ziggurat method ,which is based on the generation of random numbers and their selection to simulate a random variable that has a strictly monotonic density and allows to reduce calculations and obtait fast results based on pre-calculated tables , and it is the fastest method for normal and exponential densities and other contrasting densities using software Matlab and hardware FPGA

Key words :

FPGA , MATLAB , Ziggurat

Listes des acronymes et abréviations

ADSL : Asymmeric Digita Subscriber Line

ASIC : Application Specific Integrated Circuit

ALU : Architecture and Logic Unit

BRAM : The RAM Blocks

CLB : Configuration Logic Blocks

CNA : Analog Digital Conversion

DSP : Module Digital Signal Processor

DPRAM : Dual Ported Random Access Memory

EBR : Electronic Batch Record

FPGA : Field Programmable Gate Array

FIFO : First In First Out

FXP : Fixed Point

GRN : Random Number Generator

IEEE: Institute of Electrical and Electronic Engineers

IOB : Input Output Block

LUT : Luck Up Table

LFSR : Linear Feedback Shift Register

MOS : Metal Oxide Semiconductor

MSB : Most Sinificant Bit

PDF : Probability Density Function

RTC: Switched Telephone Network

RISC : Reducted Instruction Set Computer

SOP : Union of Postal Operators

SD : Secure Digital

TRNG : Random Number Generator

VHDL: Verilog Hardware Description Language

VLSI : Very Large Scale Integration

Table des matières

Liste des figures

Liste des tableaux

Introduction générale

La conception l'implémentation d'une générateur de nombres aléatoires gaussiens sur FPGA, exige un compromis entre complexité matérielle et performances aussi pour générer des échantillons d'une variable aléatoire suivant une loi uniforme .Si cette loi occupe une place importante de produire une séquence binaire aléatoire indépendante, uniformément distribuée. Un RNG est essentiel dans toutes sortes d'application telles que la cryptographie, les télécommunications, les simulations informatiques, les tests de circuits VLSI ou les algorithmes probabilistes.

Chapitre 1 : est organisée de la manière suivante. Dans une première partie,nous présenterons la technologie FPGA en rappelant les principales caractéristiques du FPGA et des architectures de système à base de FPGA. Le module DSP dans sa technologie et ses caracterisques sera décrit à la deuxième partie. Par la suite, nous parlerons des blocs logiques contrôlables (CLB) pour chuter sur les blocs Ram (BRAM).

Chapitre 2 : nous allons introduire l'architecture de quelques générateurs classiques des nombres aléatoires gaussiens. Pour cela, nous présenterons de prime abord la méthode de la limite centrale. Par la suite, nous parlerons de la méthode de Box-Muller. La méthode récursive (ou méthode de Wallace) sera introduite a la troisième partie de ce chapitre et nous sortirons par la méthode de rejet.

Chapitre 3 : nous essayons de présenter l'algorithme de Ziggurat en virgule fixe. Pour y arriver, nous allons dans une première partie décrire le principe de la Ziggurat en passant par son architecture matérielle. La deuxième partie sera consacrée au codage en virgule fixe. Il sera donc question de décrire le principe de fonctionnement de ce système de codage. Dans la troisième et dernière partie, nous parlerons du fonctionnement matériel de la méthode de Ziggurat en virgule fixe.

Chapitre 1 Logique programmable et calcul sur FPGA

1.1 Introduction

La technologie FPGA actuellement disponible permet de construire des systèmes qui sont configurables et/ou reconfigurables. L'idée générale avec tout les circuitsconfigurables est de proposer non pas une fonction figée dont le champ d'application est le plus large possible, mais une structure adaptable. La reconfiguration consiste alors à modifier en cours de fonctionnement l'architecture matérielle, cela permet de concevoir les circuits intelligents qui peuvent s'auto-adapter à leur environnement.

1.1.1 Présentation des FPGA

Les FPGA (field programmable gatearray) sont inventés par la société Xilinx en 1985. Xilinx fut précurseur du domaine en lançant le premier circuit FPGA commercial, le XC2000. Ce composant avait une capacité maximum de 1500 portes logiques. La technologie utilisée était alors une technologie aluminium à 2 micromètre avec 2 niveaux de métallisation. Xilinx ne sera suivi qu'un peu plus tard, et jamais lâchée, par son plus sérieux concurrent Altera qui lança en 1992 la famille de FPGA FLEX 8000 dont la capacité maximum atteignait 15 000 portes logiques [1] Les FPGA se situent entre les réseaux logiques programmables et les circuits logiques près diffusés. Les réseaux logiques programmables sont des composants qui ne nécessitent aucune étape technologique supplémentaire pour être personnalisés, ce sont des circuits standards, programmables par l'utilisateur grâce aux différents outils de développement et qui incluent un grand nombre de solutions basées sur les variantes de l'architecture des portes 'ET' et 'OU' . Les prés diffusés sont des circuits intégrés basés sur l'utilisation des réseaux de cellules dont les blocs ont été préalablement diffusés, il faut créer les connexions entre ces blocs.

Les FPGA combinent donc à la fois la souplesse de la programmation des réseaux logiques programmables et les performances des circuits prés diffuses. Les langages HDL comme le VHDL ou le Verilog sont utilisés pour décrire les fonctionnalités qui seront implémentées sur le composant, puis la description matérielle est traduite dans un fichier de configuration pour le FPGA cible [2,3]. La description matérielle peut être aussi utilisée pour la description d'un composant ASIC. Bien que ces valeurs soient relativement réduites par rapport aux ASICs, elles sont suffisantes pour une très large majorité d'applications actuelles. À partir des années 2000, les capacités des FPGA ont permis

d'offrir aux concepteurs une solution supplémentaire de réalisation pour une majorité d'applications. De plus, les outils de mise en œuvre des FPGA ont évolué et bien qu'encore pénalisants lors de la conception, ils permettent la réalisation rapide d'applications complexes.

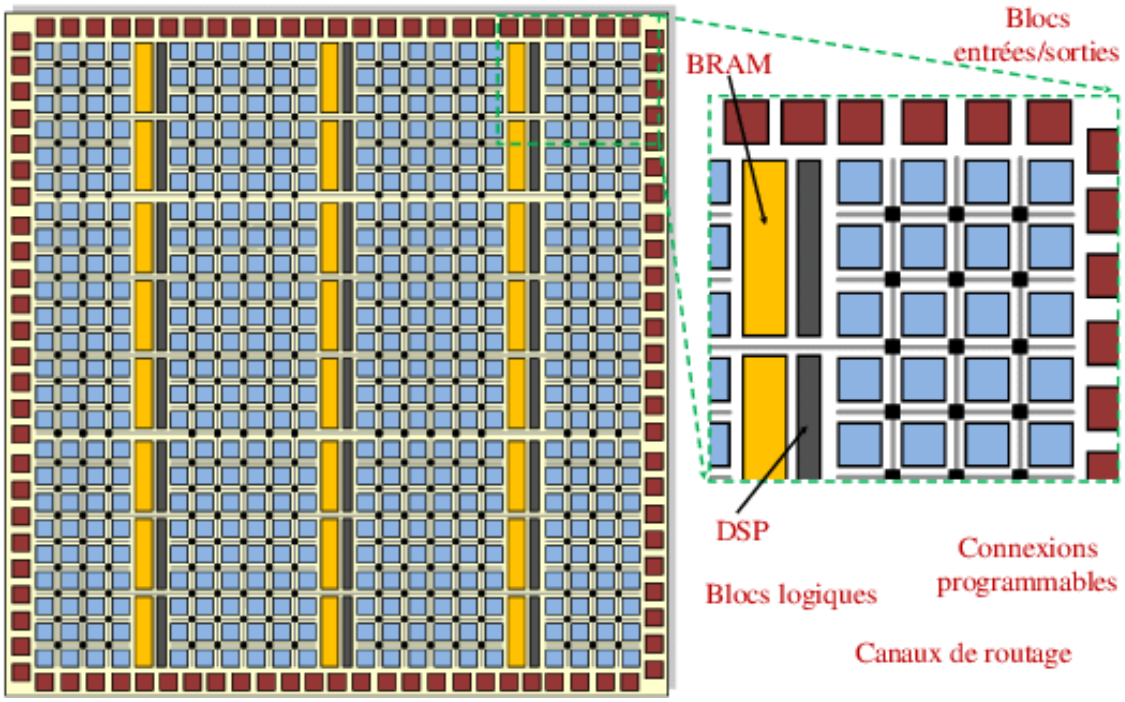

Figure1.1 *: Architecture interne d'un FPGA [4]*

L'architecture du FPGA présentée Figure 1.1 est composée :

➢ De cellules d'entrées sorties modifiables qui servent d'interfaces entre les broches du circuit et le cœur du FPGA pour adapter les signaux suivants [4] :

- Alimentation
- Signaux d'horloge
- Signaux de configuration du FPGA
- Signaux de test

➢ De blocs logiques ou éléments logiques contenant les fonctions logiques combinatoires et séquentielles.

-La partie combinatoire permet de réaliser des fonctions de complexité moyenne avec des portes classiques ET, OU et NON de deux à une dizaine d'entrées.

➢ De réseaux d'interconnexions : Ces réseaux relient entre eux les blocs logiques et les blocs d'entrées/sorties. Ces connections peuvent directement relier :

-Des éléments internes dans un bloc grâce à un système de tables logiques appelées LUT. C'est une matrice de connections où les points de routage déterminent le niveau des entrées soit haut soit bas des portes logiques.
-Des éléments proches : on parle de liaisons directes entre les blocs.

1.1.2 Les blocs principaux d'un FPGA

a- Les blocs d'entrée sortie

Les blocs d'entrée-sortie permettent l'interconnexion de la logique interne aux ports d'entrées et de sorties du FPGA. Les IOB ont leur propre mémoire de configuration, elle stocke les standards de tension et la direction des ports.

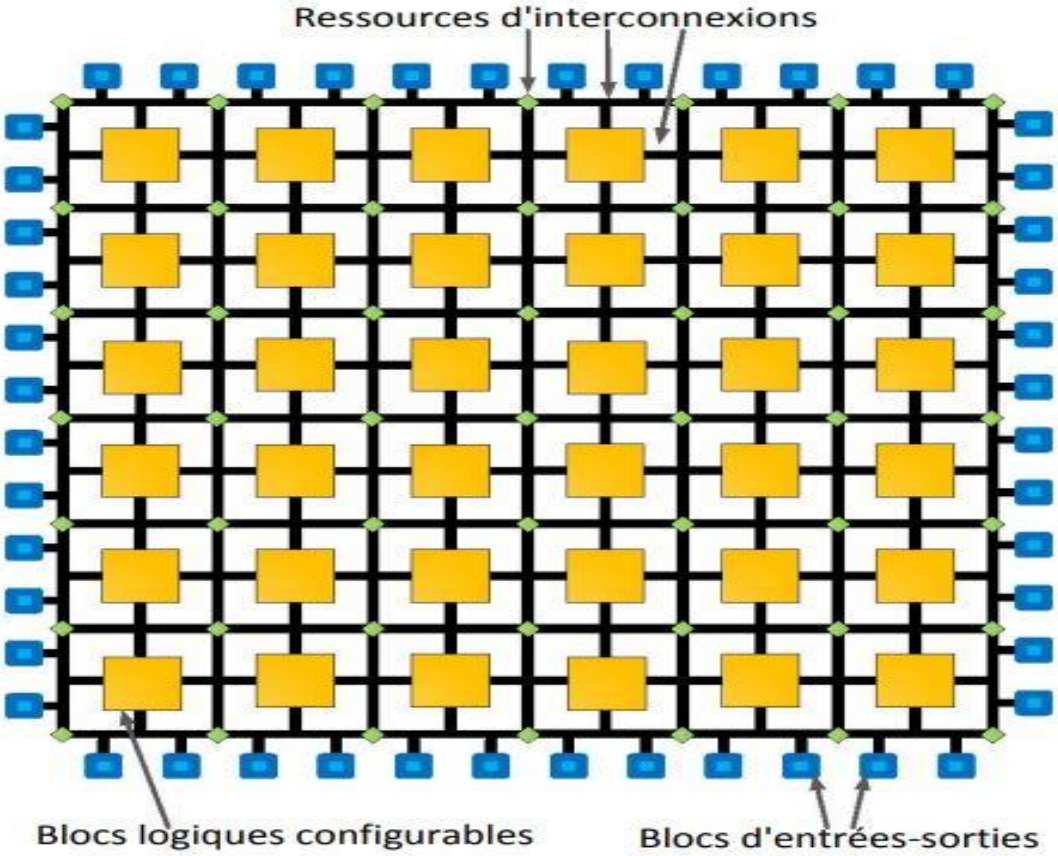

***Figure1.2** : Architecture des blocs principaux d'un FPGA [4]*

Ces blocs sont présents sur toute la périphérie du circuit FPGA. Chaque bloc IOB contrôle une broche du composant et il peut être défini en entrée, en sortie, en signaux bidirectionnels ou être inutilisé (haute impédance) [5] comme le montre la (figure 1.2)

b- Les ressources d'interconnexion

Les ressources d'interconnexion au sein d'un FPGA permettent la connexion arbitraire des CLB et des IOB. Les connexions internes dans les circuits FPGA sont composées de segments métallisés. Elles permettent les connexions entre les diverses lignes, celles-ci sont assurées par des transistors MOS dont l'état est contrôlé par des cellules de mémoire vive ou RAM. Le rôle de ces interconnexions est de relier avec un maximum d'efficacité les blocs logiques et les entrées/sorties afin que le taux d'utilisation dans un circuit donné soit le plus élevé possible. Pour parvenir à cet objectif,

Xilinx propose trois sortes d'interconnexions selon la longueur et la destination des liaisons comme le montre la (figure 1.3)

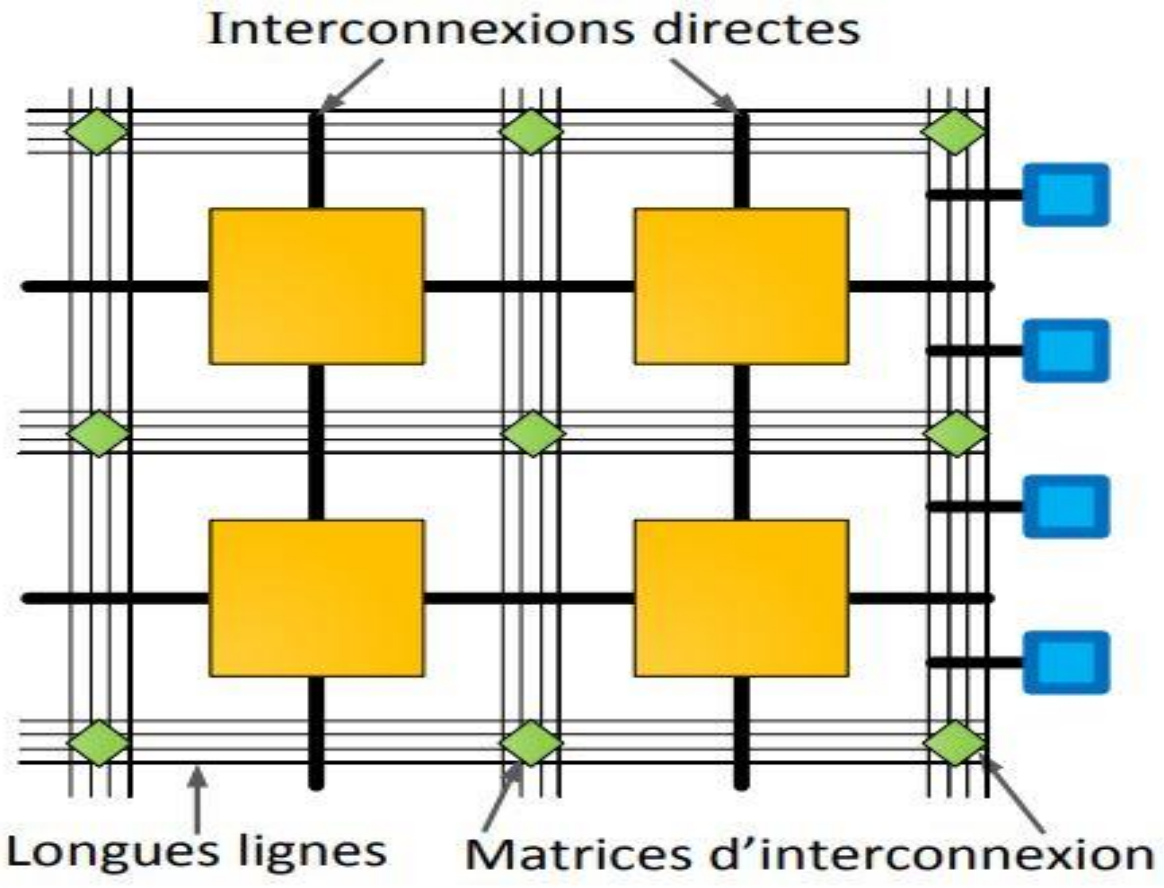

***Figure1.3** : Ressources d'interconnexion de l'architecture Xilinx [5].*

- Les interconnexions directes : Ces interconnexions permettent l'établissement des liaisons entre les CLB et les IOB. Il est possible aussi de connecter directement certaines entrées d'un CLB aux sorties d'un autre.
- Les longues lignes : Ce sont de longs segments métallisés parcourant toute la longueur et la largeur du FPGA, ces longues lignes permettent d'éviter la multiplicité des points d'interconnexion.

- Les matrices d'interconnexion : Ce sont des aiguilleurs situés à chaque intersection. Leur rôle est de raccorder les longues lignes entre elles selon diverses configurations Pour éviter l'affaiblissement des signaux traversant les longues lignes, des buffers sont implantés dans chaque matrice d'interconnexion

Les trois blocs présentés jusqu'ici sont interconnectés ensemble dans le dispositif pour créer une infrastructure de communication composée d'un réseau de communication et d'IOBs autour des CLBs, les normes de tension des entrées-sorties d'un IOB, et les équations soient commandées par des valeurs particulières stockées dans une mémoire comme le montre la (figure 1.4).

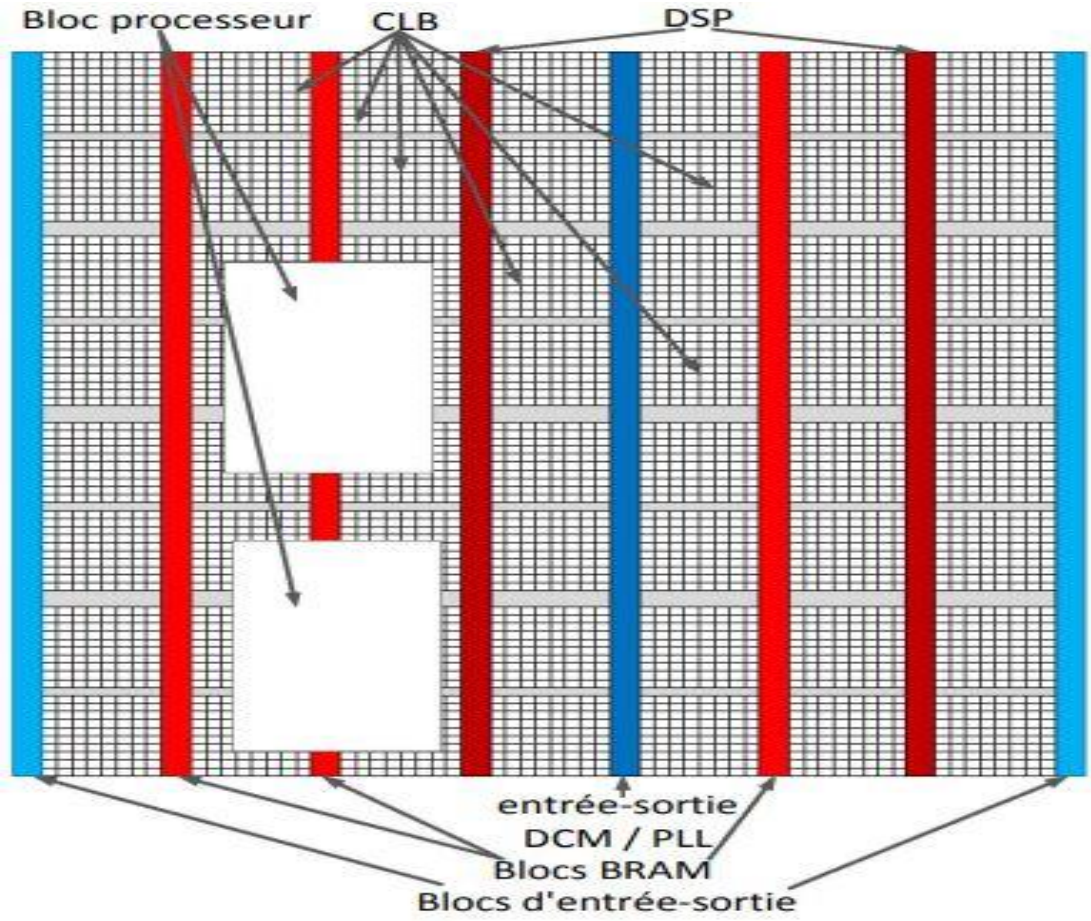

***Figure1.4** : Disposition des blocs à usage spécifique dans un circuit FPGA [4, 6]*

c- Les processeurs embarqués

Les processeurs embarqués enfouis sont l'un des ajouts les plus importants pour le FPGA. Beaucoup de conceptions nécessitent l'utilisation d'un processeur embarqué. Souvent, le choix d'un dispositif de FPGA avec un processeur embarqué (comme le Virtex de Xilinx 5) permet de simplifier grandement le processus de

conception tout en réduisant l'utilisation des ressources et la consommation d'énergie. Le PowerPC IBM 405 et 440 processeurs sont des exemples de deux processeurs inclus dans le Virtex 4 et 5 de Xilinx. Ce sont des processeurs RISC classiques qui mettent en œuvre un jeu d'instructions PowerPC [7].

1.2 Module Digital Signal Processor (DSP)

1.2.1 Présentation du module DSP

Un DSP pour digital signal processor en anglais est un processeur spécialisé dans le traitement numérique du signal. Son architecture est optimisée pour traiter une grande quantité des données en parallèles à chaque cycle d'horloge. Ce mode de fonctionnement est très efficace pour traiter des signaux numériques (filtrage, compression, extraction des signaux etc…). Les DSP sont utilisés dans la plupart des applications de traitement numériques des signaux en temps réel. On les trouve dans les modems (modem RTC, modem ADSL), les téléphones mobiles, les appareils multimédias (lecteur MP3), les récepteurs GPS,… ils sont également utilisées dans les systèmes vidéos, les chaines de traitement du son,

En réalité, afin que les informations qu'ils contiennent puissent être affiche, analysées ou convertis en un autre type de signal, les signaux ont besoins d'un tas de traitement. Dans le monde réel, les produits analogiques détectent les signaux telles que la température, le son, la lumière ou la pression et les manipules. Des convertisseurs telle qu'un convertisseur analogique numérique prennent ensuite ce signal du monde réel et les transforment au format numérique de 1 et de 0. A partir de là, le DSP prend le relais en capturant les informations numériques et en les traitants. Il renvois ensuite les informations numériques pour une utilisation dans le monde réel. Il peut le faire soit numériquement, soit en passant par un convertisseur numérique-analogique.

1.2.2 Architecture mémoire d'un DSP

L'accès à la mémoire est aussi particulièrement optimisé sur les DSP. Les DSP sont en effet capables d'effectuer plusieurs accès mémoires simultanés. Ils peuvent ainsi effectuer plusieurs lectures ou écritures en parallèle. Pour gérer ces accès en mémoires simultanés, un DSP est souvent relié à plusieurs mémoires ou à des mémoires multiports.

Ces DSP sont construits autour d'une architecture Harvard, dans laquelle les bus d'accès à la mémoire programme sont séparés des bus d'accès aux données. Cela leur permet ainsi de lire une instruction tout en changeant des opérandes.

Généralement, un DSP ne possède pas de mémoire caches pour les données. Il arrive qu'il est des mémoires caches pour des instructions, afin d'accélérer l'exécution des certaines boucles, mais c'est relativement rare (figure 1.5)

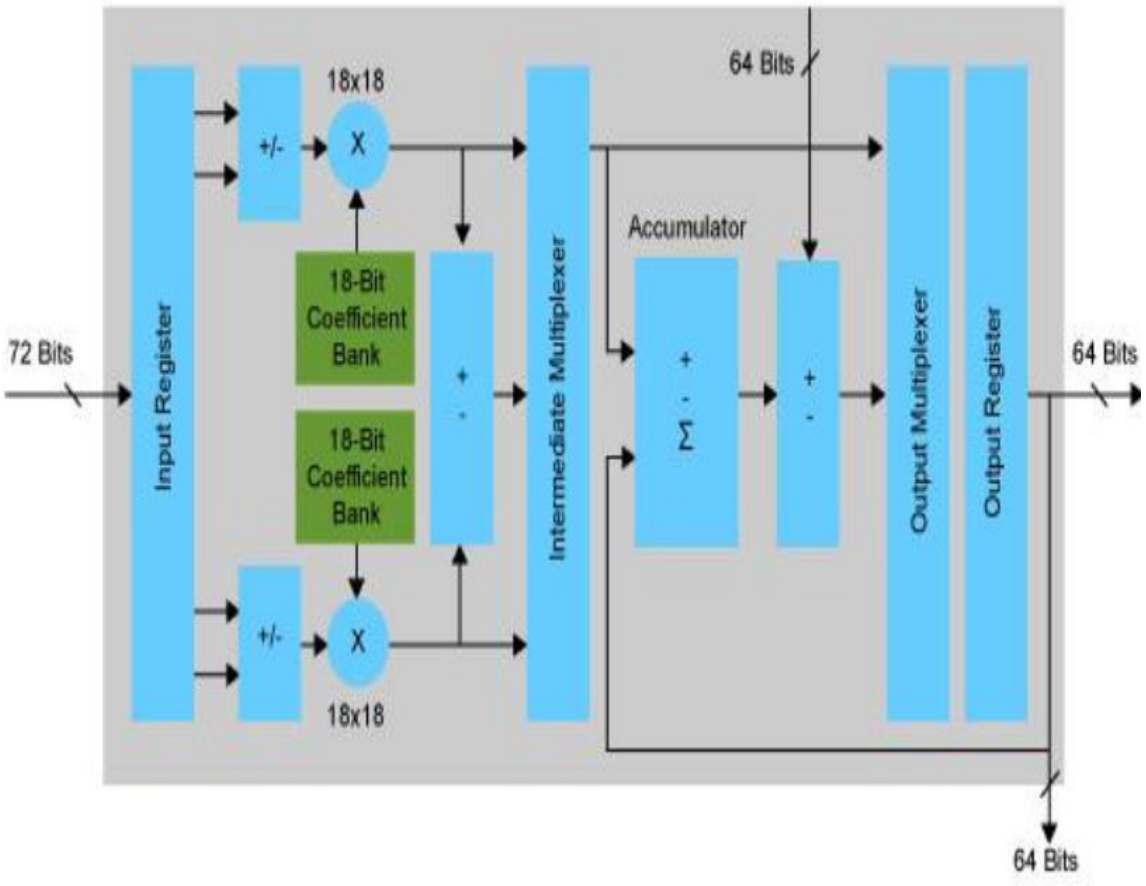

***Figure1.5** : Architecture de bloc DSP sur FPGA*

1.2.3 Caractéristiques d'un DSP

Un DSP possède quelques caractéristiques spécialisées :

- Seulement de processus parallèles pas de multitâches. Des contraintes sont imposées car les DSP n'ont qu'une gestion rudimentaire des interruptions quand ils en ont une ;
- La possibilité d'être utilisées comme périphériques accédant directement à la mémoire dans un environnement hôte ;
- Peut acquérir les données numériques d'un CAN, appliquer des traitements a ces données et les restituer au milieu extérieur grâce a un CNA ;
- Pas de mémoires virtuelles pour diminuer les couts de fabrication et la latence des accès mémoires.

1.3 Les Blocs Logiques Configurables (CLB)

Les blocs logiques configurables sont les principaux éléments d'un FPGA. Ils peuvent avoir un ou plusieurs générateurs de fonctions réalisées avec des tables de correspondance qui peuvent mettre en œuvre une logique arbitraire en fonction de leur configuration. Pendant le processus de configuration d'un FPGA , les mémoires des LUT sont écrites pour y implémenter une fonction, et la logique qui l'entoure est configurée pour router correctement les signaux afin de construire un système complexe de FPGA Spartan . Ces LUT4 peuvent mettre en œuvre une fonction booléenne.

Il existe différents types de CLB en fonction du FPGA utilisé . Pour Xilinx le bloc logique configurable FPGA est appelé slice. Les blocs logiques configurables sont des éléments déterminants des performances des FPGA. Chaque bloc est composé d'un bloc de logique combinatoire composé de deux générateurs de fonction à quatre entrées et d'un bloc de mémorisation-synchronisation composé de deux bascules D. quatre autres entrées permettent d'effectuer les connexions internes entre les différents éléments du CLB (figure 1.6) .

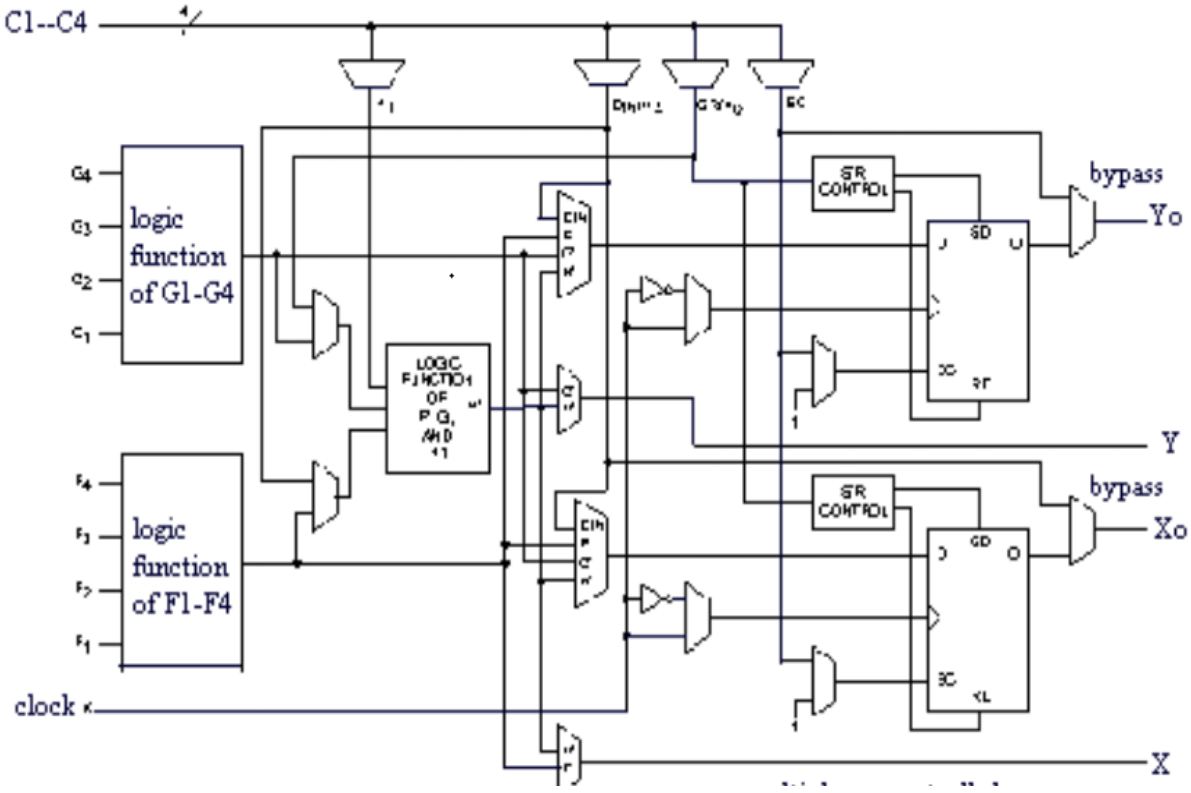

***Figure1.6** : Cellules logiques CLB [10]*

Les deux fonctions sont générées à partir d'une table de vérité câblé inscrite dans une zone mémoire rendent ainsi les délais de propagation pour chaque générateurs de

fonction indépendants de celle à réaliser. Une troisième fonction H' est réalisé a partir des sorties F' et G' et d'une troisième variable d'entrée H1 sortant d'un bloc composé de quatre signaux de contrôle H1, DIN, S/R et Ec les signaux des générateurs de fonctions peuvent sortir du CLB soit par la sortie X pour la fonction F' et G' soit Y pour les fonctions G' et H. ainsi un CLB peut être utilisé pour :

- Deux fonctions indépendantes à quatre entrées indépendantes, plus unetroisième fonction de trois variables indépendantes,
- Toutes fonctions à cinq variables,
- Toutes fonctions à quatre variables et une autre avec quelques fonctions à six variables.

1.4 Les blocs RAM (BRAM)

Les blocs RAM sont des mémoires définies par l'utilisateur, embarquées sur le circuit intégré FPGA, elles servent à stocker des ensembles de données. En fonction de la catégorie de FPGA, la mémoire RAM embarquée est configurable en blocs de 16 ou de 32 kilo-octets. Les mémoires RAM sont de type double ports, elles permettent une écriture et une lecture indépendante sur chaque port avec une horloge différente. Ceci est très utile, le composant peut produire (écrire) des données à une fréquence différente d'un autre composant qui consomme (lire) les données.

Une bloque RAM (parfois appelée mémoire intégrée ou RAM de bloc intégré (EBR) est une partie discrète d'un FPGA, ce qui signifie qu'il y'en qu'un nombre limitée disponible sur la puce. Chaque FPGA a une quantité différente de bloque RAM. Ces composants sont utilisés pour stocker des grandes quantités de données à l'intérieur du FPGA. Les blocs RAM sont de taille fini (4, 8, 16, 32 kilo-octet) sont les tailles courantes. Ils ont une largeur et une profondeur personnalisable et ont vraiment utile pour des nombreuses applications (figure 1.7).

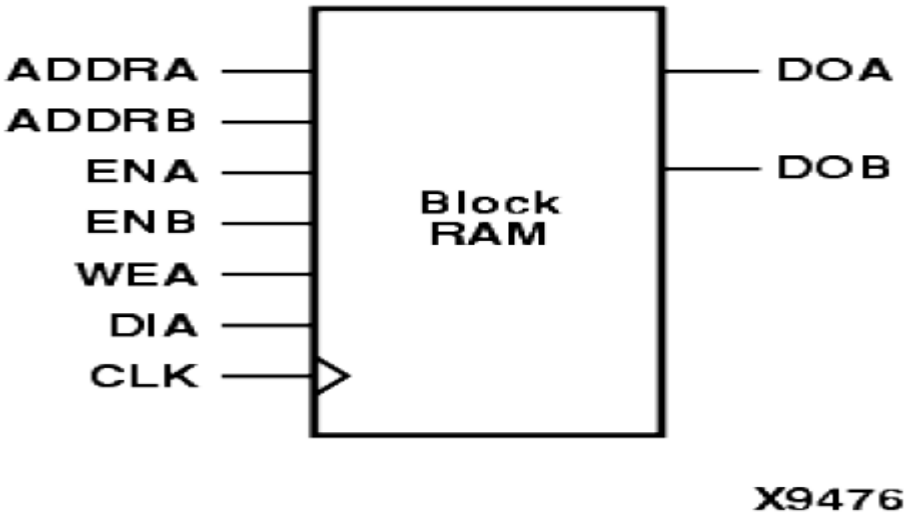

***Figure 1.7** : Bloc RAM X9476*

1.4.1 Configuration d'un bloc RAM à simple port

La configuration simple port d'in bloc RAM est utile lorsqu'une seule interface doit récupérer les données. C'est aussi la configuration la plus simple et elle est utile pour certaines applications. Un exemple serait de stocker les données en lecture seule qui sont écrite sur une valeur fixe lorsque le FPGA est programmé. Leurs fonctionnements sont entièrement basés sur une horloge. Les données seront lues sur le front positif sur cycle d'horloge. Les données lues sortent sur Rd data, ce sont les données stocker dans la BRAM (figure 1.8).

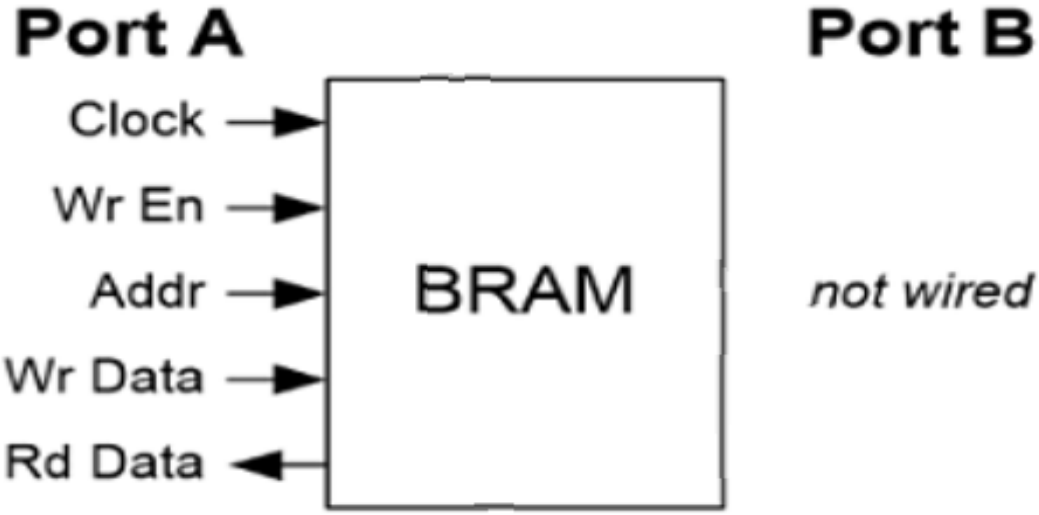

***Figure1.8** : Configuration d'un BRAM a simple port*

1.4.2 Configuration des blocs RAM à double port

La configuration duale port des blocs RAM (la DPRAM) se comporte exactement de la même manière que celle à un seul port, sauf qu'ici on distingue d'un autre port disponible pour la lecture et écriture des données, un cas d'utilisation possible serait de stocker les données sur un périphérique externe. Par exemple, vous souhaitez lire des données sur une carte SD, vous pouvez les stocker dans une RAM à double port et les lire plus tard. Ou peut-être souhaité vous vous connecter à un convertisseur analogique-numérique (ADC) et aurez-vous besoin d'un endroit pour stocker les valeurs ADC convertis. Une DPRAM serait idéale pour cela. De plus, les RAM à double port sont généralement transformé en FIFO qui sont probablement l'un des cas d'utilisation les plus courant pour les blocs RAM sur un FPGA (figure 1.8).

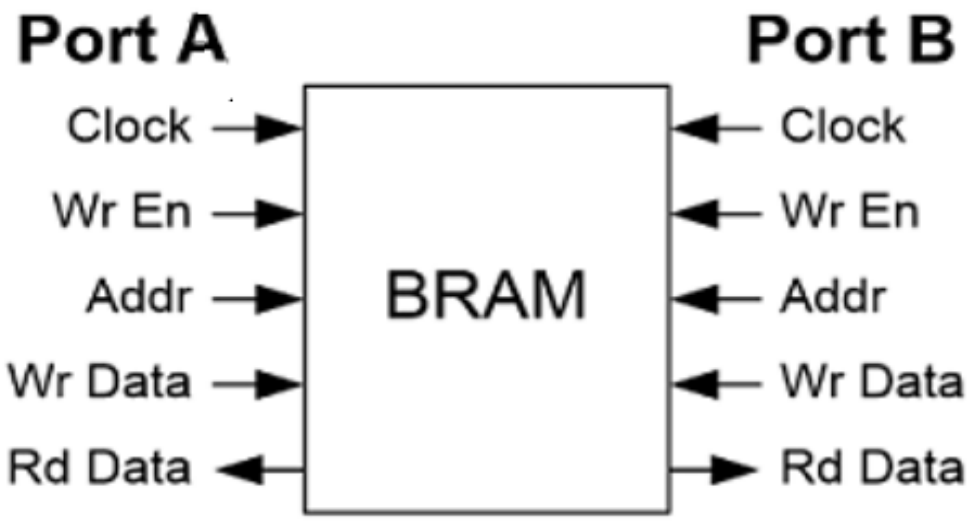

***Figure1.9**: Configuration d'un BRAM a double port*

1.5 Conclusion

Ce chapitre nous a permis dans sa globalité de présenter la notion de circuit programmable. La technologie FPGA a été mise en avant avec son architecture matérielle. Il en est ressorti les différents blocs d'un FPGA et ses principales caractéristiques notamment sa reprogrammation de ce circuit qui lui donne une large variété d'applications technologiques. Le digital signal presser nt aussi été décrit avec leurs différents applications. Tout comme les DSP, les CLB et les BRAM nous ont fait comprendre qu'ils peuvent être montés comme blocs spécifiques dans un circuit FPGA. Néanmoins, la génération des fonctions aléatoires pour ces circuit rase un problème pour une implémentation facile.

Chapitre 2 : Architecture matérielles classiques des générateurs de nombre aléatoire gaussien

2.1 Introduction

Une séquence binaire x constituee de l bits est dite aléatoire lorsque les bits sont indépendants, imprédictibles et équiprobables [11]. Par abus de langue, un générateur de nombres aléatoires et un générateur de bits aléatoires désignent souvent la même notion. Leur objectif est de générer une séquence de mots aléatoires. Les mots appartiennent a un alphabet fini A. dans le cas d'un générateur de bits aléatoires, l'alphabet est de dimension 2 et A$= \{0, 1\}$. Afin d'obtenir des nombres aleatoires, il est possible de regrouper les bits successifs en sortie du générateur de bits aléatoires.

Un générateur de nombre aléatoire est un algorithme déterministe ou indéterministe permettant de générer une séquence binaire x uniforme, inpredictible et independante. La séquence générée par un TRNG est globalement biaisée. Les PRNG sont les algorithmes déterministes basées sur une équation. Un grand nombre de PRNG existe dans la littérature.

2.2 Architecture de la méthode de la limite centrale

2.2.1. Description de la méthode

Le théorème de la limite centrale (TLC) est un théorème extrêmement efficace dans sa mise en œuvre car il produit des GRN en ajoutant simplement n nombres aleatoires avec des PDF arbitraires. Cependant, il n'est presque jamais utilise dans les applications ou une grande précision de queue est requise. En effet, le PDF des nombres produits par le TLC s'écarte très fortement du PDF gaussien idéal pour une Valeur élevé de l'erreur. Pour une Meilleur précision de queue, la valeur de n (et donc le nombre requis d'additionneurs) doit être augmentée. En fait pour produire un PDF parfait, n doit être infiniment grand [11].

2.2.2. Architecture matérielle du théorème de la limite centrale

Dans cette partie, nous décrirons la conception du générateur GRN matériel base sur le théorème de la limite centrale. Comme le montre la (figure 2.1), l'architecture peut être divisée en quatre (04) blocs distincts: Un registre a décalage à rétroaction linéaire (LFSR), un bloc de sommation, un bloc de décision et un calculateur polynomial du premier ordre. Les LFSR sont couramment utilisés dans les simulations de systèmes de communication numériques comme sources de bruit et de données car ils sont faciles à mettre en œuvre et nécessitent des ressources minimales [12, 13]. Cette conception est base sur le LFSR skip-aheadqui suit l'algorithme présenté par Leonard Colvito [13]. Une différence courante que présentent les générateurs des nombres pseudos aléatoires congruentes multiplicatifs est la corrélation entre les échantillons successifs

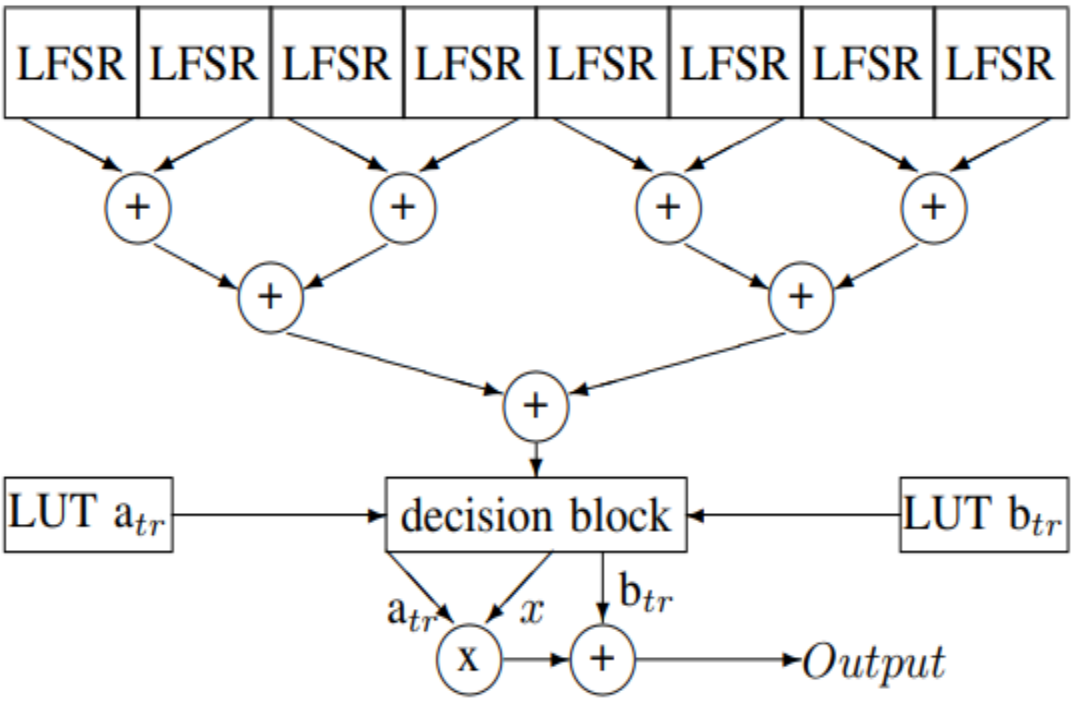

***Figure 2.1**: Architecture matérielle du GNR par le TLC [13].*

Les LFSR sont couramment utilisés dans les simulations de systèmes de communication numériques comme sources de bruit et de données car ils sont faciles à mettre en œuvre et nécessitent des ressources minimales . Cette conception est base sur le LFSR skip-aheadqui suit l'algorithme présenté par Leonard Colvito . Ce problème a été extenué à l'aide d'un schéma LFSR à plusieurs bits, qui fournit des

nombres non corrélés avec une complexité de calcul et des ressources matérielles minimales. Nous implémentons un LSFR 128 bits avec un polynôme de longueur maximale et avance de 16 pour garantir que les échantillons successifs ne sont pas corrélées[14].

Le bloc de décision est la partie la plus importante du générateur GNR matériel. Il comprend deux tables de recherche qui contiennent des coefficients à utiliser avec le calculateur de polynôme du premier ordre.L'espace de recherche du générateur d'adresses est hiérarchisé en veillant à ce que le débit de sortie soit constant [15].

2.3 Architecture matérielle de la méthode de Box-Muller

2.3.1 Description de la méthode

La distribution normale(ou loi Gaussienne) est certainement la plus célèbre des distributions non uniforme, mais elle est également l'une des plus utilisées. Etant donné la difficulté de trouver pour elle une mise en œuvre simple dans le cadre des méthodes universelles (inversion et rejet), un important effort scientifique a été consenti durant les cinquante dernières années pour trouver des méthodes spécifiques à la gaussienne qui soient adéquates [21]. La distribution normale de l'équation a l'avantage de ne pas être entièrement soumise à ses paramètres puisque toute variable x issue de la normale centrée réduite : N (μ=0, σ²=1), peut être manipulé pour obtenir $y \sim N(\mu, \sigma^2)$pour tous $\sigma > 0$et μ. On applique simplement

$$y = \sigma x + \mu$$

Aussi les algorithmes spécifiques à la distribution normale se concentrent sur la distribution normale centrée réduite :

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

La plus célèbre des méthodes spécifiques à la gaussienne est due aux chercheurs Box et Muller et porte de ce fait leur nom. Elle consiste à utiliser un certain nombre d'opérations arithmétiques sur des échantillons de U(0,1) pour aboutir à produire des échantillons x de la fonction f(x), donnée par l'équation :

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

Plus exactement, en prenant, u0 et u1, deux échantillons indépendants, issues de U(0,1), Box et Muller relèvent que les variables x0 et x1 données par :

$$\begin{cases} x_0 = \sqrt{-2\ln u0}\cos(2\pi u1) \\ x_1 = \sqrt{-2\ln u0}\sin(2\pi u1) \end{cases}$$

Elles sont indépendantes et suivent la normale N(0,1). Cette expression est généralement connue comme la forme cartésienne de l'algorithme Box-Muller puisque x0 et x1 sont les coordonnées cartésiennes du vecteur de coordonnées polaires :

$$\begin{cases} R = \sqrt{-2\ln u0} \\ \theta = 2\pi u1 \end{cases}$$

Une forme dite polaire de l'algorithme Box-Muller existe aussi et permet quant à elle d'éviter le recours aux fonctions trigonométriques qui peuvent être couteuses en temps de calcul. Ainsi, en prenant les deux échantillons u0 et u1, de U(0,1) et leur contrepartie respective $X = 2u0 - 1$et $Y = 2u1 - 1$, uniformément réparties sur $]-1,0[\cup]0,1[$, alors les variables x0 et x1 suivent N(0,1) si $s < 1$.

$$\begin{cases} x_0 = X\sqrt{\dfrac{-2}{s}\ln s} \\ x_1 = Y\sqrt{\dfrac{-2}{s}\ln s} \\ s = X^2 + Y^2 \end{cases}$$

Cette forme polaire de l'algorithme Box-Muller appartient à la famille des méthodes de rejet. La méthode polaire est une méthode d'échantillonnage à rejet qui n'utilise qu'une partie des nombres générés par la source aléatoire, mais elle est en pratique plus rapide que la transformation de Box-Muller (Box-Muller cartésienne) car elle est plus simple à calculer :

- Elle n'utilise pas de fonction trigonométrique coûteuse en temps de calcul ;
- La génération de nombres aléatoires uniformes est plutôt rapide, il n'est donc pas gênant d'en gaspiller une partie : En moyenne, la part de points rejetées est $(1 - \frac{\pi}{4}) = 21.46\%$. On génère donc en moyenne $\frac{4}{\pi} = 1.2732$ nombres aléatoires pour obtenir chaque nombre aléatoire normal.

Dans la pratique, la forme polaire compense le rejet (dont le ratio $= 1.2146$) par une acceleration des calculs des variables x0 et x1 en évitant le calcul d'un sinus et d'un cosinus.

2.3.2 Architecture matérielle de la méthode

Deux spécifications clés sont de vireur pour ce générateur: la périodicité des échantillons de bruit 10^{15} et 16 bits. Afin de respecter l'exigence de periodicite, l'URNG doit avoir une périodicité d'au moins10^{15}. Mais en meme temps, les échantillons de bruit doivent suivre au plus prés la distribution gaussienne idéale sur la période. En examinant la distribution normale N (0, 1), on constate qu'il faut pouvoir représenter jusqu'a 8,10σ pour une population de 10^{15} echantillons. En d'autres termes, la probabilité que la valeur absolue d'un seul échantillon de cette population soit supérieur a 8,1 est inferieur a 0.5 (figure 2.5).

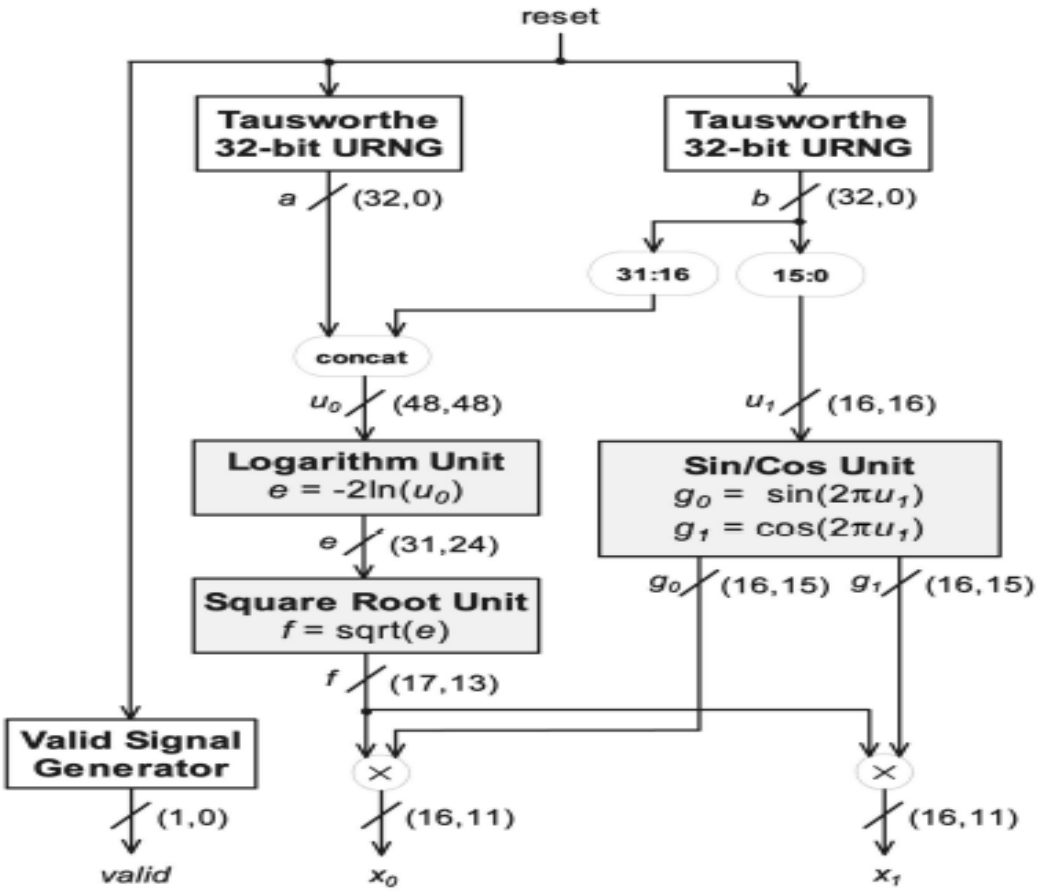

Figure 2.2: Architecture d'un générateur des nombres aléatoires gaussien basé sur l'algorithme de Box-Muller [22].

2.4 Architecture de la méthode récursive ou méthode de Wallace

Wallace propose un algorithme rapide pour générer des nombres pseudo-aléatoire moralement distribués qui génère directement les distributions cibles en utilisant leurs propriétés maximales entrées. Cet algorithme convient particulièrement à la mise en œuvre matérielle du débit élevé [16]. Un aperçu de la méthode de Wallace est décrit à la(figure 2.2).

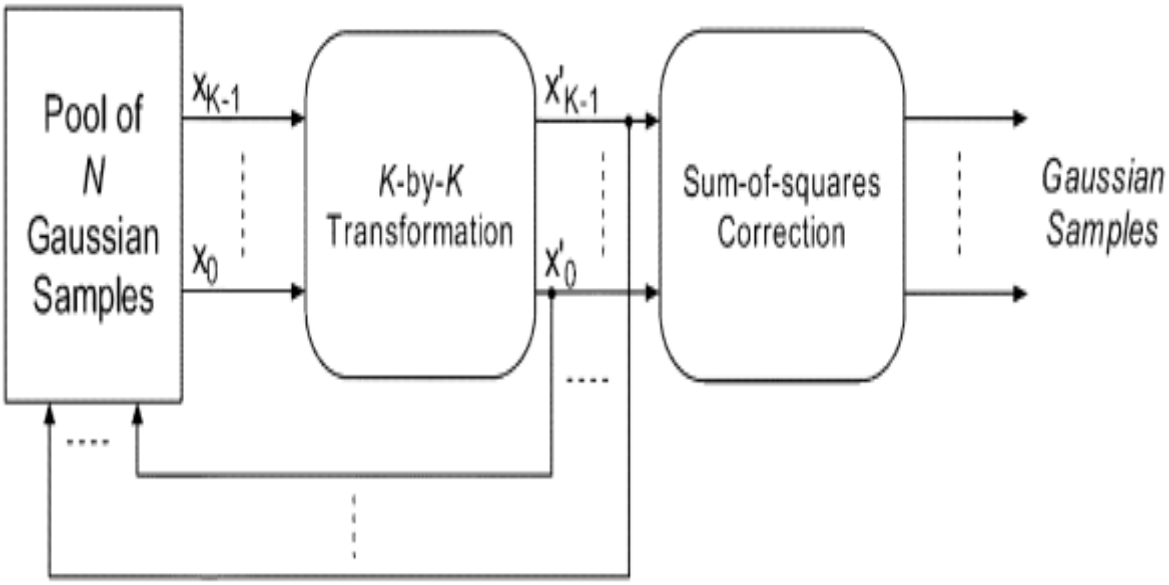

Figure 2.3 *: Un aperçu de la méthode récursive (méthode de Wallace) [16].*

L'architecture matérielle de la méthode de Wallace est une architecture a 4 étapes. Sur la(figure 2.3), les signaux de sélection pour les multiplexeurs et les signaux de validation d'horloge sont omis par soucis de simplicité. Dans la conception de la figure 4, K est choisi à 4 et L a 256 résultants en une taille de pool N de 10124 (ceci est la description originale de Wallace). Sur puce, une véritable RAM synchrone à double port de lecture/écriture est utilisée pour mettre en œuvre le pool. La RAM à double entrée permet de lire et d'écrire simultanément deux valeurs améliorant ainsi la bande passante de la mémoire, les index doivent couvrir tous les nombres du pool et en même temps réduire le nombre de corrélation entre eux. Les adresses qui indexent le pool sont démarrées à partir d'une origine "start" aléatoire, pas d'une "stride" impaire aléatoire et XOR est effectué avec un "mask" aléatoire. La combinaison de ces trois opérations est essentielle pour obtenir un bon mélange entre les échantillons gaussiens dans les tests de coulée montrant que si l'un d'entre eux n'est pas effectué, cela entraine une dégradation de la quantité globale du bruit gaussien. Afin d'obtenir un Meilleur mélange du générateurs des nombres aléatoires

gaussiens, plusieurs type de passes peuvent être utilises lors d'une passe en introduisant différentes matrices orthogonales. Comme dans l'implémentation originale de Wallace, deux matrices orthogonales A0 et A1 sont choisi pour la conception [17].

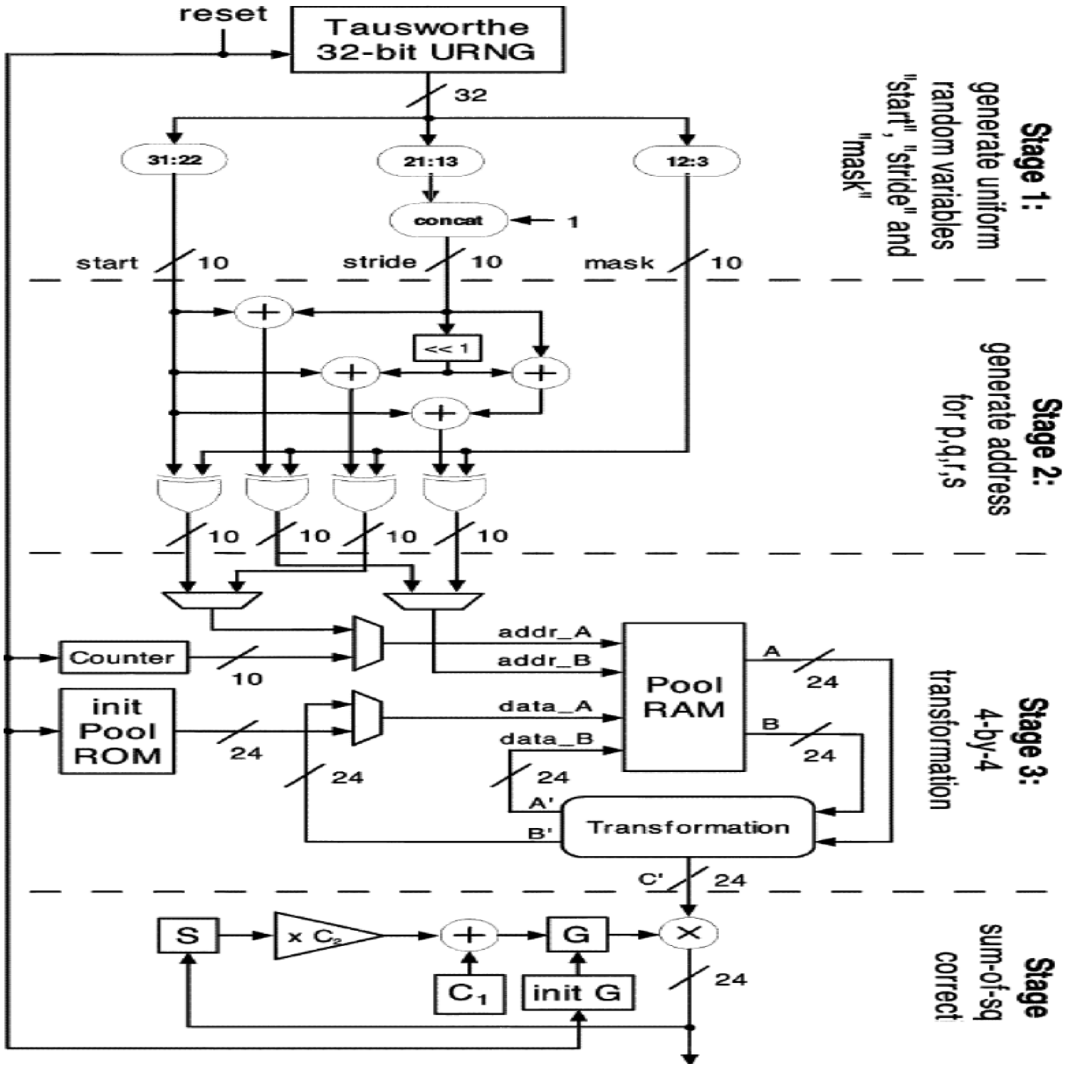

Figure 2.4 *: Architecture matérielle d'un générateur aléatoire des nombres gaussiens par la méthode de Wallace [17].*

$$A_0 = \frac{1}{2}\begin{pmatrix} 1 & -1 & -1 & -1 \\ 1 & -1 & 1 & 1 \\ 1 & 1 & -1 & 1 \\ 1 & 1 & 1 & -1 \end{pmatrix}$$

$$A_1 = \frac{1}{2}\begin{pmatrix} -1 & 1 & 1 & 1 \\ -1 & 1 & -1 & -1 \\ -1 & -1 & 1 & -1 \\ -1 & -1 & -1 & 1 \end{pmatrix}$$

Lors d'une passe, A_0 est utilise pour la premiere moitie et A_1 est utilisé pour la deuxième moitié de la passe. Comme les éléments des matrices A_0 et A_1 ne sont que des 1 et des -1, seule des opérations d'additions et de décalage d'entier sont nécessaires. Les variables aléatoires gaussiennes dans le pool sont conservées sous la forme d'entiers complets à deux de 24 bits. Pour l'exemple donne des 4 valeurs p, q, r et s a transformer et avec notre choix de A_0et A_1, les Nouvelles valeurs p', q', r' et s' peuvent etre calculées a partir des anciens comme suit:

$$p' = p - t;\ q' = t - q;\ r' = t - r;\ s' = t - s;$$

$$p' = t - p;\ q' = q - t;\ r' = r - t;\ s' = s - t$$

Ou

$$t = \frac{1}{2}(p + q + r + s)$$

Il est à noter que les opérations ci-dessus effectuent des transformations réelles, il n'est donc pas nécessaire de stocker A_0et A_1.

a- Première étape

Cette étape implique la génération des réalisations uniformément reparties "start", "stride" et "mask". Alors que les registres à décalage à rétroaction linéaire traditionnel (LFSR) [18] suffisent comme générateur de nombres aléatoires. L'URNG Tausworthe utilise ici suit l'algorithme ressente dans [19], il combine trois générateurs de nombres aléatoires bases sur le LFSR pour obtenir des propriétés statistiques améliorées, génère un nombre aléatoires uniforme de 32 bits par horloge et a une période d'environ 288. Etant donné que la «stride» doit être impaire à tout moment, nous concaténons un après un les bits de poids les moins significative.

b- Deuxième étape

Cette étape génère les adresses des quatre valeurs p, q, r et s du "start", de la "stride" et du "mask". Les quatre adresses sont calcules comme suit:

$$paddr = startmask$$

$$qaddr = (start + stride) * mask$$

$$raddr == (start + 2 * stride) * mask$$

$$saddr = (start + 3 * stride) * mask$$

La multiplication par deux est mise en oeuvre seulement par un quart de vitesse gauche, et la multiplication par 3 est mise en œuvre par un quart de vitesse gauche

suivi d'un ajout. Ce schéma d'adressage garanti que les corrélations entre les variables sont maintenues au minimum.

c- Troisième étape

Cette étape contient le pool RAM qui contient le pool de 1024 variables aléatoires gaussiennes. La RAM à double port est utilisée pour mettre en œuvre le pool. Etant donné que chaque variables du pool est de 24 bits, la taille totale du pool est de 1024x24 = 24576 bits. La «init pool ROM» et leur compteur sont utilises pour initialiser le pool avec le contenu original du pool lorsque le signal de réinitialisation est définie (figure 2.4).

Cette ROM est à port unique et a la même taille que le pool.

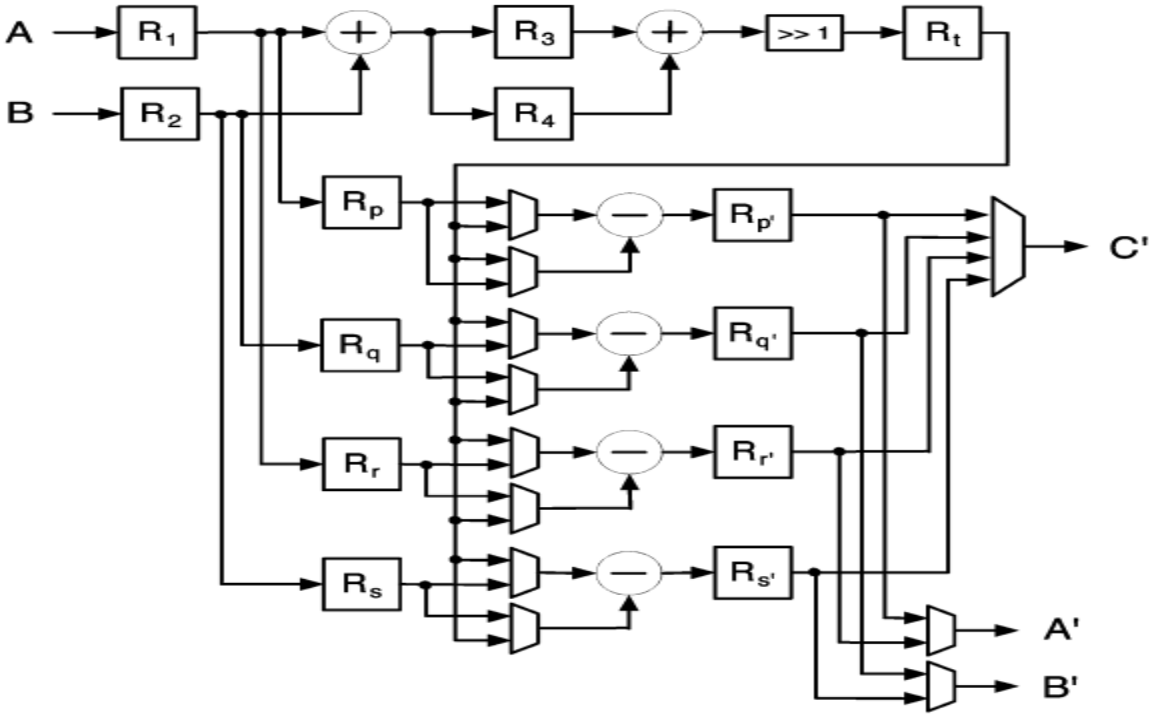

Figure 2.5: *circuit de transformation de l'étape 3. Les carrées représentent les registres. Le signal de sélections des multiplexeurs et le signal de commande d'horloge sont omis pour des simplifications [17].*

En principe, on pourrait se partager un seul additionneur en liaison avec des multiplexeurs pour effectuer toutes les opérations du circuit de transformation. Cependant, les additionneurs a grande Vitesse sont efficacement implémentés sur des FPGA par des chaines de transport rapide. En effet, un multiplexeur 24 bits a deux entrées et un additionneur 24 bits occupant 14 tranches dans un FPGA Xillinx Virtex-II. De plus, l'utilisation de multiplexeurs augmenterait considérablement le délai [20].

d- Quatrième étape

Cette étape effectue la correction de la somme des carrées. Il suit l'approche utilisée par Wallace dans les implémentations FastNorm [23]. Un échantillon aléatoire S avec une distribution x approximative peut etre obtenu comme:

$$S = \frac{1}{2}(C + Ax)^2$$

Ou à une distribution normale unitaire $A = 1 + {}^{1}/_{8N}$ et $C = \sqrt{2N - A^2}$ pour un grand nombre. S peut alors donnee: $S = \sqrt{\frac{1}{2N}}A\,(B + x)$ou $B = {}^{C}/_{A}$. Prennons $C_1 = A\sqrt{2N}$ et $C_2 = B$

le bruit stationnaire généré a la partie (c) est multiplier par G pour corriger la somme des carrees et retourner le bruit stationnaire finale. Ou Gest donne par:

$$G = SxC_2 + C_1$$

Comme C_1 et C_2 sont des constantes, ils sont calcules par le logiciel et stoker dans la conception du circuit. Avant une passe, S est attribuée aux variables de la passe précédente et G est mis à jour [21].

2.5 Architecture matérielle de la méthode de rejet

La méthode de rejet ou algorithme d'acceptation-rejet est une technique de base utilisée pour générer des observations à partir d'une distribution. C'est un type de méthode de simulation exacte et elle est basée sur l'observation que pour échantillonner une variable aléatoire dans une dimension (cela peut être étendu aux fonctions à N dimensions), on peut effectuer un échantillonnage uniformément aléatoire du graphique cartésien bidimensionnel et conserver les échantillons dans la région sous le graphique de sa fonction de densité [23].

2.5.1 Description de la méthode

Pour visualiser la motivation derrière la méthode par rejet, imaginez représenter graphiquement la fonction de densité d'une variable aléatoire sur un grand tableau rectangulaire et lui lancer des fléchettes. Supposons que les fléchettes sont uniformément reparties sur le plateau. Retirez maintenant toutes les fléchettes qui se trouvent hors de la région couverte par la fonction densité. Les fléchettes restantes seront reparties uniformément sous la courbe et les positions x de ces fléchettes seront distribuées en fonction de la fonction de densité de la variable aléatoire [23].

La visualisation telle qu'elle vient d'être décrite équivaut à une forme particulière de la méthode de rejet où la distribution de position est uniforme. La forme générale de la méthode de rejet suppose que le tableau n'est pas nécessairement rectangulaire mais qu'il est façonné en fonction de la densité d'une certaine distribution de proposition à partir de laquelle nous savons échantillonner (en utilisant par exemple la méthode par inversion) et qui est au moins aussi élevée à chaque fois. La méthode de rejet fonctionne ainsi comme suit :

- Echantillonner un point sur l'axe des x à partir de la méthode proposée ;
- Tracer une ligne verticale a cette position x jusqu'à la valeur maximale y de la fonction de densité de la méthode proposée ;
- Echantillonner uniformément le long de cette ligne de 0 jusqu'à la valeur maximale de la fonction de densité. Si la valeur échantillonner est supérieure à la valeur de la distribution souhaité sur cette ligne verticale, rejetez la valeur x et recommencez sinon, x est un échantillon de la distribution souhaitée.

De cette description, on déduit l'algorithme de la méthode de rejet comme suit :

-Obtenir une échantillon y de la diffusion Y et une échantillon u de U (0 ,&)

-Vérifier si oui ou non u < f(y) / M g(y)

- Si oui, accepter y comme un échantillon tiré de f .

-Si non rejetez la valeur de y et recommencer au début .

2.5.2 Description de l'architecture matérielle de la méthode

L'architecture matérielle du générateur de nombre aléatoire par la méthode de rejet peut se résumer comme décrite sur la (figure 2.6).

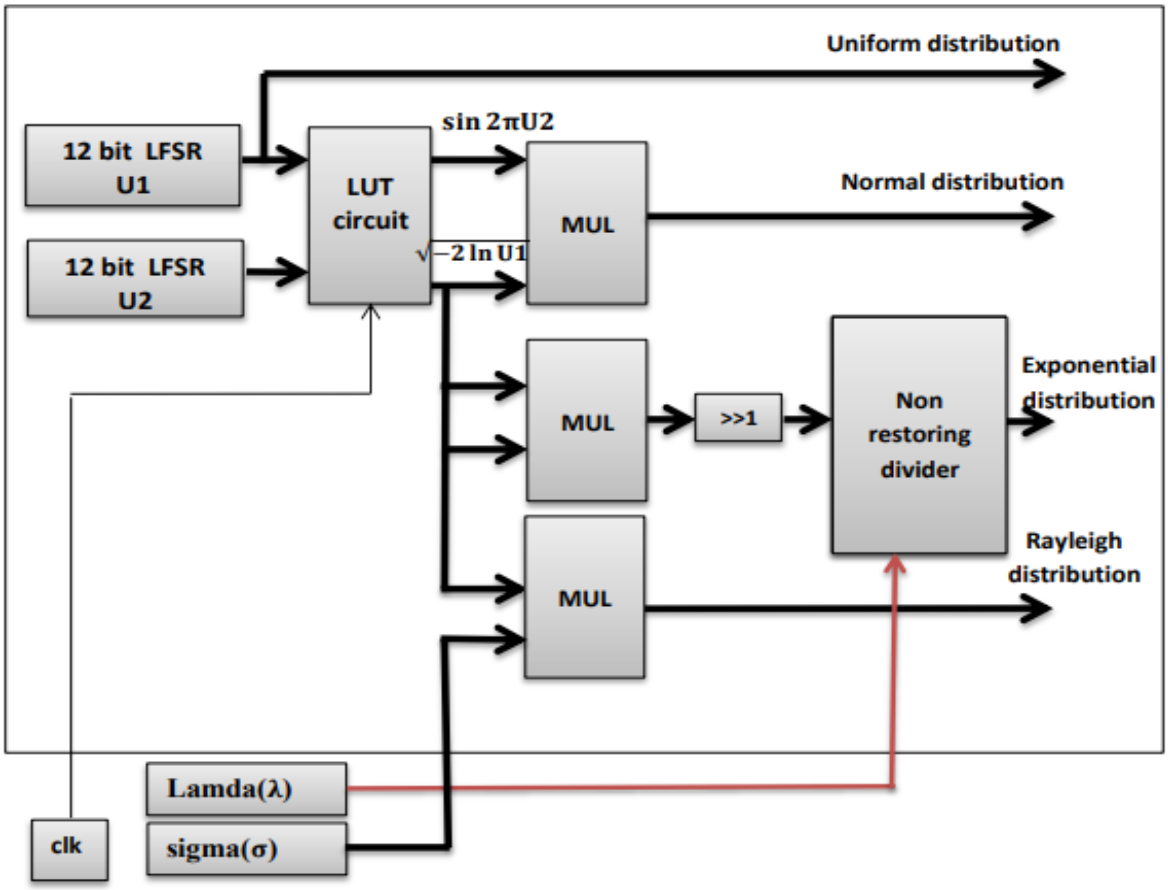

***Figure 2.6** : Architecture matérielle d'un générateur de nombre aléatoire [24]*

Deux LFSR de 12 buts sont utilisés ici pour générer deux variables aléatoires uniformes. La méthode la table de consultation est aussi utilisée pour évaluer les deux variables générées. Puisque le nombre d'état dans le LFSR est de 4096, alors, le nombre d'état des variables est également de 4096 et l'architecture utilise 4 blocs RAM de Spartan3E (chaque bloc RAM est a$1k * 16bits$) [24].

2.6. Conclusion

Dans ce chapitre des générateurs des nombres aléatoire gaussien, nous avons présenté quatre (04) méthodes de conception des générateurs des nombres aléatoires. La méthode de la limite centrale a été présentée dans la première partie et nous y avons fait ressortir et le principe de la méthode qui est basé sur le théorème statistique de la limite centrale mais aussi son architecture matérielle. Dans la deuxième et la troisième partie, nous avons décrit respectivement la méthode récursive et la méthode de Box-Muller : Nous sortons du chapitre par la méthode de rejet. Il est important de remarquer que tant la méthode de Wallace que la méthode de Box-Muller peuvent être vu comme une forme de méthode de rejet car ils fonctionnent tus sur le même principe de rejet. Toujours dans les soucis d'analyser de ces méthodes de rejet, nous verrons au chapitre 3 la méthode de Ziggourat qui se présente être un autre type de méthode de Rejet.

Chapitre 3 Etude de la méthode Ziggurat en virgule fixe

3.1 Introduction

Tout comme la méthode de wallace, la ziggurat s'applique autant à la gaussienne qu'à la distribution exponentielle. Bien que la méthode de wallace soit plus rapide que la ziggourat, la ziggurat est néanmoins privilégiée à cause des garanties théoriques qui l'accompagnent. Ainsi, la ziggurat est largement exploitée en simulation numérique et sert de modèle au générateur de la distribution normale accompagnant la boite à outil statistique de Matlab . De plus, la ziggourat peut s'appliquer à l'ensemble des distributions monotones strictement décroissantes ou symétriquement (comme c'est le cas pour la normale), tout en préservant des qualités de performance remarquables.

La ziggourat repose sur un échantillonnage dans le plan dont l'origine de l'énoncé est créditée par Knuth [25]. Plusieurs énoncés intermédiaires furent étudiés avant d'aboutir à l'élégance de l'expression de la ziggurat qu'offrit Marsaglia en 2000 [26].

L'échantillonnage dans le plan repose sur le découpage de la surface sous la courbe de distribution visée en zones rectangulaires dont l'union couvre une surface Z légèrement supérieur à C. L'algorithme consiste ensuite à générer un point (x, y) de Z et de ne retourner x que si (x, y) est situé à l'intérieur de C (nous avons donc affaire a une méthode de rejet sensiblement identique à celle proposé par John Von Neumann).

3.2 Principe de la méthode de Ziggourat

3.2.1 Définition

La méthode de Ziggourat est un algorithme pour engendrer des nombres aléatoires de la loi non uniforme. Il s'agit d'une méthode de rejet et peut être choisie pour simuler une variable aléatoire ayant une densité strictement monotone. Cette méthode peut aussi être appliquée a des lois symétriques uni modales telles que la loi normale en choisissant un point sur l'une des moitie et en choisissant le cote aléatoirement. Cette méthode a été développée par George Marsaglia et Wai Wan Tang [27].

Comme la plupart des algorithmes de ce type, il suppose que l'on dispose d'un générateur des nombres aléatoires de loi uniforme, en général, un générateur pseudo-aléatoire. Dans la plupart des cas, comme la loi normale ou la loi exponentielle, l'algorithme consiste à engendrer un nombre flottant, un index aléatoire de table, faire une recherche dans une table, une multiplication et une comparaison. Cet algorithme est considérablement plus rapide que les autres méthodes pour simuler les variables aléatoires de la loi normale comme la méthode de Box-Muller qui requièrent de calculer une racine carrée ou un logarithme (voir le chapitre 2) et aussi dans ce (figure 3.1)

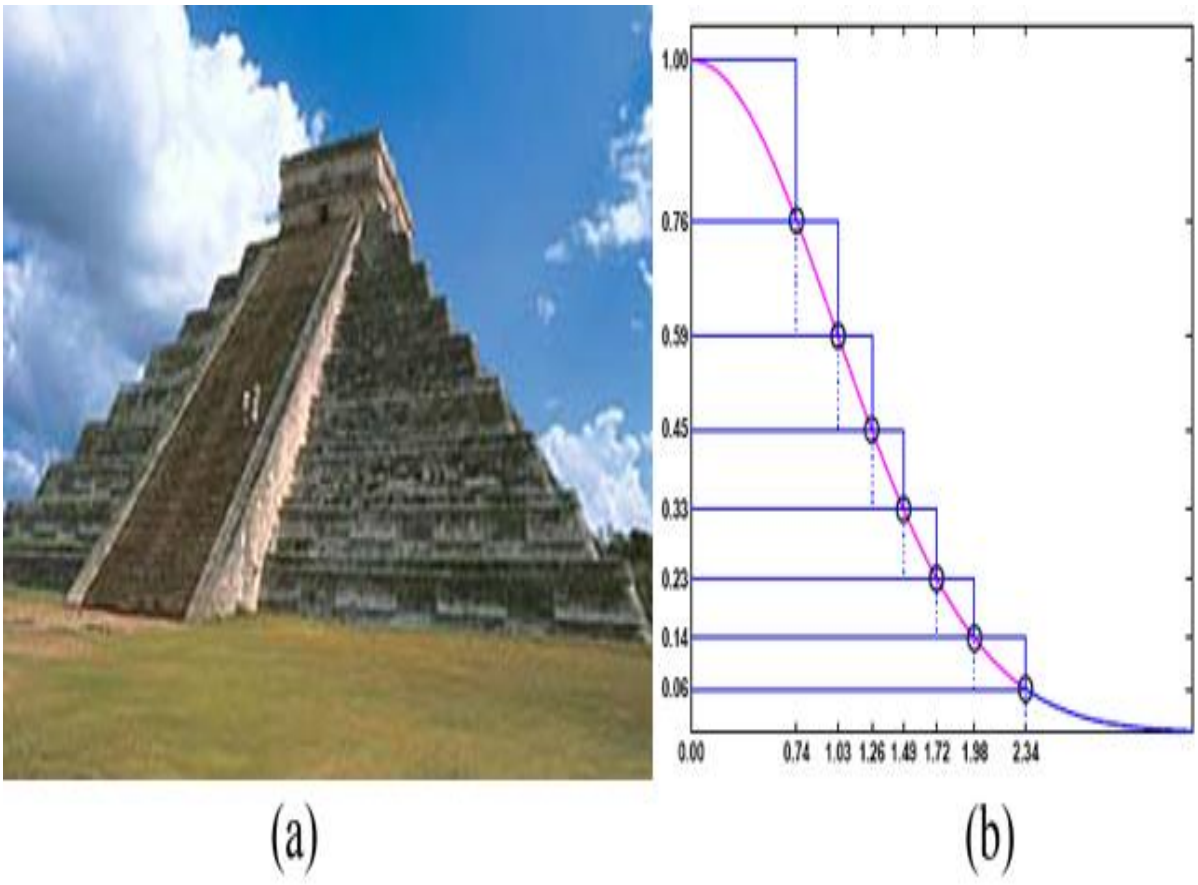

Figure 3.1 : *Schéma de la construction de la Ziggourat : (a) Les temples Ziggourat ont uneconstruction en escalier à laquelle le nom de la méthode Ziggourat réfère, (b) Schéma dela construction en escalier de la méthode Ziggourat. Images tirées de la documentation technique de Matlab [27].*

N	R	V	% efficiency
8	2.3383716982472524	1.7617364011877759e-1	88.93
16	2.6755367657376135	8.3989463747827300e-2	93.26
32	2.9613001212640193	4.0758744432219871e-2	96.09
64	3.2136576271588955	2.0024457157351700e-2	97.80
128	3.4426198558966519	9.9125630353364726e-3	98.78
256	3.8520461503683916	.9286732339746571e-3	99.33
512	3.8520461503683916	2.4567663515413529e-3	99.64

***Tableau 3.1**:Table pré-calcul de la méthode Ziggourat*

Les rectangles et la bande inferieur sont choisis de telle sorte qu'ils soient tous de même aire v et leur bord droit est notéx_i voir (table3.1) .Le rectangle le plus a guauche R_0 est suppose etre vide et $x_0 = 0$. Le pseudo code ci-dessus décrit l'algorithme complet de la méthode de Ziggourat.

Les valeurs de x_i $(i = 1,2,3 \dots (n-1))$ sont necessaires pour les tables dans l'implementation materielle et sont determines en faisant correspondre l'aire de chacun des rectangles a celle de la région de base. Si cette aire est$v$, les équations sont données par:

$$v = x_i[f(x_{i-1}) - f(x_i)] = rf(r) + \int_r^{\infty} f(x)dx$$

3.3 Fonctionnement logicielle de la méthode à virgule fixe

3.3.1 MATLAB

MATLAB « matrix laboratory » est un logiciel de calul matriciel à syntaxe simple avec ses fonctions spécialisées , il peut aussi considéré comme un langage de programmation adapté pour les problèmes scientifiques .Matlab est un interpréteur les instructions sont interpétées et exécutées ligne par ligne ,Il permet de manipulé les matrices et d'afiche les courbes et des données , de mettre en oeuvre les algorithme

➢ L'interface de matlab

- Fenêtrecommande window : permet d'executer les commandes en dehors de programme et affiche les resultats.
- Fenêtre current directory : contenu du répertoire courant ou doit situer vos programme .
- Fenêtre graphique : MATLAB trace les graphiques dans ces fenêtre .
- Fichiers M : Ce sont des programme en langage MATLAB.
- Workespace : Affiche l'ensemble des variables utilisées .
- Commande history : permet de visualiser les dernières commandes exécutées

Pour commancer un nouveux programme en tape sur **HOME** en suite **NEW SCRIPT**

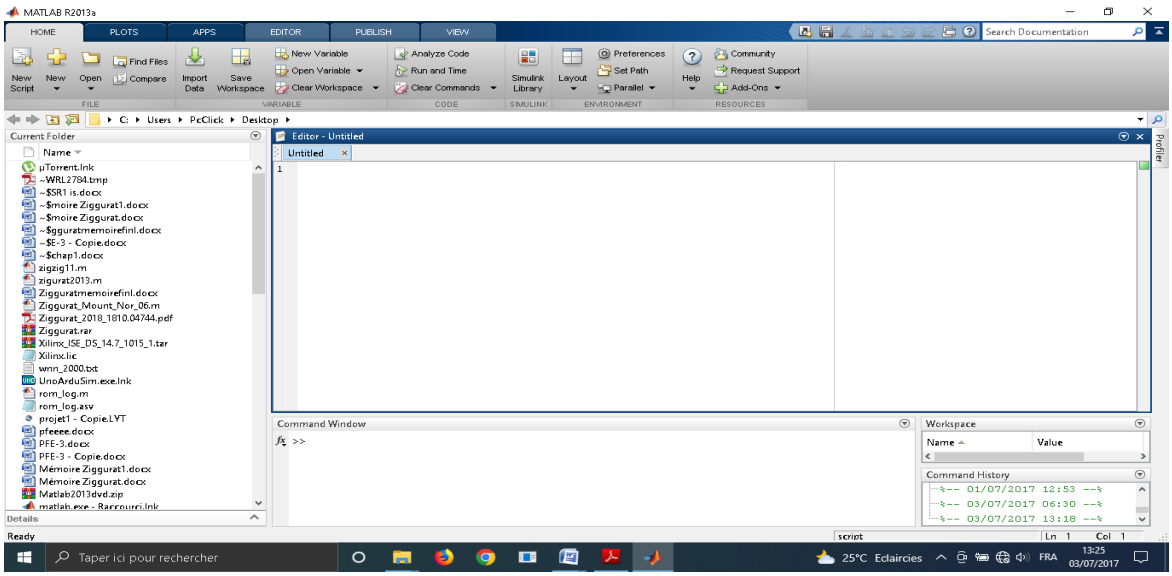

***Figure 3.2**:L'interface de MATLAB*

3.3.2 Interpétation

- **Fonctions**

Les commandes Matlab peuvent toutes être considérées comme des fonctions, c'est-à-dire des entités nommées, qui prennent des paramètres éventuels (arguments) et qui renvoient des résultats (valeurs de retour). Ce sont en quelque sorte des sous-programmes, bien que la notion de programme n'ait pas beaucoup de sens pour un interpréteur de langage comme Matlab.

-La fonction générale f:

$$\mathbf{f}(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2}$$

C'est un fonction exponentielle avec un variable x1 de valeur entre [0 1] , le résultat de fonction f c'est des valeurs aléatoire aussi parce que les valeurs de x1 aléatoire .

-En peut faire un programme Matlab qui fait calcul la fonction f(x1) ,en utilise cet algorithme pour générer un nombre aléatoire n avec des tables donnant une approximation de la densité gaussienne par des rectangles. Ces tables étant pré calculées (table 3.1).

On va faire ces étapes :

-Dans la première étape on prendre le nombre 128 comme une taille d'adresse, on suite en prendre le nombre d'un surface vn =9.91256303526217E-03 et après on établie les vecteurs zéros (n, 1) pour CHAces sorties : sigma fn fn2 wn exp.

-Dans la deuxième étape on à fait une boucle pour rejeté les nombres qui sont à l'extérieur d'une courbe f et accepté juste qu'ils sont à l'intérieur

- Dans la troisième étape on à fait ces commande pour trouver la valeur Max et min pour chaque une des sorties : sigma ,fn,fn2 ,wn,exp aussi on à trouvé la variation log2(Max/min) et on à fait la numirisation pour convertir les valeurs décimale en binaires les valeurs dec2bin(fn3,15) et bin2dec(fs1), pour convertir les valeurs binaires en décimale .

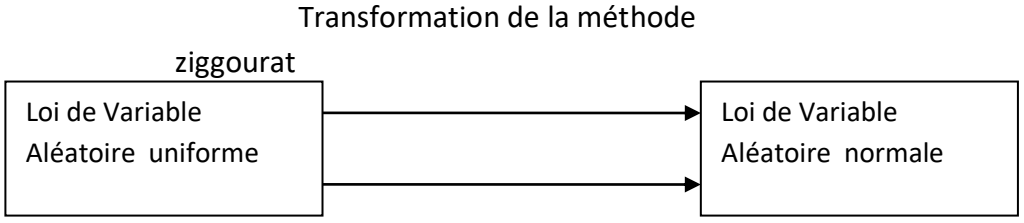

***Figure 3.3** : Plan d'un transformation d'un V.A uniforme en V.A normale*

```
clear all
clc
n=128;
vn = 9.91256303526217E-03; % n=128
dn = 3.442619855899; % n=128
kn = zeros ( n,1); % sigma
fn = zeros ( n,1 );% y
wn = zeros ( n,1 );% x
r=dn;
tn=dn;
q = vn/exp(-0.5*dn*dn);
kn(1,1) = (dn/q)
kn(2,1) = 0;
wn(1,1) = q;
wn(n,1) = dn
fn(1,1) = 1.0;
fn(n,1) = exp(-0.5*r*r)

for    i = (n-1) : -1 : 2
dn = sqrt (-2.0*log(vn/dn + exp(-0.5*dn*dn)))
kn(i+1,1) = (dn/tn);
tn = dn;
fn(i,1) = exp(-0.5*dn*dn)
wn(i,1) = dn
q(i) = vn/exp(-0.5*dn*dn);
end
M0=max(q)
m0=min(q)
t0=log2(M0/m0);
T0=ceil(t0)
i=sort(q,'ascend');
var=(diff(i));
m00=min(var)
t00=log2(M0/m00)
E=ceil(log2(M0/m00))
L=log2(n)
M=max(fn)
m1=fn(127)
t1=log2(M/m1);
T1=ceil(t1)
A=sort(fn,'ascend');
```

```
var=(diff(A));
m11=min(var)
t11=log2(M/m11)
E1=ceil(log2(M/m11))
fn1=fn*2.^6;
fs1=dec2bin(fn1,15);
fb1=bin2dec(fs1);
f11=fb1/2.^6
L=log2(n)
M1=max(kn)
m2=kn(127)
```

```
t2=log2(M1/m2);
T2=ceil(t2)
B=sort(kn,'ascend');
var=(diff(B));
m22=min(var)
t22=log2(M1/m22)
E2=ceil(log2(M1/m22))
fn2=kn/2.^6;
fs2=dec2bin(fn2,15);
fb2=bin2dec(fs2);
f22=fb2*2.^6
L=log2(n)
M2=max(wn)
m3=wn(127)
t3=log2(M2/m3);
T3=ceil(t3)
C=sort(wn,'ascend');
var=(diff(C));
m33=min(var)
t33=log2(M2/m33)
E3=ceil(log2(M2/m33))
fn3=wn*2.^6;
fs3=dec2bin(fn3,15);
fb3=bin2dec(fs3);
f33=fb3/2.^6
```

```
z=wn;
sigma=kn;
m=6;
N=10^m;
```

```
i=0;
jt=0;
jtt=0;
jt2=0;
v=zeros(N,1);

kr=1/(2*(1-cdf('normal',r,0,1)));
```

```
tic
while(i<N)
j=randi([1,n]);
u0=2*rand-1;
x=u0*z(j,1);
sx=sign(x);
if abs(u0)<sigma(j,1)
i=i+1;
v(i,1)=x;
else
if j==1
while ( true )
y=log(rand);
x=-(log(rand))/r;
jt=jt+1;
if (-2*y > x*x )
i=i+1;
jtt=jtt+1;
v(i,1)=sx*(x+r);
break
end
end
else
if (fn(j,1)+rand*(fn(j-1)-fn(j,1))) < exp(-0.5*x*x)
i=i+1;
v(i,1)=x;
else
jt2=jt2+1;
    end
end
    end
   end
t=toc;
% t=toc/60
```

```
Tic
[ch2v1, p1]=chi2gof(v);
t_chi=toc;
[fxb, xb] = hist(v, 500);
gf = 1 / sqrt(2 * pi) * exp(-0.5 * xb .^ 2);
figure(6)
bar(xb, fxb / trapz(xb, fxb), 'FaceColor',[.8 .8 .8],'EdgeColor',[.5 .5 .5]); hold on
plot(xb, gf, 'r');
 hold off
Trejet = 100*(jt2/(N+jt2));
Taccept = 100-Trejet;
```

- Pour réaliser l'autocorrélation on ajoute les commandes suivantes comme une suite de programme précédé :

```
for  i=1:(N-1)
v1s(i,1)=v((i+1),1);
end
v1s(N,1)=v(1,1);
figure (1)
scatter(v1s,v,6,'.') %scatter plot v1
```

○ Numérisation

On faire un programme sur MATLAB pour numérisée les entrées suivants : Sigma, fn, fn2, wn, exp

Fichiers .m

Pour crée un fichier ROM il faut utiliser le vecteurKn en vergule fixe et utilisé un programme Matlab avec les étapes suivants :

- Création de boucle for .
- Convertir les valeur en binaire.
- Stocké les valeurs obtenue dans un fichier texte .

1 er étape

Création de boucle for pour le vecteur Kn avec un

pas =128 et { j=j+1 ;
x(j)=j ;

2 em étape

Après les calculs du vecteur Sigma dans 1 er étape il faut convertir les valeurs décimale e vergule fixe avec dec2bin(Kn) .La petite valeur , que nous avons calculée , kn min est au moins codée sur (01)bit ,pour que toutes les valeurs possibles de Kn eront des codes binaires non nuls.

```
% clear all
clc
fid1= fopen('sigma_2000.txt', 'wt');
fid2= fopen('binsigma_2000.txt', 'wt');
j=0;

for i = 1:128
    j=j+1;
    x(j)=j;

% y = sqrt(-2*log(i));
% z(j)=y;
sg=sigma(i,1)*2^16;
z(j)=sg/2^16;
fprintf(fid1, '"%d" ,  '   , sg);

sgbin=dec2bin(sg,16);
fprintf(fid2, '"%s" ,  ' , sgbin );
end
fclose(fid1);
fclose(fid2);
plot(x,z)
```

3 emétape

Stocké les valeurs obtenue dans un fichier texte (avec un programme Matlab)

Lefichier ROM obtenue en décimale pour Sigma

```
"6.076227e+04" ,  "0" ,  "4.918221e+04" ,  "5.575256e+04" ,
"5.855055e+04" ,  "6.009604e+04" ,  "6.107426e+04" ,  "6.174829e+04" ,
"6.224055e+04" ,  "6.261561e+04" ,  "6.291077e+04" ,  "6.314902e+04" ,
```

```
"6.334532e+04" ,  "6.350981e+04" ,  "6.364960e+04" ,  "6.376985e+04" ,
"6.387435e+04" ,  "6.396599e+04" ,  "6.404698e+04" ,  "6.411907e+04" ,
"6.418361e+04" ,  "6.424173e+04" ,  "6.429432e+04" ,  "6.434211e+04" ,
"6.438573e+04" ,  "6.442568e+04" ,  "6.446239e+04" ,  "6.449623e+04" ,
"6.452751e+04" ,  "6.455649e+04" ,  "6.458342e+04" ,  "6.460848e+04" ,
"6.463187e+04" ,  "6.465371e+04" ,  "6.467417e+04" ,  "6.469334e+04" ,
"6.471134e+04" ,  "6.472827e+04" ,  "6.474420e+04" ,  "6.475920e+04" ,
"6.477336e+04" ,  "6.478672e+04" ,  "6.479935e+04" ,  "6.481128e+04" ,
"6.482257e+04" ,  "6.483326e+04" ,  "6.484338e+04" ,  "6.485296e+04" ,
"6.486204e+04" ,  "6.487064e+04" ,  "6.487879e+04" ,  "6.488652e+04" ,
"6.489384e+04" ,  "6.490077e+04" ,  "6.490733e+04" ,  "6.491353e+04" ,
"6.491940e+04" ,  "6.492494e+04" ,  "6.493018e+04" ,  "6.493510e+04" ,
"6.493974e+04" ,  "6.494410e+04" ,  "6.494818e+04" ,  "6.495200e+04" ,
"6.495556e+04" ,  "6.495887e+04" ,  "6.496192e+04" ,  "6.496474e+04" ,
"6.496732e+04" ,  "6.496966e+04" ,  "6.497177e+04" ,  "6.497365e+04" ,
"6.497530e+04" ,  "6.497672e+04" ,  "6.497791e+04" ,  "6.497888e+04" ,
"6.497961e+04" ,  "6.498012e+04" ,  "6.498039e+04" ,  "6.498042e+04" ,
"6.498021e+04" ,  "6.497975e+04" ,  "6.497905e+04" ,  "6.497808e+04" ,
"6.497685e+04" ,  "6.497535e+04" ,  "6.497356e+04" ,  "6.497147e+04" ,
"6.496908e+04" ,  "6.496637e+04" ,  "6.496332e+04" ,  "6.495991e+04" ,
"6.495612e+04" ,  "6.495194e+04" ,  "6.494734e+04" ,  "6.494229e+04" ,
"6.493675e+04" ,  "6.493069e+04" ,  "6.492407e+04" ,  "6.491685e+04" ,
"6.490897e+04" ,  "6.490037e+04" ,  "6.489098e+04" ,  "6.488072e+04" ,
"6.486950e+04" ,  "6.485722e+04" ,  "6.484374e+04" ,  "6.482893e+04" ,
"6.481259e+04" ,  "6.479454e+04" ,  "6.477450e+04" ,  "6.475218e+04" ,
"6.472720e+04" ,  "6.469908e+04" ,  "6.466723e+04" ,  "6.463090e+04" ,
"6.458910e+04" ,  "6.454050e+04" ,  "6.448333e+04" ,  "6.441510e+04" ,
"6.433220e+04" ,  "6.422926e+04" ,  "6.409775e+04" ,  "6.392341e+04" ,
"6.368012e+04" ,  "6.331409e+04" ,  "6.269226e+04" ,  "6.135679e+04" ,
```

Les resultats binaires obtenu :

```
"1110110101011010" ,  "0000000000000000" ,  "1100000000011110" ,
"1101100111001000" ,  "1110010010110110" ,  "1110101011000000" ,
"1110111010010010" ,  "1111000100110100" ,  "1111001100100000" ,
"1111010010010111" ,  "1111010110111110" ,  "1111011010101101" ,
"1111011101110001" ,  "1111100000010101" ,  "1111100010100001" ,
"1111100100011001" ,  "1111100110000010" ,  "1111100111011101" ,
"1111101000101110" ,  "1111101001110111" ,  "1111101010110111" ,
"1111101011110001" ,  "1111101100100110" ,  "1111101101010110" ,
"1111101110000001" ,  "1111101110101001" ,  "1111101111001110" ,
"1111101111110000" ,  "1111110000001111" ,  "1111110000101100" ,
"1111110001000111" ,  "1111110001100000" ,  "1111110001110111" ,
"1111110010001101" ,  "1111110010100010" ,  "1111110010110101" ,
"1111110011000111" ,  "1111110011011000" ,  "1111110011101000" ,
"1111110011110111" ,  "1111110100000101" ,  "1111110100010010" ,
"1111110100011111" ,  "1111110100101011" ,  "1111110100110110" ,
"1111110101000001" ,  "1111110101001011" ,  "1111110101010100" ,
"1111110101011110" ,  "1111110101100110" ,  "1111110101101110" ,
"1111110101110110" ,  "1111110101111101" ,  "1111110110000100" ,
"1111110110001011" ,  "1111110110010001" ,  "1111110110010111" ,
"1111110110011100" ,  "1111110110100010" ,  "1111110110100111" ,
"1111110110101011" ,  "1111110110110000" ,  "1111110110110100" ,
"1111110110111000" ,  "1111110110111011" ,  "1111110110111110" ,
"1111110111000001" ,  "1111110111000100" ,  "1111110111000111" ,
"1111110111001001" ,  "1111110111001011" ,  "1111110111001101" ,
"1111110111001111" ,  "1111110111010000" ,  "1111110111010001" ,
"1111110111010010" ,  "1111110111010011" ,  "1111110111010100" ,
"1111110111010100" ,  "1111110111010100" ,  "1111110111010100" ,
```

```
"1111110111010011" ,  "1111110111010011" ,  "1111110111010010" ,
"1111110111010000" ,  "1111110111001111" ,  "1111110111001101" ,
"1111110111001011" ,  "1111110111001001" ,  "1111110111000110" ,
"1111110111000011" ,  "1111110110111111" ,  "1111110110111100" ,
"1111110110110111" ,  "1111110110110011" ,  "1111110110101110" ,
"1111110110101000" ,  "1111110110100010" ,  "1111110110011100" ,
"1111110110010100" ,  "1111110110001100" ,  "1111110110000100" ,
"1111110101111010" ,  "1111110101110000" ,  "1111110101100101" ,
"1111110101011001" ,  "1111110101001011" ,  "1111110100111100" ,
"1111110100101100" ,  "1111110100011010" ,  "1111110100000110" ,
"1111110011110000" ,  "1111110011010111" ,  "1111110010111011" ,
"1111110010011011" ,  "1111110001110110" ,  "1111110001001101" ,
"1111110000011100" ,  "1111101111100011" ,  "1111101110011111" ,
"1111101101001100" ,  "1111101011100101" ,  "1111101001100001" ,
"1111100110110011" ,  "1111100011000000" ,
```

Les calculs du vecteur fn :

```
% clear all
clc
fid1= fopen('fn_2000.txt', 'wt');
fid2= fopen('binfn_2000.txt', 'wt');
j=0;

for i = 1:128
    j=j+1;
    x(j)=j;


f=fn(j,1)*2^16;
z(j)=f/2^16;
fprintf(fid1, '"%d" ,  '    , f);

fbin=dec2bin(f,16);
fprintf(fid2, '"%s" ,  ' , fbin );
end
fclose(fid1);
fclose(fid2);

plot(x,z)
```

Le fichier ROM obtenue en décimale pour fn :

"65536" , "6.315047e+04" , "6.136022e+04" , "5.983723e+04" , "5.847657e+04" ,
"5.722886e+04" , "5.606609e+04" , "5.497052e+04" , "5.393005e+04" ,
"5.293594e+04" , "5.198165e+04" , "5.106212e+04" , "5.017332e+04" ,
"4.931201e+04" , "4.847549e+04" , "4.766151e+04" , "4.686817e+04" ,
"4.609383e+04" , "4.533709e+04" , "4.459671e+04" , "4.387161e+04" ,
"4.316083e+04" , "4.246352e+04" , "4.177890e+04" , "4.110631e+04" ,

"4.044511e+04" , "3.979474e+04" , "3.915469e+04" , "3.852449e+04" ,
"3.790371e+04" , "3.729195e+04" , "3.668885e+04" , "3.609407e+04" ,
"3.550730e+04" , "3.492824e+04" , "3.435663e+04" , "3.379221e+04" ,
"3.323475e+04" , "3.268402e+04" , "3.213983e+04" , "3.160196e+04" ,
"3.107024e+04" , "3.054450e+04" , "3.002458e+04" , "2.951031e+04" ,
"2.900156e+04" , "2.849819e+04" , "2.800006e+04" , "2.750705e+04" ,
"2.701905e+04" , "2.653595e+04" , "2.605763e+04" , "2.558400e+04" ,
"2.511496e+04" , "2.465041e+04" , "2.419028e+04" , "2.373448e+04" ,
"2.328293e+04" , "2.283556e+04" , "2.239228e+04" , "2.195304e+04" ,
"2.151777e+04" , "2.108640e+04" , "2.065887e+04" , "2.023513e+04" ,
"1.981513e+04" , "1.939880e+04" , "1.898610e+04" , "1.857698e+04" ,
"1.817140e+04" , "1.776931e+04" , "1.737066e+04" , "1.697543e+04" ,
"1.658358e+04" , "1.619506e+04" , "1.580984e+04" , "1.542789e+04" ,
"1.504918e+04" , "1.467369e+04" , "1.430137e+04" , "1.393222e+04" ,
"1.356620e+04" , "1.320328e+04" , "1.284346e+04" , "1.248671e+04" ,
"1.213301e+04" , "1.178234e+04" , "1.143470e+04" , "1.109006e+04" ,
"1.074842e+04" , "1.040976e+04" , "1.007408e+04" , "9.741374e+03" ,
"9.411630e+03" , "9.084848e+03" , "8.761026e+03" , "8.440166e+03" ,
"8.122268e+03" , "7.807340e+03" , "7.495386e+03" , "7.186417e+03" ,
"6.880445e+03" , "6.577484e+03" , "6.277552e+03" , "5.980671e+03" ,
"5.686865e+03" , "5.396162e+03" , "5.108596e+03" , "4.824203e+03" ,
"4.543029e+03" , "4.265121e+03" , "3.990538e+03" , "3.719343e+03" ,
"3.451611e+03" , "3.187428e+03" , "2.926894e+03" , "2.670125e+03" ,
"2.417255e+03" , "2.168448e+03" , "1.923896e+03" , "1.683836e+03" ,
"1.448562e+03" , "1.218452e+03" , "9.940040e+02" , "7.759121e+02" ,
"5.652142e+02" , "3.636590e+02" , "1.749568e+02" ,

Le fichier ROM obtenue en binaires pour fn :

"10000000000000000" , "1111011010101110" , "1110111110110000" ,
"1110100110111101" , "1110010001101100" , "1101111110001100" ,
"1101101100000010" , "1101011010111010" , "1101001010101010" ,
"1100111011000111" , "1100101100001101" , "1100011101110110" ,
"1100001111111101" , "1100000010100000" , "1011110101011011" ,
"1011101000101101" , "1011011100010100" , "1011010000001101" ,
"1011000100011001" , "1010111000110100" , "1010101101011111" ,
"1010100010011000" , "1010010111011111" , "1010001100110010" ,
"1010000010010010" , "1001110111111101" , "1001101101110010" ,
"1001100011110010" , "1001011001111100" , "1001010000001111" ,
"1001000110101011" , "1000111101010000" , "1000110011111110" ,
"1000101010110011" , "1000100001110000" , "1000011000110100" ,
"1000010000000000" , "1000000111010010" , "0111111110101100" ,

"0111110110001011", "0111101101110001", "0111100101011110",
"0111011101010000", "0111010101001000", "0111001101000110",
"0111000101001001", "0110111101010010", "0110110101100000",
"0110101101110011", "0110100110001011", "0110011110100111",
"0110010111001001", "0110001111101111", "0110001000011010",
"0110000001001010", "0101111001111110", "0101110010110110",
"0101101011110010", "0101100100110011", "0101011101111000",
"0101010111000001", "0101010000001101", "0101001001011110",
"0101000010110010", "0100111100001011", "0100110101100111",
"0100101111000110", "0100101000101010", "0100100010010000",
"0100011011111011", "0100010101101001", "0100001111011010",
"0100001001001111", "0100000011000111", "0011111101000011",
"0011110111000001", "0011110001000011", "0011101011001001",
"0011100101010001", "0011011111011101", "0011011001101100",
"0011010011111110", "0011001110010011", "0011001000101011",
"0011000011000110", "0010111101100101", "0010111000000110",
"0010110010101010", "0010101101010010", "0010100111111100",
"0010100010101001", "0010011101011010", "0010011000001101",
"0010010011000011", "0010001101111100", "0010001000111001",
"0010000011111000", "0001111110111010", "0001111001111111",
"0001110101000111", "0001110000010010", "0001101011100000",
"0001100110110001", "0001100010000101", "0001011101011100",
"0001011000110110", "0001010100010100", "0001001111110100",
"0001001011011000", "0001000110111111", "0001000010101001",
"0000111110010110", "0000111010000111", "0000110101111011",
"0000110001110011", "0000101101101110", "0000101001101110",
"0000100101110001", "0000100001111000", "0000011110000011",
"0000011010010011", "0000010110101000", "0000010011000010",
"0000001111100010", "0000001100000111", "0000001000110101",
"0000000101101011", "0000000010101110",

Les calculs du vecteur fn2 :

```
% clear all
clc
fid1= fopen('fn2_2000.txt', 'wt');
fid2= fopen('binfn2_2000.txt', 'wt');
j=2;

for i = 1:127
    j=i+1;
    x(i)=i;

h=(fn(j-1)-fn(j,1))*2^16;
z2(i)=h/2^16;
fprintf(fid1, '"%d" ,  '    ,h);
```

```
hbin=dec2bin(h,16);
fprintf(fid2, '"%s" ,  ' , hbin );
end
fclose(fid1);
fclose(fid2);

plot(x,z2)
```

Le fichier ROM obtenue en décimale pour fn2 :

"2.385531e+03" , "1.790248e+03" , "1.522993e+03" , "1.360658e+03" ,
"1.247714e+03" , "1.162769e+03" , "1.095565e+03" , "1.040475e+03" ,
"9.941101e+02" , "9.542882e+02" , "9.195308e+02" , "8.887935e+02" ,
"8.613144e+02" , "8.365222e+02" , "8.139784e+02" , "7.933402e+02" ,
"7.743346e+02" , "7.567413e+02" , "7.403800e+02" , "7.251017e+02" ,
"7.107817e+02" , "6.973148e+02" , "6.846116e+02" , "6.725954e+02" ,
"6.612002e+02" , "6.503684e+02" , "6.400499e+02" , "6.302005e+02" ,
"6.207815e+02" , "6.117582e+02" , "6.031002e+02" , "5.947798e+02" ,
"5.867725e+02" , "5.790561e+02" , "5.716107e+02" , "5.644179e+02" ,
"5.574615e+02" , "5.507262e+02" , "5.441985e+02" , "5.378657e+02" ,
"5.317162e+02" , "5.257395e+02" , "5.199257e+02" , "5.142657e+02" ,
"5.087513e+02" , "5.033745e+02" , "4.981281e+02" , "4.930055e+02" ,
"4.880002e+02" , "4.831064e+02" , "4.783187e+02" , "4.736318e+02" ,
"4.690410e+02" , "4.645416e+02" , "4.601293e+02" , "4.558001e+02" ,
"4.515503e+02" , "4.473761e+02" , "4.432741e+02" , "4.392411e+02" ,
"4.352741e+02" , "4.313699e+02" , "4.275259e+02" , "4.237394e+02" ,
"4.200078e+02" , "4.163286e+02" , "4.126996e+02" , "4.091185e+02" ,
"4.055830e+02" , "4.020911e+02" , "3.986409e+02" , "3.952302e+02" ,
"3.918574e+02" , "3.885204e+02" , "3.852176e+02" , "3.819472e+02" ,
"3.787074e+02" , "3.754968e+02" , "3.723135e+02" , "3.691560e+02" ,
"3.660227e+02" , "3.629121e+02" , "3.598226e+02" , "3.567526e+02" ,
"3.537006e+02" , "3.506651e+02" , "3.476445e+02" , "3.446372e+02" ,
"3.416416e+02" , "3.386562e+02" , "3.356793e+02" , "3.327091e+02" ,
"3.297440e+02" , "3.267822e+02" , "3.238217e+02" , "3.208607e+02" ,
"3.178972e+02" , "3.149289e+02" , "3.119536e+02" , "3.089689e+02" ,
"3.059722e+02" , "3.029608e+02" , "2.999315e+02" , "2.968812e+02" ,
"2.938063e+02" , "2.907029e+02" , "2.875664e+02" , "2.843922e+02" ,
"2.811746e+02" , "2.779075e+02" , "2.745837e+02" , "2.711950e+02" ,
"2.677317e+02" , "2.641826e+02" , "2.605340e+02" , "2.567697e+02" ,
"2.528693e+02" , "2.488076e+02" , "2.445521e+02" , "2.400600e+02" ,
"2.352734e+02" , "2.301101e+02" , "2.244480e+02" , "2.180920e+02" ,
"2.106979e+02" , "2.015553e+02" , "1.887021e+02" ,

Le fichier ROM obtenue en binaires pour fn2 :

```
"0000100101010001", "0000011011111110", "0000010111110010",
"0000010101010000", "0000010011011111", "0000010010001010",
"0000010001000111", "0000010000010000", "0000001111100010",
"0000001110111010", "0000001110010111", "0000001101111000",
"0000001101011101", "0000001101000100", "0000001100101101",
"0000001100011001", "0000001100000110", "0000001011110100",
"0000001011100100", "0000001011010101", "0000001011000110",
"0000001010111001", "0000001010101100", "0000001010100000",
"0000001010010101", "0000001010001010", "0000001010000000",
"0000001001110110", "0000001001101100", "0000001001100011",
"0000001001011011", "0000001001010010", "0000001001001010",
"0000001001000011", "0000001000111011", "0000001000110100",
"0000001000101101", "0000001000100110", "0000001000100000",
"0000001000011001", "0000001000010011", "0000001000001101",
"0000001000000111", "0000001000000010", "0000000111111100",
"0000000111110111", "0000000111110010", "0000000111101101",
"0000000111101000", "0000000111100011", "0000000111011110",
"0000000111011001", "0000000111010101", "0000000111010000",
"0000000111001100", "0000000111000111", "0000000111000011",
"0000000110111111", "0000000110111011", "0000000110110111",
"0000000110110011", "0000000110101111", "0000000110101011",
"0000000110100111", "0000000110100100", "0000000110100000",
"0000000110011100", "0000000110011001", "0000000110010101",
"0000000110010010", "0000000110001110", "0000000110001011",
"0000000110000111", "0000000110000100", "0000000110000001",
"0000000101111101", "0000000101111010", "0000000101110111",
"0000000101110100", "0000000101110001", "0000000101101110",
"0000000101101010", "0000000101100111", "0000000101100100",
"0000000101100001", "0000000101011110", "0000000101011011",
"0000000101011000", "0000000101010101", "0000000101010010",
"0000000101001111", "0000000101001100", "0000000101001001",
"0000000101000110", "0000000101000011", "0000000101000000",
"0000000100111101", "0000000100111010", "0000000100110111",
"0000000100110100", "0000000100110001", "0000000100101110",
"0000000100101011", "0000000100101000", "0000000100100101",
"0000000100100010", "0000000100011111", "0000000100011100",
"0000000100011001", "0000000100010101", "0000000100010010",
"0000000100001111", "0000000100001011", "0000000100001000",
"0000000100000100", "0000000100000000", "0000000011111100",
"0000000011111000", "0000000011110100", "0000000011110000",
```

"0000000011101011", "0000000011100110", "0000000011100000",
"0000000011011010", "0000000011010010", "0000000011001001",
"0000000010111100",

Les calculs du vecteur exp :

```
% clear all
clc
fid1= fopen('exp_2000.txt', 'wt');
fid2= fopen('binexp_2000.txt', 'wt');
j=0;

for i = 1:1024
    pas=1/2^10;
    x=pas*i;
    j=j+1;
%     x(j)=j;
%     r = 3.442619855899;
% y = sqrt(-2*log(i));
% z(j)=y;
o(i,1)=exp(-0.5*x^2);
t=o(i,1)*2^16;
z(j)=t/2^16;
fprintf(fid1, '"%d" ,  '   , t);

tbin=dec2bin(t,16);
fprintf(fid2, '"%s" ,  ' , tbin );
end
fclose(fid1);
fclose(fid2);

plot(x,z)
```

Le fichier ROM obtenue en décimale pour exp :

```
"6.553597e+04" ,  "6.553588e+04" ,  "6.553572e+04" ,  "6.553550e+04" ,
"6.553522e+04" ,  "6.553488e+04" ,  "6.553447e+04" ,  "6.553400e+04" ,
"6.553347e+04" ,  "6.553288e+04" ,  "6.553222e+04" ,  "6.553150e+04" ,
"6.553072e+04" ,  "6.552988e+04" ,  "6.552897e+04" ,  "6.552800e+04" ,
"6.552697e+04" ,  "6.552588e+04" ,  "6.552472e+04" ,  "6.552350e+04" ,
"6.552222e+04" ,  "6.552088e+04" ,  "6.551947e+04" ,  "6.551800e+04" ,
"6.551647e+04" ,  "6.551488e+04" ,  "6.551322e+04" ,  "6.551150e+04" ,
"6.550972e+04" ,  "6.550788e+04" ,  "6.550598e+04" ,  "6.550401e+04" ,
"6.550198e+04" ,  "6.549988e+04" ,  "6.549773e+04" ,  "6.549551e+04" ,
"6.549323e+04" ,  "6.549089e+04" ,  "6.548849e+04" ,  "6.548602e+04" ,
"6.548349e+04" ,  "6.548090e+04" ,  "6.547824e+04" ,  "6.547553e+04" ,
"6.547275e+04" ,  "6.546991e+04" ,  "6.546701e+04" ,  "6.546404e+04" ,
"6.546101e+04" ,  "6.545792e+04" ,  "6.545477e+04" ,  "6.545155e+04" ,
"6.544828e+04" ,  "6.544494e+04" ,  "6.544154e+04" ,  "6.543807e+04" ,
"6.543455e+04" ,  "6.543096e+04" ,  "6.542731e+04" ,  "6.542360e+04" ,
"6.541982e+04" ,  "6.541599e+04" ,  "6.541209e+04" ,  "6.540812e+04" ,
```

"6.540410e+04" , "6.540002e+04" , "6.539587e+04" , "6.539166e+04" ,
"6.538739e+04" , "6.538305e+04" , "6.537866e+04" , "6.537420e+04" ,
"6.536968e+04" , "6.536510e+04" , "6.536045e+04" , "6.535575e+04" ,
"6.535098e+04" , "6.534615e+04" , "6.534126e+04" , "6.533630e+04" ,
"6.533129e+04" , "6.532621e+04" , "6.532107e+04" , "6.531587e+04" ,
"6.531061e+04" , "6.530528e+04" , "6.529990e+04" , "6.529445e+04" ,
"6.528894e+04" , "6.528336e+04" , "6.527773e+04" , "6.527203e+04" ,
"6.526628e+04" , "6.526046e+04" , "6.525457e+04" , "6.524863e+04" ,
"6.524263e+04" , "6.523656e+04" , "6.523043e+04" , "6.522424e+04" ,
"6.521799e+04" , "6.521168e+04" , "6.520531e+04" , "6.519887e+04" ,
"6.519237e+04" , "6.518581e+04" , "6.517919e+04" , "6.517251e+04" ,
"6.516577e+04" , "6.515896e+04" , "6.515210e+04" , "6.514517e+04" ,
"6.513818e+04" , "6.513113e+04" , "6.512402e+04" , "6.511685e+04" ,
"6.510961e+04" , "6.510232e+04" , "6.509496e+04" , "6.508754e+04" ,
"6.508006e+04" , "6.507252e+04" , "6.506492e+04" , "6.505726e+04" ,
"6.504953e+04" , "6.504175e+04" , "6.503390e+04" , "6.502599e+04" ,
"6.501803e+04" , "6.501000e+04" , "6.500191e+04" , "6.499376e+04" ,
"6.498554e+04" , "6.497727e+04" , "6.496894e+04" , "6.496054e+04" ,
"6.495209e+04" , "6.494357e+04" , "6.493499e+04" , "6.492635e+04" ,
"6.491765e+04" , "6.490889e+04" , "6.490007e+04" , "6.489119e+04" ,
"6.488225e+04" , "6.487325e+04" , "6.486419e+04" , "6.485506e+04" ,
"6.484588e+04" , "6.483663e+04" , "6.482733e+04" , "6.481796e+04" ,
"6.480854e+04" , "6.479905e+04" , "6.478950e+04" , "6.477990e+04" ,
"6.477023e+04" , "6.476050e+04" , "6.475071e+04" , "6.474086e+04" ,
"6.473095e+04" , "6.472099e+04" , "6.471096e+04" , "6.470087e+04" ,
"6.469072e+04" , "6.468051e+04" , "6.467024e+04" , "6.465991e+04" ,
"6.464952e+04" , "6.463907e+04" , "6.462856e+04" , "6.461799e+04" ,
"6.460736e+04" , "6.459667e+04" , "6.458592e+04" , "6.457511e+04" ,
"6.456425e+04" , "6.455332e+04" , "6.454233e+04" , "6.453128e+04" ,
"6.452017e+04" , "6.450901e+04" , "6.449778e+04" , "6.448649e+04" ,
"6.447515e+04" , "6.446374e+04" , "6.445228e+04" , "6.444076e+04" ,
"6.442917e+04" , "6.441753e+04" , "6.440583e+04" , "6.439407e+04" ,
"6.438225e+04" , "6.437037e+04" , "6.435843e+04" , "6.434643e+04" ,
"6.433437e+04" , "6.432226e+04" , "6.431008e+04" , "6.429785e+04" ,
"6.428555e+04" , "6.427320e+04" , "6.426079e+04" , "6.424832e+04" ,
"6.423579e+04" , "6.422320e+04" , "6.421056e+04" , "6.419785e+04" ,
"6.418509e+04" , "6.417226e+04" , "6.415938e+04" , "6.414644e+04" ,
"6.413344e+04" , "6.412039e+04" , "6.410727e+04" , "6.409410e+04" ,
"6.408087e+04" , "6.406758e+04" , "6.405423e+04" , "6.404082e+04" ,
"6.402735e+04" , "6.401383e+04" , "6.400025e+04" , "6.398661e+04" ,
"6.397291e+04" , "6.395915e+04" , "6.394534e+04" , "6.393147e+04" ,
"6.391754e+04" , "6.390355e+04" , "6.388950e+04" , "6.387540e+04" ,
"6.386124e+04" , "6.384702e+04" , "6.383274e+04" , "6.381841e+04" ,
"6.380402e+04" , "6.378957e+04" , "6.377506e+04" , "6.376049e+04" ,
"6.374587e+04" , "6.373119e+04" , "6.371646e+04" , "6.370166e+04" ,
"6.368681e+04" , "6.367190e+04" , "6.365693e+04" , "6.364191e+04" ,
"6.362683e+04" , "6.361169e+04" , "6.359650e+04" , "6.358125e+04" ,
"6.356594e+04" , "6.355057e+04" , "6.353515e+04" , "6.351967e+04" ,
"6.350413e+04" , "6.348854e+04" , "6.347289e+04" , "6.345718e+04" ,
"6.344142e+04" , "6.342560e+04" , "6.340973e+04" , "6.339379e+04" ,
"6.337780e+04" , "6.336176e+04" , "6.334566e+04" , "6.332950e+04" ,
"6.331329e+04" , "6.329702e+04" , "6.328069e+04" , "6.326431e+04" ,
"6.324787e+04" , "6.323137e+04" , "6.321482e+04" , "6.319822e+04" ,
"6.318155e+04" , "6.316483e+04" , "6.314806e+04" , "6.313123e+04" ,
"6.311434e+04" , "6.309740e+04" , "6.308041e+04" , "6.306335e+04" ,
"6.304625e+04" , "6.302908e+04" , "6.301186e+04" , "6.299459e+04" ,
"6.297726e+04" , "6.295987e+04" , "6.294243e+04" , "6.292494e+04" ,
"6.290739e+04" , "6.288978e+04" , "6.287212e+04" , "6.285441e+04" ,
"6.283664e+04" , "6.281881e+04" , "6.280093e+04" , "6.278300e+04" ,
"6.276501e+04" , "6.274696e+04" , "6.272886e+04" , "6.271071e+04" ,
"6.269250e+04" , "6.267424e+04" , "6.265592e+04" , "6.263755e+04" ,

"6.261912e+04" , "6.260064e+04" , "6.258211e+04" , "6.256352e+04" ,
"6.254488e+04" , "6.252618e+04" , "6.250743e+04" , "6.248863e+04" ,
"6.246977e+04" , "6.245086e+04" , "6.243189e+04" , "6.241287e+04" ,
"6.239380e+04" , "6.237467e+04" , "6.235549e+04" , "6.233625e+04" ,
"6.231696e+04" , "6.229762e+04" , "6.227823e+04" , "6.225878e+04" ,
"6.223928e+04" , "6.221972e+04" , "6.220012e+04" , "6.218045e+04" ,
"6.216074e+04" , "6.214097e+04" , "6.212115e+04" , "6.210128e+04" ,
"6.208135e+04" , "6.206138e+04" , "6.204134e+04" , "6.202126e+04" ,
"6.200112e+04" , "6.198093e+04" , "6.196069e+04" , "6.194040e+04" ,
"6.192005e+04" , "6.189965e+04" , "6.187920e+04" , "6.185870e+04" ,
"6.183814e+04" , "6.181753e+04" , "6.179687e+04" , "6.177616e+04" ,
"6.175540e+04" , "6.173458e+04" , "6.171372e+04" , "6.169280e+04" ,
"6.167183e+04" , "6.165080e+04" , "6.162973e+04" , "6.160860e+04" ,
"6.158743e+04" , "6.156620e+04" , "6.154492e+04" , "6.152358e+04" ,
"6.150220e+04" , "6.148077e+04" , "6.145928e+04" , "6.143775e+04" ,
"6.141616e+04" , "6.139452e+04" , "6.137283e+04" , "6.135109e+04" ,
"6.132930e+04" , "6.130746e+04" , "6.128557e+04" , "6.126363e+04" ,
"6.124163e+04" , "6.121959e+04" , "6.119749e+04" , "6.117535e+04" ,
"6.115315e+04" , "6.113091e+04" , "6.110861e+04" , "6.108627e+04" ,
"6.106387e+04" , "6.104143e+04" , "6.101893e+04" , "6.099639e+04" ,
"6.097379e+04" , "6.095115e+04" , "6.092845e+04" , "6.090571e+04" ,
"6.088291e+04" , "6.086007e+04" , "6.083718e+04" , "6.081424e+04" ,
"6.079125e+04" , "6.076820e+04" , "6.074511e+04" , "6.072198e+04" ,
"6.069879e+04" , "6.067555e+04" , "6.065226e+04" , "6.062893e+04" ,
"6.060555e+04" , "6.058211e+04" , "6.055863e+04" , "6.053510e+04" ,
"6.051152e+04" , "6.048790e+04" , "6.046422e+04" , "6.044050e+04" ,
"6.041673e+04" , "6.039290e+04" , "6.036904e+04" , "6.034512e+04" ,
"6.032116e+04" , "6.029714e+04" , "6.027308e+04" , "6.024897e+04" ,
"6.022482e+04" , "6.020061e+04" , "6.017636e+04" , "6.015206e+04" ,
"6.012772e+04" , "6.010332e+04" , "6.007888e+04" , "6.005439e+04" ,
"6.002986e+04" , "6.000527e+04" , "5.998064e+04" , "5.995596e+04" ,
"5.993124e+04" , "5.990647e+04" , "5.988165e+04" , "5.985678e+04" ,
"5.983187e+04" , "5.980691e+04" , "5.978191e+04" , "5.975686e+04" ,
"5.973176e+04" , "5.970661e+04" , "5.968142e+04" , "5.965619e+04" ,
"5.963090e+04" , "5.960557e+04" , "5.958020e+04" , "5.955478e+04" ,
"5.952931e+04" , "5.950379e+04" , "5.947824e+04" , "5.945263e+04" ,
"5.942698e+04" , "5.940128e+04" , "5.937554e+04" , "5.934976e+04" ,
"5.932392e+04" , "5.929805e+04" , "5.927212e+04" , "5.924615e+04" ,
"5.922014e+04" , "5.919408e+04" , "5.916798e+04" , "5.914183e+04" ,
"5.911564e+04" , "5.908940e+04" , "5.906312e+04" , "5.903679e+04" ,
"5.901042e+04" , "5.898400e+04" , "5.895754e+04" , "5.893104e+04" ,
"5.890449e+04" , "5.887790e+04" , "5.885126e+04" , "5.882458e+04" ,
"5.879785e+04" , "5.877108e+04" , "5.874427e+04" , "5.871741e+04" ,
"5.869051e+04" , "5.866357e+04" , "5.863658e+04" , "5.860955e+04" ,
"5.858247e+04" , "5.855536e+04" , "5.852820e+04" , "5.850099e+04" ,
"5.847374e+04" , "5.844645e+04" , "5.841912e+04" , "5.839174e+04" ,
"5.836432e+04" , "5.833686e+04" , "5.830936e+04" , "5.828181e+04" ,
"5.825422e+04" , "5.822659e+04" , "5.819891e+04" , "5.817120e+04" ,
"5.814344e+04" , "5.811564e+04" , "5.808779e+04" , "5.805991e+04" ,
"5.803198e+04" , "5.800401e+04" , "5.797600e+04" , "5.794794e+04" ,
"5.791985e+04" , "5.789171e+04" , "5.786354e+04" , "5.783532e+04" ,
"5.780706e+04" , "5.777875e+04" , "5.775041e+04" , "5.772203e+04" ,
"5.769360e+04" , "5.766514e+04" , "5.763663e+04" , "5.760808e+04" ,
"5.757949e+04" , "5.755086e+04" , "5.752219e+04" , "5.749348e+04" ,
"5.746473e+04" , "5.743594e+04" , "5.740711e+04" , "5.737823e+04" ,
"5.734932e+04" , "5.732037e+04" , "5.729138e+04" , "5.726234e+04" ,
"5.723327e+04" , "5.720416e+04" , "5.717501e+04" , "5.714582e+04" ,
"5.711659e+04" , "5.708732e+04" , "5.705801e+04" , "5.702866e+04" ,
"5.699927e+04" , "5.696984e+04" , "5.694037e+04" , "5.691087e+04" ,
"5.688132e+04" , "5.685174e+04" , "5.682212e+04" , "5.679246e+04" ,
"5.676276e+04" , "5.673302e+04" , "5.670324e+04" , "5.667343e+04" ,

"5.664357e+04" , "5.661368e+04" , "5.658375e+04" , "5.655378e+04" ,
"5.652378e+04" , "5.649373e+04" , "5.646365e+04" , "5.643353e+04" ,
"5.640337e+04" , "5.637318e+04" , "5.634294e+04" , "5.631267e+04" ,
"5.628237e+04" , "5.625202e+04" , "5.622164e+04" , "5.619122e+04" ,
"5.616076e+04" , "5.613027e+04" , "5.609974e+04" , "5.606917e+04" ,
"5.603857e+04" , "5.600793e+04" , "5.597725e+04" , "5.594653e+04" ,
"5.591578e+04" , "5.588500e+04" , "5.585417e+04" , "5.582331e+04" ,
"5.579242e+04" , "5.576149e+04" , "5.573052e+04" , "5.569951e+04" ,
"5.566848e+04" , "5.563740e+04" , "5.560629e+04" , "5.557514e+04" ,
"5.554396e+04" , "5.551274e+04" , "5.548149e+04" , "5.545020e+04" ,
"5.541888e+04" , "5.538752e+04" , "5.535613e+04" , "5.532470e+04" ,
"5.529323e+04" , "5.526174e+04" , "5.523020e+04" , "5.519864e+04" ,
"5.516703e+04" , "5.513540e+04" , "5.510373e+04" , "5.507202e+04" ,
"5.504028e+04" , "5.500851e+04" , "5.497670e+04" , "5.494486e+04" ,
"5.491298e+04" , "5.488107e+04" , "5.484913e+04" , "5.481715e+04" ,
"5.478514e+04" , "5.475310e+04" , "5.472102e+04" , "5.468891e+04" ,
"5.465676e+04" , "5.462458e+04" , "5.459237e+04" , "5.456013e+04" ,
"5.452785e+04" , "5.449554e+04" , "5.446320e+04" , "5.443083e+04" ,
"5.439842e+04" , "5.436598e+04" , "5.433351e+04" , "5.430100e+04" ,
"5.426846e+04" , "5.423589e+04" , "5.420329e+04" , "5.417066e+04" ,
"5.413799e+04" , "5.410529e+04" , "5.407257e+04" , "5.403980e+04" ,
"5.400701e+04" , "5.397419e+04" , "5.394133e+04" , "5.390844e+04" ,
"5.387552e+04" , "5.384257e+04" , "5.380959e+04" , "5.377658e+04" ,
"5.374354e+04" , "5.371046e+04" , "5.367736e+04" , "5.364422e+04" ,
"5.361106e+04" , "5.357786e+04" , "5.354463e+04" , "5.351137e+04" ,
"5.347809e+04" , "5.344477e+04" , "5.341142e+04" , "5.337804e+04" ,
"5.334463e+04" , "5.331119e+04" , "5.327772e+04" , "5.324422e+04" ,
"5.321070e+04" , "5.317714e+04" , "5.314355e+04" , "5.310994e+04" ,
"5.307629e+04" , "5.304261e+04" , "5.300891e+04" , "5.297518e+04" ,
"5.294141e+04" , "5.290762e+04" , "5.287380e+04" , "5.283995e+04" ,
"5.280607e+04" , "5.277217e+04" , "5.273823e+04" , "5.270427e+04" ,
"5.267028e+04" , "5.263626e+04" , "5.260221e+04" , "5.256813e+04" ,
"5.253403e+04" , "5.249990e+04" , "5.246574e+04" , "5.243155e+04" ,
"5.239733e+04" , "5.236309e+04" , "5.232882e+04" , "5.229452e+04" ,
"5.226020e+04" , "5.222584e+04" , "5.219146e+04" , "5.215706e+04" ,
"5.212262e+04" , "5.208816e+04" , "5.205367e+04" , "5.201916e+04" ,
"5.198462e+04" , "5.195005e+04" , "5.191545e+04" , "5.188083e+04" ,
"5.184618e+04" , "5.181151e+04" , "5.177681e+04" , "5.174209e+04" ,
"5.170733e+04" , "5.167256e+04" , "5.163775e+04" , "5.160292e+04" ,
"5.156807e+04" , "5.153319e+04" , "5.149828e+04" , "5.146335e+04" ,
"5.142839e+04" , "5.139341e+04" , "5.135840e+04" , "5.132337e+04" ,
"5.128831e+04" , "5.125323e+04" , "5.121812e+04" , "5.118299e+04" ,
"5.114783e+04" , "5.111265e+04" , "5.107744e+04" , "5.104221e+04" ,
"5.100696e+04" , "5.097168e+04" , "5.093638e+04" , "5.090105e+04" ,
"5.086570e+04" , "5.083032e+04" , "5.079492e+04" , "5.075950e+04" ,
"5.072405e+04" , "5.068858e+04" , "5.065309e+04" , "5.061757e+04" ,
"5.058203e+04" , "5.054647e+04" , "5.051088e+04" , "5.047527e+04" ,
"5.043964e+04" , "5.040398e+04" , "5.036831e+04" , "5.033260e+04" ,
"5.029688e+04" , "5.026113e+04" , "5.022537e+04" , "5.018957e+04" ,
"5.015376e+04" , "5.011792e+04" , "5.008207e+04" , "5.004619e+04" ,
"5.001028e+04" , "4.997436e+04" , "4.993841e+04" , "4.990245e+04" ,
"4.986646e+04" , "4.983045e+04" , "4.979441e+04" , "4.975836e+04" ,
"4.972228e+04" , "4.968619e+04" , "4.965007e+04" , "4.961393e+04" ,
"4.957777e+04" , "4.954159e+04" , "4.950539e+04" , "4.946917e+04" ,
"4.943293e+04" , "4.939666e+04" , "4.936038e+04" , "4.932407e+04" ,
"4.928775e+04" , "4.925141e+04" , "4.921504e+04" , "4.917866e+04" ,
"4.914225e+04" , "4.910583e+04" , "4.906938e+04" , "4.903292e+04" ,
"4.899643e+04" , "4.895993e+04" , "4.892341e+04" , "4.888687e+04" ,
"4.885031e+04" , "4.881372e+04" , "4.877713e+04" , "4.874051e+04" ,
"4.870387e+04" , "4.866721e+04" , "4.863054e+04" , "4.859384e+04" ,
"4.855713e+04" , "4.852040e+04" , "4.848365e+04" , "4.844688e+04" ,

```
"4.841009e+04" ,  "4.837329e+04" ,  "4.833647e+04" ,  "4.829963e+04" ,
"4.826277e+04" ,  "4.822589e+04" ,  "4.818900e+04" ,  "4.815209e+04" ,
"4.811516e+04" ,  "4.807821e+04" ,  "4.804124e+04" ,  "4.800426e+04" ,
"4.796726e+04" ,  "4.793025e+04" ,  "4.789321e+04" ,  "4.785616e+04" ,
"4.781909e+04" ,  "4.778201e+04" ,  "4.774491e+04" ,  "4.770779e+04" ,
"4.767066e+04" ,  "4.763351e+04" ,  "4.759634e+04" ,  "4.755916e+04" ,
"4.752196e+04" ,  "4.748474e+04" ,  "4.744751e+04" ,  "4.741026e+04" ,
"4.737299e+04" ,  "4.733571e+04" ,  "4.729842e+04" ,  "4.726111e+04" ,
"4.722378e+04" ,  "4.718644e+04" ,  "4.714908e+04" ,  "4.711171e+04" ,
"4.707432e+04" ,  "4.703691e+04" ,  "4.699949e+04" ,  "4.696206e+04" ,
"4.692461e+04" ,  "4.688715e+04" ,  "4.684967e+04" ,  "4.681217e+04" ,
"4.677467e+04" ,  "4.673714e+04" ,  "4.669961e+04" ,  "4.666206e+04" ,
"4.662449e+04" ,  "4.658691e+04" ,  "4.654932e+04" ,  "4.651171e+04" ,
"4.647409e+04" ,  "4.643645e+04" ,  "4.639880e+04" ,  "4.636114e+04" ,
"4.632346e+04" ,  "4.628577e+04" ,  "4.624807e+04" ,  "4.621035e+04" ,
"4.617262e+04" ,  "4.613488e+04" ,  "4.609712e+04" ,  "4.605935e+04" ,
"4.602157e+04" ,  "4.598378e+04" ,  "4.594597e+04" ,  "4.590815e+04" ,
"4.587031e+04" ,  "4.583247e+04" ,  "4.579461e+04" ,  "4.575674e+04" ,
"4.571885e+04" ,  "4.568096e+04" ,  "4.564305e+04" ,  "4.560513e+04" ,
"4.556720e+04" ,  "4.552926e+04" ,  "4.549130e+04" ,  "4.545334e+04" ,
"4.541536e+04" ,  "4.537737e+04" ,  "4.533937e+04" ,  "4.530135e+04" ,
"4.526333e+04" ,  "4.522529e+04" ,  "4.518725e+04" ,  "4.514919e+04" ,
"4.511112e+04" ,  "4.507304e+04" ,  "4.503495e+04" ,  "4.499685e+04" ,
"4.495874e+04" ,  "4.492062e+04" ,  "4.488249e+04" ,  "4.484434e+04" ,
"4.480619e+04" ,  "4.476803e+04" ,  "4.472985e+04" ,  "4.469167e+04" ,
"4.465348e+04" ,  "4.461527e+04" ,  "4.457706e+04" ,  "4.453884e+04" ,
"4.450060e+04" ,  "4.446236e+04" ,  "4.442411e+04" ,  "4.438585e+04" ,
"4.434758e+04" ,  "4.430930e+04" ,  "4.427101e+04" ,  "4.423271e+04" ,
"4.419440e+04" ,  "4.415609e+04" ,  "4.411776e+04" ,  "4.407943e+04" ,
"4.404108e+04" ,  "4.400273e+04" ,  "4.396437e+04" ,  "4.392601e+04" ,
"4.388763e+04" ,  "4.384924e+04" ,  "4.381085e+04" ,  "4.377245e+04" ,
"4.373404e+04" ,  "4.369562e+04" ,  "4.365720e+04" ,  "4.361877e+04" ,
"4.358033e+04" ,  "4.354188e+04" ,  "4.350342e+04" ,  "4.346496e+04" ,
"4.342649e+04" ,  "4.338801e+04" ,  "4.334953e+04" ,  "4.331103e+04" ,
"4.327253e+04" ,  "4.323403e+04" ,  "4.319551e+04" ,  "4.315699e+04" ,
"4.311847e+04" ,  "4.307993e+04" ,  "4.304139e+04" ,  "4.300285e+04" ,
"4.296429e+04" ,  "4.292573e+04" ,  "4.288717e+04" ,  "4.284859e+04" ,
"4.281002e+04" ,  "4.277143e+04" ,  "4.273284e+04" ,  "4.269425e+04" ,
"4.265564e+04" ,  "4.261704e+04" ,  "4.257842e+04" ,  "4.253980e+04" ,
"4.250118e+04" ,  "4.246255e+04" ,  "4.242391e+04" ,  "4.238527e+04" ,
"4.234663e+04" ,  "4.230798e+04" ,  "4.226932e+04" ,  "4.223066e+04" ,
```

Le fichier ROM obtenue en binaires pour exp :

```
"1111111111111111" ,  "1111111111111111" ,  "1111111111111111" ,
"1111111111111111" ,  "1111111111111111" ,  "1111111111111110" ,
"1111111111111110" ,  "1111111111111110" ,  "1111111111111101" ,
"1111111111111100" ,  "1111111111111100" ,  "1111111111111011" ,
"1111111111111010" ,  "1111111111111001" ,  "1111111111111000" ,
"1111111111111000" ,  "1111111111110110" ,  "1111111111110101" ,
"1111111111110100" ,  "1111111111110011" ,  "1111111111110010" ,
"1111111111110000" ,  "1111111111101111" ,  "1111111111101110" ,
"1111111111101100" ,  "1111111111101010" ,  "1111111111101001" ,
"1111111111100111" ,  "1111111111100101" ,  "1111111111100011" ,
"1111111111100001" ,  "1111111111100000" ,  "1111111111011101" ,
"1111111111011011" ,  "1111111111011001" ,  "1111111111010111" ,
"1111111111010101" ,  "1111111111010010" ,  "1111111111010000" ,
"1111111111001110" ,  "1111111111001011" ,  "1111111111001000" ,
"1111111111000110" ,  "1111111111000011" ,  "1111111111000000" ,
```

```
"1111111110111101" ,  "1111111110111011" ,  "1111111110111000" ,
"1111111110110101" ,  "1111111110110001" ,  "1111111110101110" ,
"1111111110101011" ,  "1111111110101000" ,  "1111111110100100" ,
"1111111110100001" ,  "1111111110011110" ,  "1111111110011010" ,
"1111111110010110" ,  "1111111110010011" ,  "1111111110001111" ,
"1111111110001011" ,  "1111111110000111" ,  "1111111110000100" ,
"1111111110000000" ,  "1111111101111100" ,  "1111111101111000" ,
"1111111101110011" ,  "1111111101101111" ,  "1111111101101011" ,
"1111111101100111" ,  "1111111101100010" ,  "1111111101011110" ,
"1111111101011001" ,  "1111111101010101" ,  "1111111101010000" ,
"1111111101001011" ,  "1111111101000110" ,  "1111111101000010" ,
"1111111100111101" ,  "1111111100111000" ,  "1111111100110011" ,
"1111111100101110" ,  "1111111100101001" ,  "1111111100100011" ,
"1111111100011110" ,  "1111111100011001" ,  "1111111100010011" ,
"1111111100001110" ,  "1111111100001000" ,  "1111111100000011" ,
"1111111011111101" ,  "1111111011111000" ,  "1111111011110010" ,
"1111111011101100" ,  "1111111011100110" ,  "1111111011100000" ,
"1111111011011010" ,  "1111111011010100" ,  "1111111011001110" ,
"1111111011001000" ,  "1111111011000001" ,  "1111111010111011" ,
"1111111010110101" ,  "1111111010101110" ,  "1111111010101000" ,
"1111111010100001" ,  "1111111010011011" ,  "1111111010010100" ,
"1111111010001101" ,  "1111111010000110" ,  "1111111010000000" ,
"1111111001111001" ,  "1111111001110010" ,  "1111111001101011" ,
"1111111001100100" ,  "1111111001011100" ,  "1111111001010101" ,
"1111111001001110" ,  "1111111001000110" ,  "1111111000111111" ,
"1111111000111000" ,  "1111111000110000" ,  "1111111000101000" ,
"1111111000100001" ,  "1111111000011001" ,  "1111111000010001" ,
"1111111000001001" ,  "1111111000000001" ,  "1111110111111010" ,
"1111110111110001" ,  "1111110111101001" ,  "1111110111100001" ,
"1111110111011001" ,  "1111110111010001" ,  "1111110111001000" ,
"1111110111000000" ,  "1111110110111000" ,  "1111110110101111" ,
"1111110110100110" ,  "1111110110011110" ,  "1111110110010101" ,
"1111110110001100" ,  "1111110110000100" ,  "1111110101111011" ,
"1111110101110010" ,  "1111110101101001" ,  "1111110101100000" ,
"1111110101010111" ,  "1111110101001101" ,  "1111110101000100" ,
"1111110100111011" ,  "1111110100110001" ,  "1111110100101000" ,
"1111110100011111" ,  "1111110100010101" ,  "1111110100001011" ,
"1111110100000010" ,  "1111110011111000" ,  "1111110011101110" ,
"1111110011100100" ,  "1111110011011010" ,  "1111110011010000" ,
"1111110011000110" ,  "1111110010111100" ,  "1111110010110010" ,
"1111110010101000" ,  "1111110010011110" ,  "1111110010010011" ,
"1111110010001001" ,  "1111110001111111" ,  "1111110001110100" ,
"1111110001101001" ,  "1111110001011111" ,  "1111110001010100" ,
"1111110001001001" ,  "1111110000111111" ,  "1111110000110100" ,
"1111110000101001" ,  "1111110000011110" ,  "1111110000010011" ,
"1111110000001000" ,  "1111101111111101" ,  "1111101111110001" ,
"1111101111100110" ,  "1111101111011011" ,  "1111101111001111" ,
"1111101111000100" ,  "1111101110111000" ,  "1111101110101101" ,
"1111101110100001" ,  "1111101110010101" ,  "1111101110001010" ,
"1111101101111110" ,  "1111101101110010" ,  "1111101101100110" ,
"1111101101011010" ,  "1111101101001110" ,  "1111101101000010" ,
"1111101100110110" ,  "1111101100101001" ,  "1111101100011101" ,
"1111101100010001" ,  "1111101100000100" ,  "1111101011111000" ,
"1111101011101011" ,  "1111101011011111" ,  "1111101011010010" ,
"1111101011000101" ,  "1111101010111001" ,  "1111101010101100" ,
"1111101010011111" ,  "1111101010010010" ,  "1111101010000101" ,
"1111101001111000" ,  "1111101001101011" ,  "1111101001011110" ,
"1111101001010000" ,  "1111101001000011" ,  "1111101000110110" ,
"1111101000101000" ,  "1111101000011011" ,  "1111101000001101" ,
"1111101000000000" ,  "1111100111110010" ,  "1111100111100100" ,
"1111100111010111" ,  "1111100111001001" ,  "1111100110111011" ,
```

```
"1111100110101101" ,  "1111100110011111" ,  "1111100110010001" ,
"1111100110000011" ,  "1111100101110101" ,  "1111100101100111" ,
"1111100101011000" ,  "1111100101001010" ,  "1111100100111100" ,
"1111100100101101" ,  "1111100100011111" ,  "1111100100010000" ,
"1111100100000001" ,  "1111100011110011" ,  "1111100011100100" ,
"1111100011010101" ,  "1111100011000110" ,  "1111100010110111" ,
"1111100010101000" ,  "1111100010011001" ,  "1111100010001010" ,
"1111100001111011" ,  "1111100001101100" ,  "1111100001011101" ,
"1111100001001101" ,  "1111100000111110" ,  "1111100000101111" ,
"1111100000011111" ,  "1111100000010000" ,  "1111100000000000" ,
"1111011111110000" ,  "1111011111100001" ,  "1111011111010001" ,
"1111011111000001" ,  "1111011110110001" ,  "1111011110100001" ,
"1111011110010001" ,  "1111011110000001" ,  "1111011101110001" ,
"1111011101100001" ,  "1111011101010001" ,  "1111011101000001" ,
"1111011100110000" ,  "1111011100100000" ,  "1111011100001111" ,
"1111011011111111" ,  "1111011011101110" ,  "1111011011011110" ,
"1111011011001101" ,  "1111011010111100" ,  "1111011010101100" ,
"1111011010011011" ,  "1111011010001010" ,  "1111011001111001" ,
"1111011001101000" ,  "1111011001010111" ,  "1111011001000110" ,
"1111011000110101" ,  "1111011000100011" ,  "1111011000010010" ,
"1111011000000001" ,  "1111010111101111" ,  "1111010111011110" ,
"1111010111001100" ,  "1111010110111011" ,  "1111010110101001" ,
"1111010110011000" ,  "1111010110000110" ,  "1111010101110100" ,
"1111010101100010" ,  "1111010101010000" ,  "1111010100111110" ,
"1111010100101101" ,  "1111010100011010" ,  "1111010100001000" ,
"1111010011110110" ,  "1111010011100100" ,  "1111010011010010" ,
"1111010010111111" ,  "1111010010101101" ,  "1111010010011011" ,
"1111010010001000" ,  "1111010001110110" ,  "1111010001100011" ,
"1111010001010000" ,  "1111010000111110" ,  "1111010000101011" ,
"1111010000011000" ,  "1111010000000101" ,  "1111001111110010" ,
"1111001111011111" ,  "1111001111001100" ,  "1111001110111001" ,
"1111001110100110" ,  "1111001110010011" ,  "1111001110000000" ,
"1111001101101100" ,  "1111001101011001" ,  "1111001101000110" ,
"1111001100110010" ,  "1111001100011111" ,  "1111001100001011" ,
"1111001011111000" ,  "1111001011100100" ,  "1111001011010000" ,
"1111001010111100" ,  "1111001010101001" ,  "1111001010010101" ,
"1111001010000001" ,  "1111001001101101" ,  "1111001001011001" ,
"1111001001000101" ,  "1111001000110001" ,  "1111001000011100" ,
"1111001000001000" ,  "1111000111110100" ,  "1111000111100000" ,
"1111000110011011" ,  "1111000110110111" ,  "1111000110100010" ,
"1111000110001110" ,  "1111000101111001" ,  "1111000101100100" ,
"1111000101010000" ,  "1111000100111011" ,  "1111000100100110" ,
"1111000100010001" ,  "1111000011111100" ,  "1111000011100111" ,
"1111000011010010" ,  "1111000010111101" ,  "1111000010101000" ,
"1111000010010011" ,  "1111000001111110" ,  "1111000001101000" ,
"1111000001010011" ,  "1111000000111110" ,  "1111000000101000" ,
"1111000000010011" ,  "1110111111111101" ,  "1110111111101000" ,
"1110111111010010" ,  "1110111110111100" ,  "1110111110100111" ,
"1110111110010001" ,  "1110111101111011" ,  "1110111101100101" ,
"1110111101001111" ,  "1110111100111001" ,  "1110111100100011" ,
"1110111100001101" ,  "1110111011110111" ,  "1110111011100001" ,
"1110111011001010" ,  "1110111010110100" ,  "1110111010011110" ,
"1110111010000111" ,  "1110111001110001" ,  "1110111001011010" ,
"1110111001000100" ,  "1110111000101101" ,  "1110111000010111" ,
"1110111000000000" ,  "1110110111101001" ,  "1110110111010010" ,
"1110110110111100" ,  "1110110110100101" ,  "1110110110001110" ,
"1110110101110111" ,  "1110110101100000" ,  "1110110101001001" ,
"1110110100110001" ,  "1110110100011010" ,  "1110110100000011" ,
"1110110011101100" ,  "1110110011010100" ,  "1110110010111101" ,
"1110110010100110" ,  "1110110010001110" ,  "1110110001110111" ,
"1110110001011111" ,  "1110110001000111" ,  "1110110000110000" ,
```

```
"1110110000011000" ,  "1110110000000000" ,  "1110101111101000" ,
"1110101111010001" ,  "1110101110111001" ,  "1110101110100001" ,
"1110101110001001" ,  "1110101101110001" ,  "1110101101011000" ,
"1110101101000000" ,  "1110101100101000" ,  "1110101100010000" ,
"1110101011111000" ,  "1110101011011111" ,  "1110101011000111" ,
"1110101010101110" ,  "1110101010010110" ,  "1110101001111101" ,
"1110101001100101" ,  "1110101001001100" ,  "1110101000110011" ,
"1110101000011011" ,  "1110101000000010" ,  "1110100111101001" ,
"1110100111010000" ,  "1110100110110111" ,  "1110100110011110" ,
"1110100110000101" ,  "1110100101101100" ,  "1110100101010011" ,
"1110100100111010" ,  "1110100100100001" ,  "1110100100001000" ,
"1110100011101110" ,  "1110100011010101" ,  "1110100010111100" ,
"1110100010100010" ,  "1110100010001001" ,  "1110100001101111" ,
"1110100001010110" ,  "1110100000111100" ,  "1110100000100010" ,
"1110100000001001" ,  "1110011111101111" ,  "1110011111010101" ,
"1110011110111011" ,  "1110011110100010" ,  "1110011110001000" ,
"1110011101101110" ,  "1110011101010100" ,  "1110011100111010" ,
"1110011100011111" ,  "1110011100000101" ,  "1110011011101011" ,
"1110011011010001" ,  "1110011010110111" ,  "1110011010011100" ,
"1110011010000010" ,  "1110011001101000" ,  "1110011001001101" ,
"1110011000110011" ,  "1110011000011000" ,  "1110010111111101" ,
"1110010111100011" ,  "1110010111001000" ,  "1110010110101101" ,
"1110010110010011" ,  "1110010101111000" ,  "1110010101011101" ,
"1110010101000010" ,  "1110010100100111" ,  "1110010100001100" ,
"1110010011110001" ,  "1110010011010110" ,  "1110010010111011" ,
"1110010010100000" ,  "1110010010000100" ,  "1110010001101001" ,
"1110010001001110" ,  "1110010000110011" ,  "1110010000010111" ,
"1110001111111100" ,  "1110001111100000" ,  "1110001111000101" ,
"1110001110101001" ,  "1110001110001110" ,  "1110001101110010" ,
"1110001101010110" ,  "1110001100111011" ,  "1110001100011111" ,
"1110001100000011" ,  "1110001011100111" ,  "1110001011001011" ,
"1110001010101111" ,  "1110001010010100" ,  "1110001001110111" ,
"1110001001011011" ,  "1110001000111111" ,  "1110001000100011" ,
"1110001000000111" ,  "1110000111101011" ,  "1110000111001111" ,
"1110000110110010" ,  "1110000110010110" ,  "1110000101111010" ,
"1110000101011101" ,  "1110000101000001" ,  "1110000100100100" ,
"1110000100001000" ,  "1110000011101011" ,  "1110000011001110" ,
"1110000010110010" ,  "1110000010010101" ,  "1110000001111000" ,
"1110000001011011" ,  "1110000000111111" ,  "1110000000100010" ,
"1110000000000101" ,  "1101111111101000" ,  "1101111111001011" ,
"1101111110101110" ,  "1101111110010001" ,  "1101111101110100" ,
"1101111101010111" ,  "1101111100111001" ,  "1101111100011100" ,
"1101111011111111" ,  "1101111011100010" ,  "1101111011000100" ,
"1101111010100111" ,  "1101111010001001" ,  "1101111001101100" ,
"1101111001001110" ,  "1101111000110001" ,  "1101111000010011" ,
"1101110111110110" ,  "1101110111011000" ,  "1101110110111010" ,
"1101110110011101" ,  "1101110101111111" ,  "1101110101100001" ,
"1101110101000011" ,  "1101110100100101" ,  "1101110100000111" ,
"1101110011101001" ,  "1101110011001011" ,  "1101110010101101" ,
"1101110010001111" ,  "1101110001110001" ,  "1101110001010011" ,
"1101110000110101" ,  "1101110000010110" ,  "1101101111111000" ,
"1101101111011010" ,  "1101101110111100" ,  "1101101110011101" ,
"1101101101111111" ,  "1101101101100000" ,  "1101101101000010" ,
"1101101100100011" ,  "1101101100000101" ,  "1101101011100110" ,
"1101101011000111" ,  "1101101010101001" ,  "1101101010001010" ,
"1101101001101011" ,  "1101101001001100" ,  "1101101000101110" ,
"1101101000001111" ,  "1101100111110000" ,  "1101100111010001" ,
"1101100110110010" ,  "1101100110010011" ,  "1101100101110100" ,
"1101100101010101" ,  "1101100100110110" ,  "1101100100010111" ,
"1101100011110111" ,  "1101100011011000" ,  "1101100010111001" ,
"1101100010011010" ,  "1101100001111010" ,  "1101100001011011" ,
```

```
"1101100000111100" ,  "1101100000011100" ,  "1101011111111101" ,
"1101011111011101" ,  "1101011110111110" ,  "1101011110011110" ,
"1101011101111111" ,  "1101011101011111" ,  "1101011100111111" ,
"1101011100100000" ,  "1101011100000000" ,  "1101011011100000" ,
"1101011011000000" ,  "1101011010100000" ,  "1101011010000000" ,
"1101011001100001" ,  "1101011001000001" ,  "1101011000100001" ,
"1101011000000001" ,  "1101010111100001" ,  "1101010111000001" ,
"1101010110100000" ,  "1101010110000000" ,  "1101010101100000" ,
"1101010101000000" ,  "1101010100100000" ,  "1101010011111111" ,
"1101010011011111" ,  "1101010010111111" ,  "1101010010011110" ,
"1101010001111110" ,  "1101010001011101" ,  "1101010000111101" ,
"1101010000011101" ,  "1101001111111100" ,  "1101001111011011" ,
"1101001110111011" ,  "1101001110011010" ,  "1101001101111001" ,
"1101001101011001" ,  "1101001100111000" ,  "1101001100010111" ,
"1101001011110111" ,  "1101001011010110" ,  "1101001010110101" ,
"1101001010010100" ,  "1101001001110011" ,  "1101001001010010" ,
"1101001000110001" ,  "1101001000010000" ,  "1101000111101111" ,
"1101000111001110" ,  "1101000110101101" ,  "1101000110001100" ,
"1101000101101011" ,  "1101000101001001" ,  "1101000100101000" ,
"1101000100000111" ,  "1101000011100110" ,  "1101000011000100" ,
"1101000010100011" ,  "1101000010000010" ,  "1101000001100000" ,
"1101000000111111" ,  "1101000000011101" ,  "1100111111111100" ,
"1100111111011010" ,  "1100111110111001" ,  "1100111110010111" ,
"1100111101110101" ,  "1100111101010100" ,  "1100111100110010" ,
"1100111100010000" ,  "1100111011101111" ,  "1100111011001101" ,
"1100111010101011" ,  "1100111010001001" ,  "1100111001100111" ,
"1100111001000110" ,  "1100111000100100" ,  "1100111000000010" ,
"1100110111100000" ,  "1100110110111110" ,  "1100110110011100" ,
"1100110101111010" ,  "1100110101011000" ,  "1100110100110110" ,
"1100110100010011" ,  "1100110011110001" ,  "1100110011001111" ,
"1100110010101101" ,  "1100110010001011" ,  "1100110001101000" ,
"1100110001000110" ,  "1100110000100100" ,  "1100110000000001" ,
"1100101111011111" ,  "1100101110111101" ,  "1100101110011010" ,
"1100101101111000" ,  "1100101101010101" ,  "1100101100110011" ,
"1100101100010000" ,  "1100101011101110" ,  "1100101011001011" ,
"1100101010101000" ,  "1100101010000110" ,  "1100101001100011" ,
"1100101001000000" ,  "1100101000011110" ,  "1100100111111011" ,
"1100100111011000" ,  "1100100110110101" ,  "1100100110010010" ,
"1100100101110000" ,  "1100100101001101" ,  "1100100100101010" ,
"1100100100000111" ,  "1100100011100100" ,  "1100100011000001" ,
"1100100010011110" ,  "1100100001111011" ,  "1100100001011000" ,
"1100100000110101" ,  "1100100000010010" ,  "1100011111101110" ,
"1100011111001011" ,  "1100011110101000" ,  "1100011110000101" ,
"1100011101100010" ,  "1100011100111110" ,  "1100011100011011" ,
"1100011011111000" ,  "1100011011010101" ,  "1100011010110001" ,
"1100011010001110" ,  "1100011001101010" ,  "1100011001000111" ,
"1100011000100100" ,  "1100011000000000" ,  "1100010111011101" ,
"1100010110111001" ,  "1100010110010110" ,  "1100010101110010" ,
"1100010101001110" ,  "1100010100101011" ,  "1100010100000111" ,
```

les calculs du vecteur wn :

```
% clear all
clc
fid1= fopen('wn_2000.txt', 'wt');
fid2= fopen('binwn_2000.txt', 'wt');
j=0;

for i = 1:128
    j=j+1;
    x(j)=j;
```

```
%       dn = 3.442619855899;
% wn(i,1) = dn;
n=wn(i,1)*2^16;
z(j)=n/2^16;
fprintf(fid1, '"%d" ,  '    , n);

nbin=dec2bin(n,16);
fprintf(fid2, '"%s" ,  ' , nbin );
end
fclose(fid1);
fclose(fid2);

plot(x,z)
```

Le fichier ROM obtenue en décimale pour wn :

```
"2.433408e+05" ,  "1.784682e+04" ,  "2.378114e+04" ,  "2.795425e+04" ,
"3.128937e+04" ,  "3.412171e+04" ,  "3.661445e+04" ,  "3.886042e+04" ,
"4.091797e+04" ,  "4.282638e+04" ,  "4.461350e+04" ,  "4.629985e+04" ,
"4.790104e+04" ,  "4.942926e+04" ,  "5.089421e+04" ,  "5.230376e+04" ,
"5.366441e+04" ,  "5.498157e+04" ,  "5.625983e+04" ,  "5.750308e+04" ,
"5.871471e+04" ,  "5.989763e+04" ,  "6.105440e+04" ,  "6.218728e+04" ,
"6.329828e+04" ,  "6.438917e+04" ,  "6.546157e+04" ,  "6.651690e+04" ,
"6.755649e+04" ,  "6.858152e+04" ,  "6.959307e+04" ,  "7.059214e+04" ,
"7.157966e+04" ,  "7.255646e+04" ,  "7.352333e+04" ,  "7.448100e+04" ,
"7.543016e+04" ,  "7.637144e+04" ,  "7.730544e+04" ,  "7.823273e+04" ,
"7.915384e+04" ,  "8.006928e+04" ,  "8.097952e+04" ,  "8.188504e+04" ,
"8.278625e+04" ,  "8.368359e+04" ,  "8.457746e+04" ,  "8.546824e+04" ,
"8.635631e+04" ,  "8.724204e+04" ,  "8.812578e+04" ,  "8.900788e+04" ,
"8.988867e+04" ,  "9.076848e+04" ,  "9.164763e+04" ,  "9.252646e+04" ,
"9.340527e+04" ,  "9.428437e+04" ,  "9.516408e+04" ,  "9.604471e+04" ,
"9.692656e+04" ,  "9.780995e+04" ,  "9.869519e+04" ,  "9.958258e+04" ,
"1.004724e+05" ,  "1.013651e+05" ,  "1.022609e+05" ,  "1.031601e+05" ,
"1.040631e+05" ,  "1.049702e+05" ,  "1.058818e+05" ,  "1.067982e+05" ,
"1.077198e+05" ,  "1.086470e+05" ,  "1.095802e+05" ,  "1.105197e+05" ,
"1.114660e+05" ,  "1.124196e+05" ,  "1.133808e+05" ,  "1.143502e+05" ,
"1.153283e+05" ,  "1.163155e+05" ,  "1.173125e+05" ,  "1.183198e+05" ,
"1.193380e+05" ,  "1.203677e+05" ,  "1.214097e+05" ,  "1.224646e+05" ,
"1.235332e+05" ,  "1.246164e+05" ,  "1.257149e+05" ,  "1.268298e+05" ,
"1.279620e+05" ,  "1.291127e+05" ,  "1.302829e+05" ,  "1.314740e+05" ,
"1.326873e+05" ,  "1.339242e+05" ,  "1.351865e+05" ,  "1.364759e+05" ,
"1.377942e+05" ,  "1.391438e+05" ,  "1.405269e+05" ,  "1.419462e+05" ,
"1.434046e+05" ,  "1.449054e+05" ,  "1.464524e+05" ,  "1.480497e+05" ,
"1.497022e+05" ,  "1.514153e+05" ,  "1.531953e+05" ,  "1.550497e+05" ,
"1.569872e+05" ,  "1.590179e+05" ,  "1.611542e+05" ,  "1.634110e+05" ,
"1.658067e+05" ,  "1.683642e+05" ,  "1.711127e+05" ,  "1.740903e+05" ,
"1.773479e+05" ,  "1.809560e+05" ,  "1.850163e+05" ,  "1.896837e+05" ,
"1.952118e+05" ,  "2.020625e+05" ,  "2.112281e+05" ,  "2.256155e+05" ,
```

Le fichier ROM obtenue en binaires pour wn :

```
"1110110110100011100" ,  "0100001011011O" ,  "0101110011100101" ,
"0110110100110010" ,  "0111101000111001" ,  "1000010101001001" ,
"1000111100000110" ,  "1001011111001100" ,  "1001111111010101" ,
```

```
"1010011101001010" ,  "1010111001000101" ,  "1011010011011011" ,
"1011101100011101" ,  "1100000100010101" ,  "1100011011001110" ,
"1100110001001111" ,  "1101000110100000" ,  "1101011011000101" ,
"1101101111000011" ,  "1110000010011111" ,  "1110010101011010" ,
"1110100111111001" ,  "1110111001111110" ,  "1111001011101011" ,
"1111011101000010" ,  "1111101110000101" ,  "1111111110110101" ,
"10000001111010100" ,  "10000011111100100" ,  "10000101111100101" ,
"10000111111011001" ,  "10001001111000000" ,  "10001011110011011" ,
"10001101101101100" ,  "10001111100110011" ,  "10010001011110000" ,
"10010011010100110" ,  "10010101001010011" ,  "10010110111111001" ,
"10011000110011000" ,  "10011010100110001" ,  "10011100011000101" ,
"10011110001010011" ,  "10100000111011101" ,  "10100001101100010" ,
"10100011011100011" ,  "10100101001100001" ,  "10100110111011100" ,
"10101000101010100" ,  "10101010011001010" ,  "10101100000111101" ,
"10101101110101111" ,  "10101111100100000" ,  "10110001010010000" ,
"10110010111111111" ,  "10110100101101110" ,  "10110110011011101" ,
"10111000001001100" ,  "10111001110111100" ,  "10111011100101100" ,
"10111101010011110" ,  "10111111000010001" ,  "11000000110000111" ,
"11000010011111110" ,  "11000100001111000" ,  "11000101111110101" ,
"11000111101110100" ,  "11001001011111000" ,  "11001011001111111" ,
"11001101000001010" ,  "11001110110011001" ,  "11010000100101110" ,
"11010010011000111" ,  "11010100001100111" ,  "11010110000001100" ,
"11010111110110111" ,  "11011001101101010" ,  "11011011100100011" ,
"11011101011100100" ,  "11011111010101110" ,  "11100001010000000" ,
"11100011001011011" ,  "11100101001000000" ,  "11100111000101111" ,
"11101001000101001" ,  "11101011000101111" ,  "11101101001000001" ,
"11101111001100000" ,  "11110001010001101" ,  "11110011011001000" ,
"11110101100010010" ,  "11110111101101101" ,  "11111001111011010" ,
"11111100001011000" ,  "11111110011101010" ,  "100000000110010001" ,
"100000011001001111" ,  "100000101100100100" ,  "100001000000010010" ,
"100001010100011011" ,  "100001101001000010" ,  "100001111110000111" ,
"100010010011101110" ,  "100010101001111010" ,  "100011000000101100" ,
"100011011000001001" ,  "100011110000010100" ,  "100100001001010001" ,
"100100100011000110" ,  "100100111101110111" ,  "100101011001101011" ,
"100101110110101001" ,  "100110010100111011" ,  "100110110100101001" ,
"100111010110000010" ,  "100111111001010011" ,  "101000011110101110" ,
"101001000110101100" ,  "101001110001101000" ,  "101010100000001010" ,
"101011010011000011" ,  "101100001011011011" ,  "101101001010111000" ,
"101110010011110011" ,  "101111101010001011" ,  "110001010101001110" ,
"110011100100011100" ,  "110111000101001111" ,
```

3.4 Fonctionnement matériel de la méthode de Ziggurat en virgule fixe

3.4.1 Interprétation

- Interface de xilinx

Figure 3.4 *: Interface de xilinx*

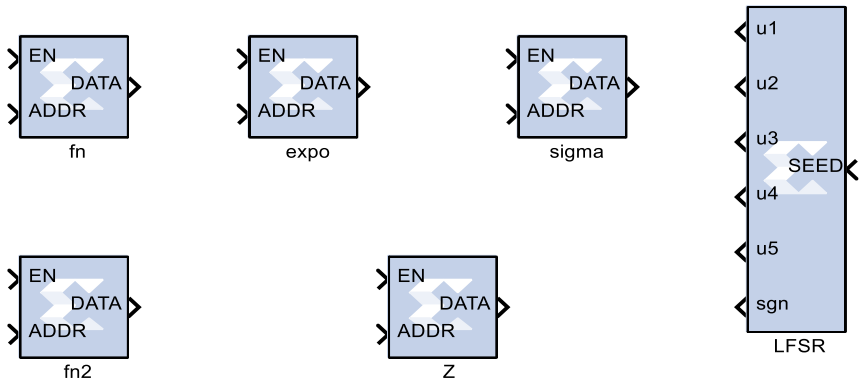

Figure3.5 *: les blacs de xilinx*

Pour générateur Ziggourat proposé dans notre travail comprend les élément suivant :

-Un registre de décalage à linéaire (LFSR) qui générer des variables aléatoire uniforme .

-u1,u2et u3 trois variables aléatoires indépendantes uniformément distrébuées dans]0 ,1] .

-Rom de Sigma,fn,fn2, wn,et exp .

La fonction sigma , fn , fn2 , wn , et exp sont des ROM développées par VHDL.

Block ROM sigma :

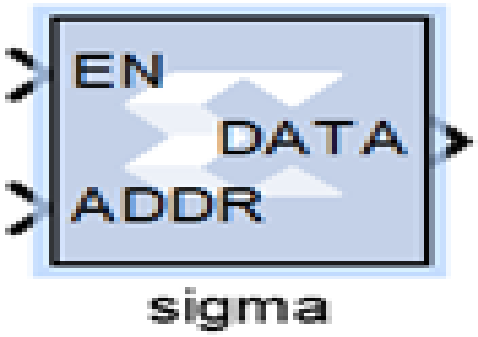

Figure 3.6 *: Bloc ROM Sigma*

Paramètre de Sigma :

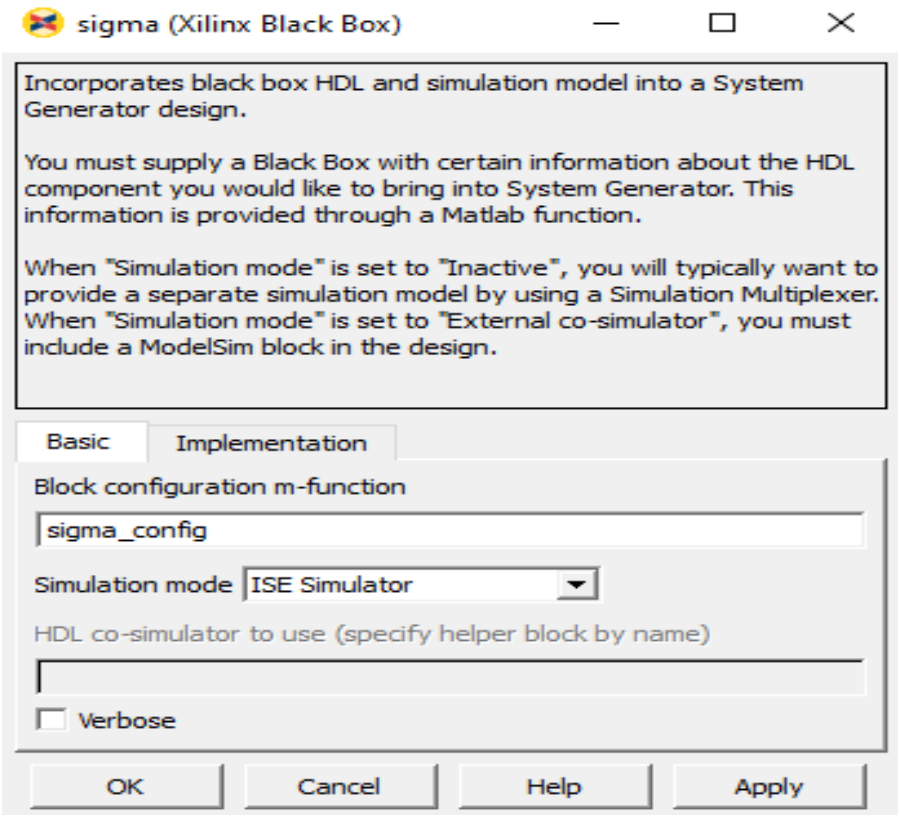

Figure 3.7 *: Paramétre de Sigma*

Contient :

```
entity sigma is

port (C : in std_logic;

CE : in std_logic

   EN : in std_logic;

 ADDR : in std_logic_vector(6 downto 0);

 DATA : out std_logic_vector( 15 downto 0));

end sigma;
```

```
architecture Behavioral of sigma is

type rom_type is array (127 downto 0) of std_logic_vector ( 15
downto 0);

    signal ROM : rom_type:= ( "1110110101011010" ,
"0000000000000000" ,  "1100000000011110" ,  "1101100111001000" ,
"1110010010110110" ,  "1110101011000000" ,  "1110111010010010" ,
"1111000100110100" ,  "1111001100100000" ,  "1111010010010111" ,
"1111010110111110" ,  "1111011010101101" ,  "1111011101110001" ,
"1111100000010101" ,  "1111100010100001" ,  "1111100100011001" ,
"1111100110000010" ,  "1111100111011101" ,  "1111101000101110" ,
"1111101001110111" ,  "1111101010110111" ,  "1111101011110001" ,
"1111101100100110" ,  "1111101101010110" ,  "1111101110000001" ,
"1111101110101001" ,  "1111101111001110" ,  "1111101111110000" ,
"1111110000001111" ,  "1111110000101100" ,  "1111110001000111" ,
"1111110001100000" ,  "1111110001110111" ,  "1111110010001101" ,
"1111110010100010" ,  "1111110010110101" ,  "1111110011000111" ,
"1111110011011000" ,  "1111110011101000" ,  "1111110011110111" ,
"1111110100000101" ,  "1111110100010010" ,  "1111110100011111" ,
"1111110100101011" ,  "1111110100110110" ,  "1111110101000001" ,
"1111110101001011" ,  "1111110101010100" ,  "1111110101011110" ,
"1111110101100110" ,  "1111110101101110" ,  "1111110101110110" ,
"1111110101111101" ,  "1111110110000100" ,  "1111110110001011" ,
"1111110110010001" ,  "1111110110010111" ,  "1111110110011100" ,
"1111110110100010" ,  "1111110110100111" ,  "1111110110101011" ,
"1111110110110000" ,  "1111110110110100" ,  "1111110110111000" ,
"1111110110111011" ,  "1111110110111110" ,  "1111110111000001" ,
"1111110111000100" ,  "1111110111000111" ,  "1111110111001001" ,
"1111110111001011" ,  "1111110111001101" ,  "1111110111001111" ,
"1111110111010000" ,  "1111110111010001" ,  "1111110111010010" ,
"1111110111010011" ,  "1111110111010100" ,  "1111110111010100" ,
"1111110111010100" ,  "1111110111010100" ,  "1111110111010011" ,
"1111110111010011" ,  "1111110111010010" ,  "1111110111010000" ,
"1111110111001111" ,  "1111110111001101" ,  "1111110111001011" ,
"1111110111001001" ,  "1111110111000110" ,  "1111110111000011" ,
"1111110110111111" ,  "1111110110111100" ,  "1111110110110111" ,
"1111110110110011" ,  "1111110110101110" ,  "1111110110101000" ,
```

```
"1111110110100010" ,  "1111110110011100" ,  "1111110110010100" ,
"1111110110001100" ,  "1111110110000100" ,  "1111110101111010" ,
"1111110101110000" ,  "1111110101100101" ,  "1111110101011001" ,
"1111110101001011" ,  "1111110100111100" ,  "1111110100101100" ,
"1111110100011010" ,  "1111110100000110" ,  "1111110011110000" ,
"1111110011010111" ,  "1111110010111011" ,  "1111110010011011" ,
"1111110001110110" ,  "1111110001001101" ,  "1111110000011100" ,
"1111101111100011" ,  "1111101110011111" ,  "1111101101001100" ,
"1111101011100101" ,  "1111101001100001" ,  "1111100110110011" ,
"1111100011000000" ,  "1111011101010010" ,  "1111010011100100" ,
"1110111110101100" );

signal rdata : std_logic_vector( 15 downto 0);

begin

    rdata <= ROM(conv_integer(ADDR));

    process (C)

    begin

        if (C'event and C = '1') then

             if (CE = '1') then

            if (EN = '1') then

                DATA <= rdata;

            end if;

        end if;

  end if;

end process; end Behavioral;
```

3.5 Conclusion

Dans ce chapitre nous avons étudie la méthode de Ziggourat en virgule fixe, nous avons dans un premier temps, décrit le principe de fonctionnement de l'algorithme de Ziggourat. Cela nous a permis de comprendre que cet algorithme est une forme d'algorithme de rejet. Il en est aussi ressorti que, malgré sa lenteur, cet algorithme le de loin préférer par rapport à la méthode de Wallace décrite au chapitre précédent. Dans la deuxième partie, nous faisons ressortir le principe logique du codage en virgule fixe : il en est ressorti que le codage en virgule fixe est le plus rapide et aussi le plus utilisé malgré le fait qu'il ne soit pas aussi précis les autres méthodes de codages. La troisième partie un peu plus intéressante nous permet de décrire l'architecture matérielle de l'algorithme de Ziggourat en virgule fixe. Nous y avons alors ressortit les différents constituants de l'implémentation matérielle de cette algorithme dans un circuit

4.1 Introduction

La dernier étape pour compléter notre travail consiste à utiliser le langage de description de matériel VHDL et un logiciel Xilinx ISE .La description VHDL utilisé pour synthèse l' architecture globale de notre générateur de nombre aléatoire gaussien .Le système Générateur et Xilinx sont utilisés pour validation et test

Nous comparons les résultats des générateur avec les résultats du Matlab, et les résultats du système Générateur .Nous concluent le chapitre par une conclusion

4.2 Implémentation et test sur MatLab

- **Resultat de l'algorithme**

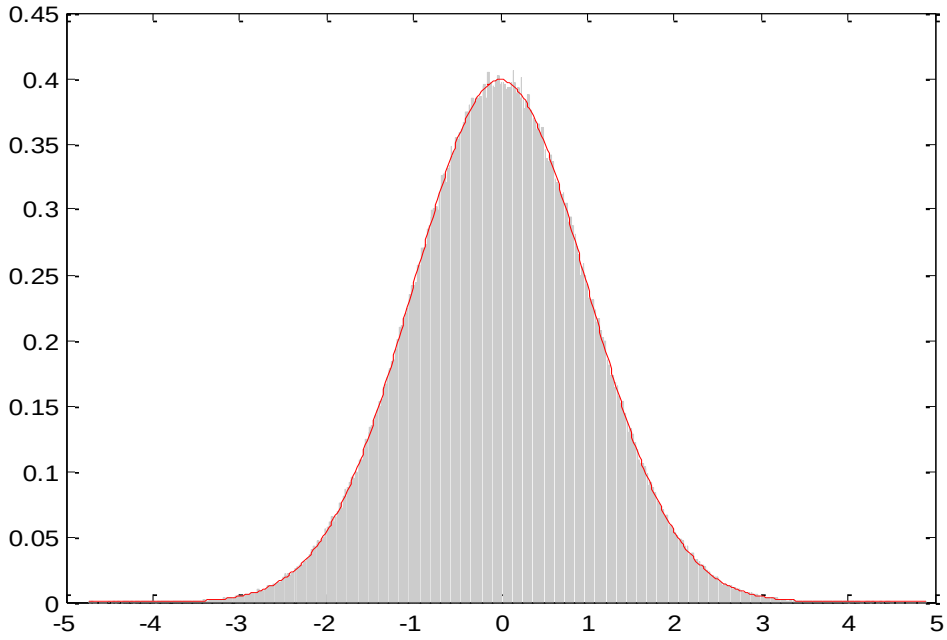

***Figure4.1** : Distribution d'un variable aléatoire gaussiene de Ziggouratdans MATLAB*

Remarque : Cette figure représente la distribution de variable aléatoire gaussien générée par notre approche la méthode Ziggourat et une distribution idéale normale N (0.1)

-Dans cette distribution on à 3 régions : la première les nombre aléatoire qui situe sous la courbe de F (courbe tracé en rouge) on choisit les xi de telle sorte que la portion d'aire de f située dans chaque rectangle soit constante, la deuxième régions les nombres aléatoire qui sont a l'extérieur de ce courbe on faire un autre traitement pour choisit seulement qui sont dans le courbe ,et la troisième région on a le Tail

- Résultat de l'algorithme pour générer l'autocorrélation

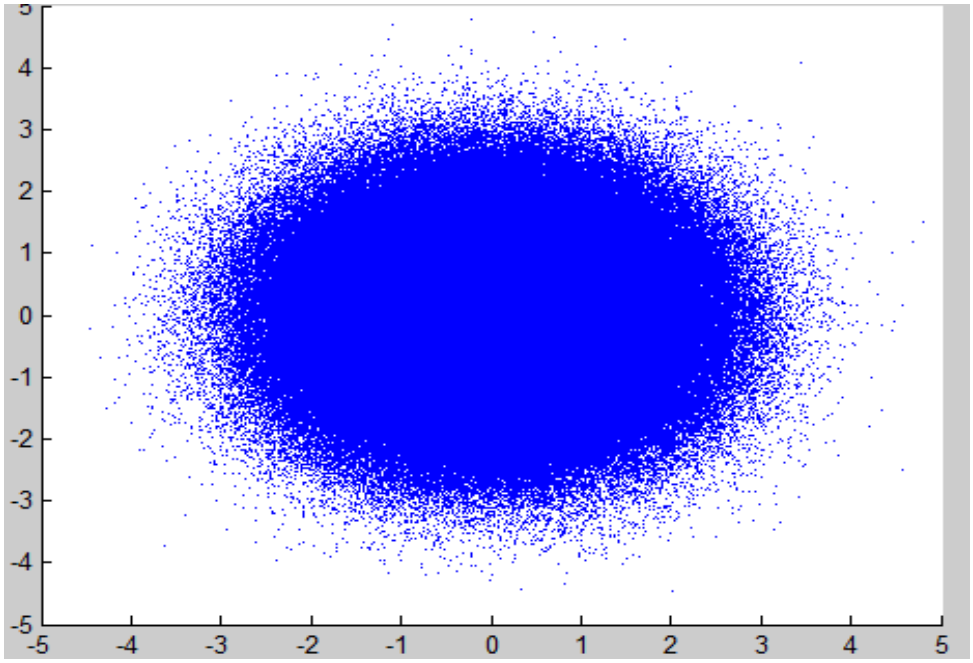

Figure 4.2 *: Auto-corrélation*

Remarque : cette figure représentée l'autocorrélation de Ziggourat

Le créateur spectral d'une Ziggourat dans Matlab représente :

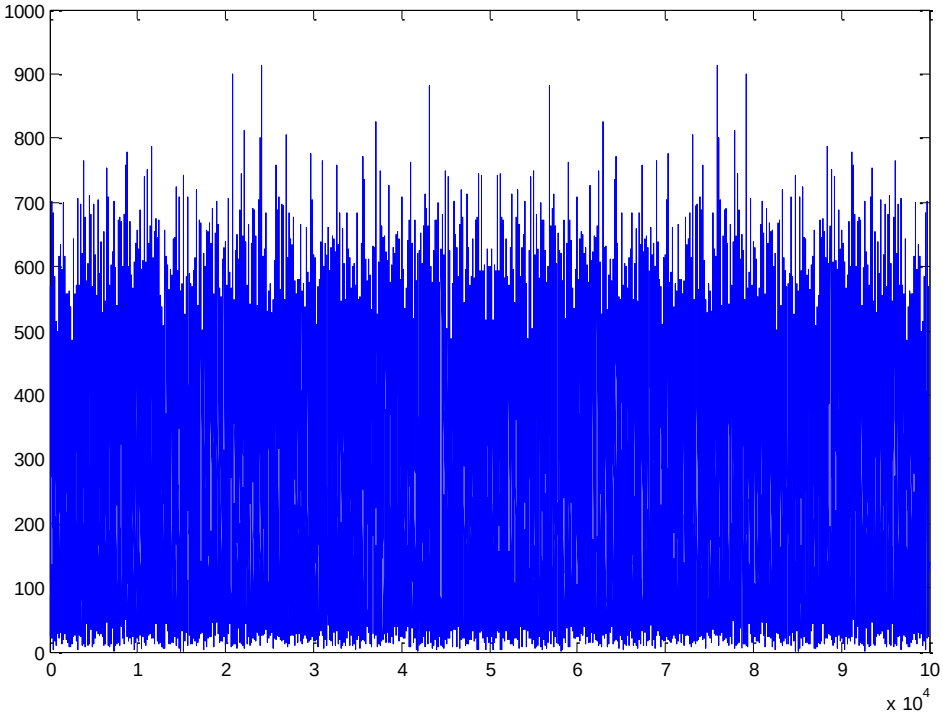

Figure 4 .3 *: Spectrale d'un Ziggourat dans Matlab*

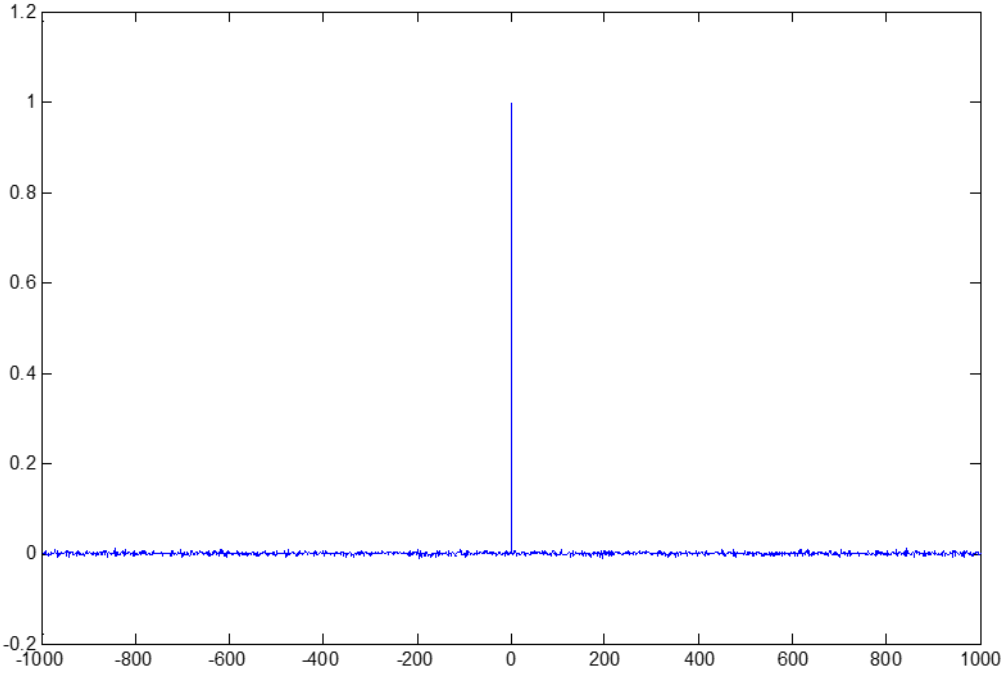

Figure 4 .4 *: Auto-corrélation d'un Ziggourat dans Matlab*

Remarque : Le figure représente autocorrélation d'un Ziggourat dans Matlab ,en obtient comme un résultat un autocorrélation nulle et un pic centrale dans 0 .

-Les résultats d'une numérisation les sorties sigma ,fn,fn2 ,wn ,exp en bloc ROM

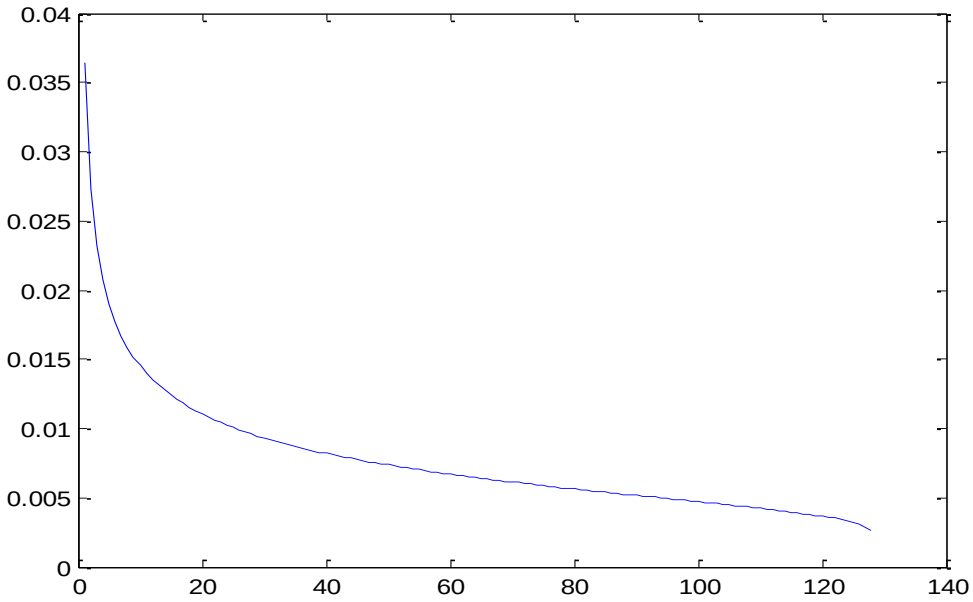

***Figure 4.5** : Numérisation d'un bloc ROM sigma*

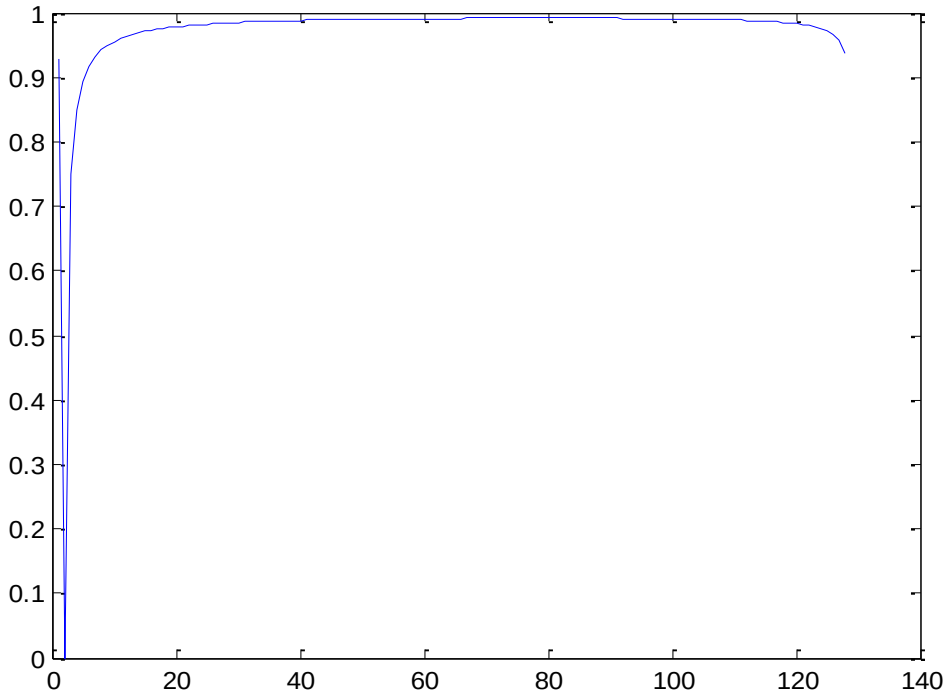

***Figure 4.6** : Numérisation d'un bloc ROM fn*

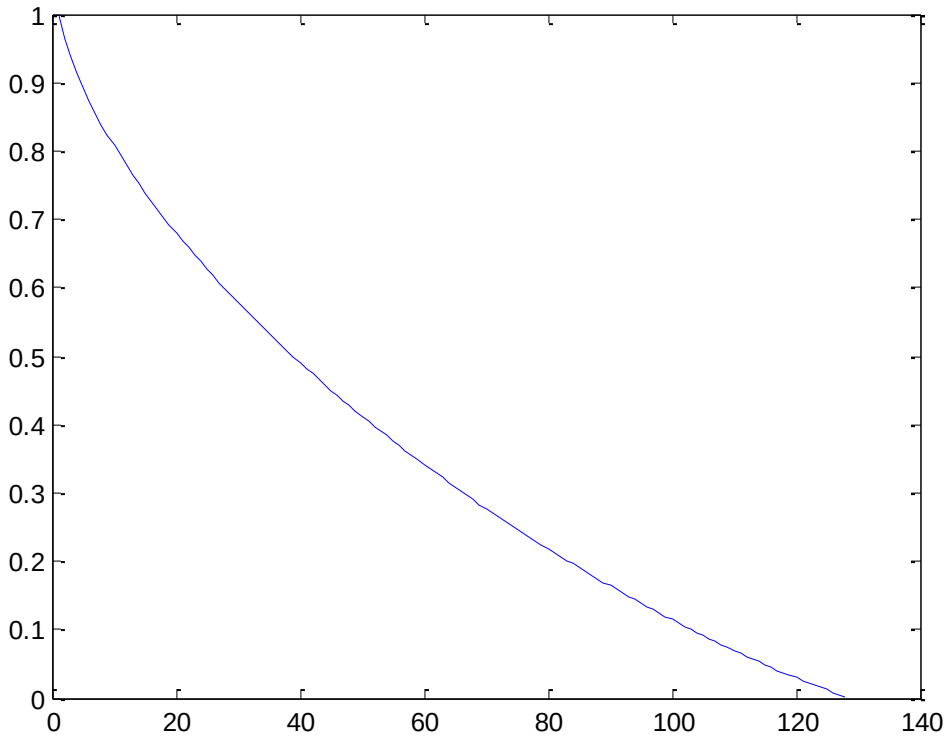

Figure 4.7 *: Numérisation d'un bloc ROM fn2*

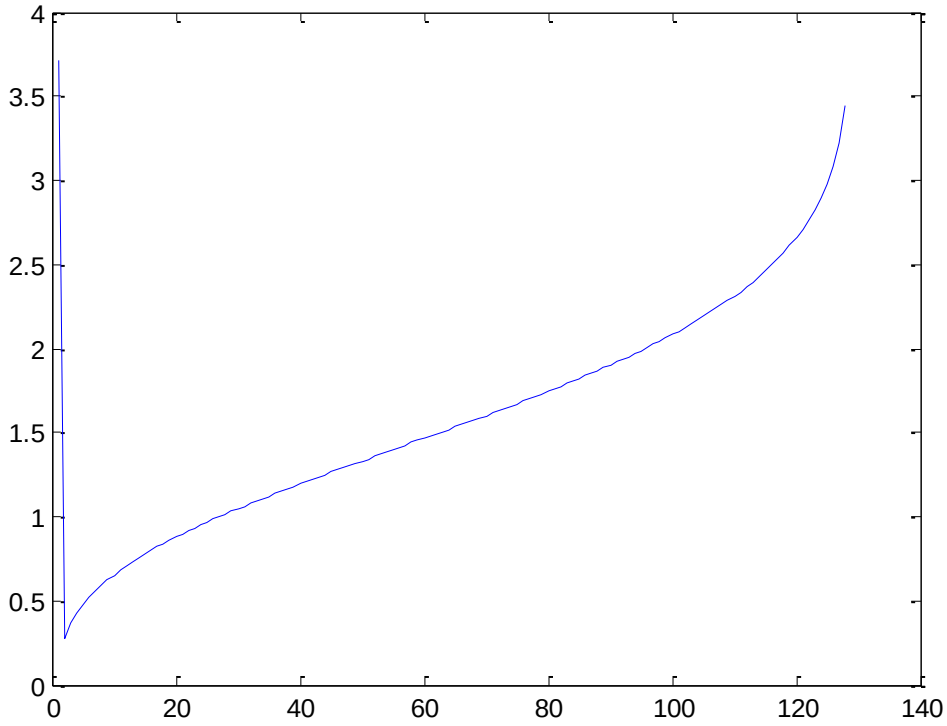

Figure 4.8 *: Numérisation d'un bloc wn*

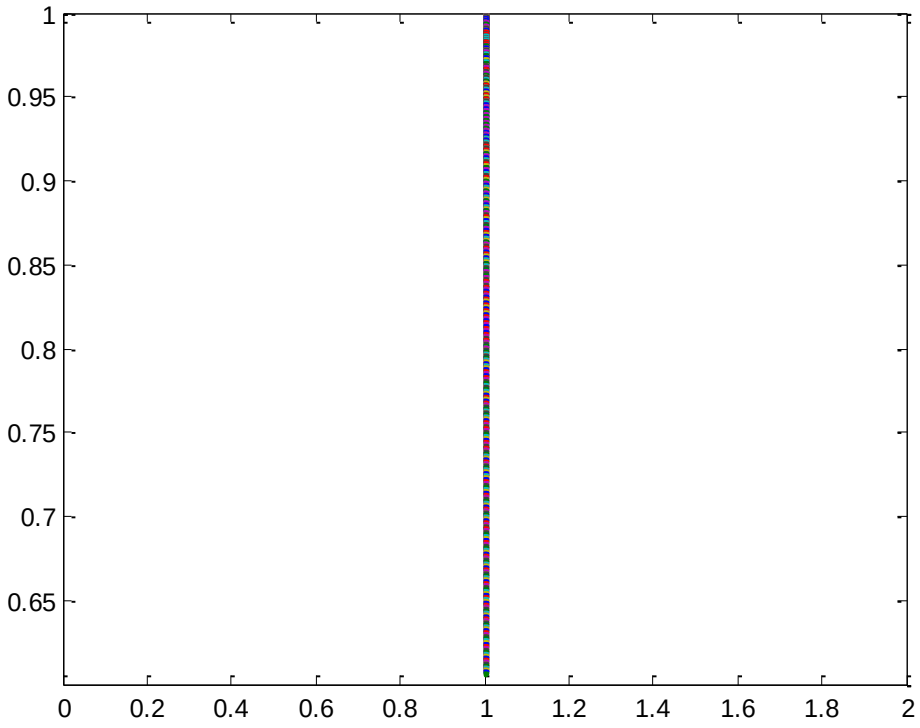

***Figure 4.9** : Numérisation d'un bloc exp*

4.3 Implémentation sur FPGA

Pourréaliser l'architecture de la méthode Ziggourat en besoin Système générateur

-Bibliographique de Simulink d'un Système générateur

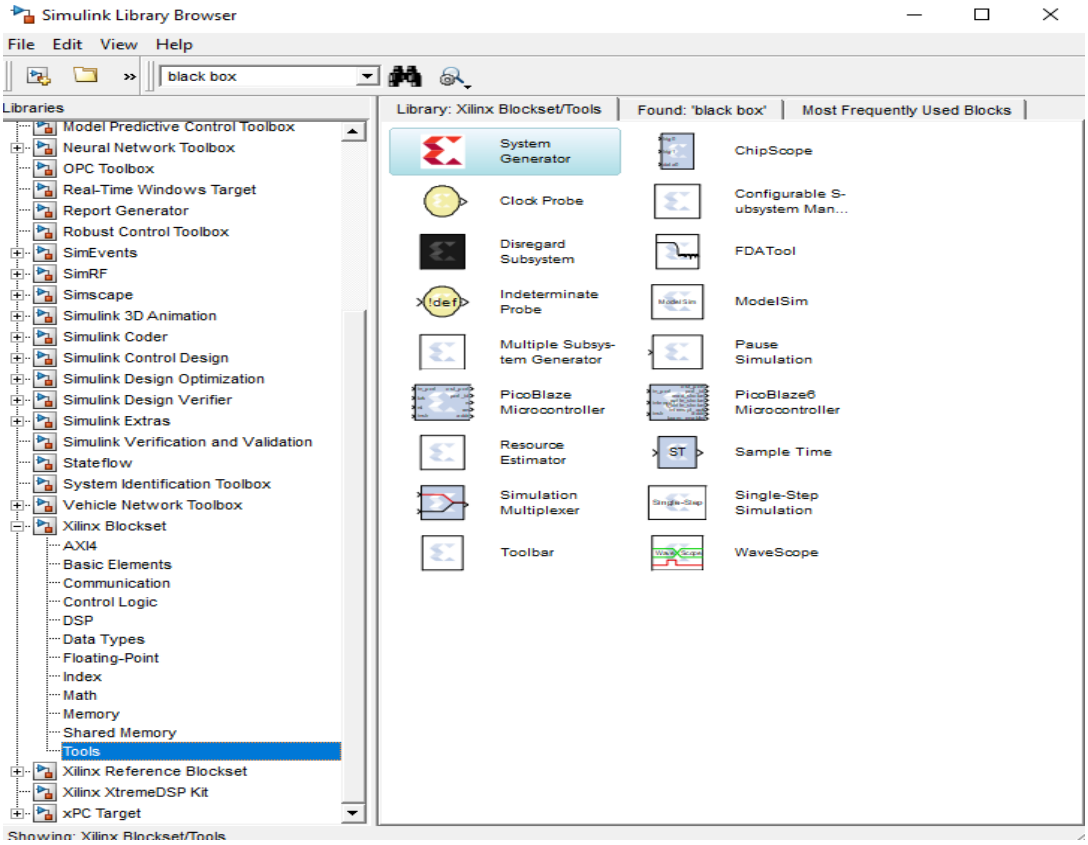

***Figure 4.10**: Bibliographique d'un Système Générateur*

***Figure 4.11** : Système Générateur*

Paramétre de syséme génértor :

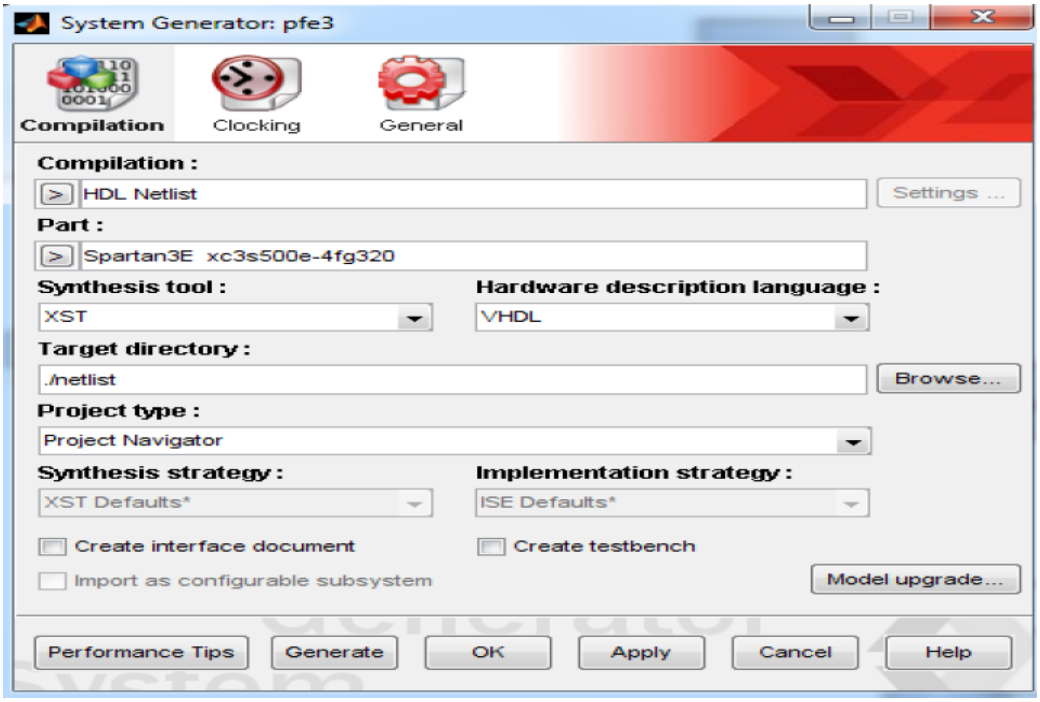

Figure 4.12 *: Paramétre de syséme génértor*

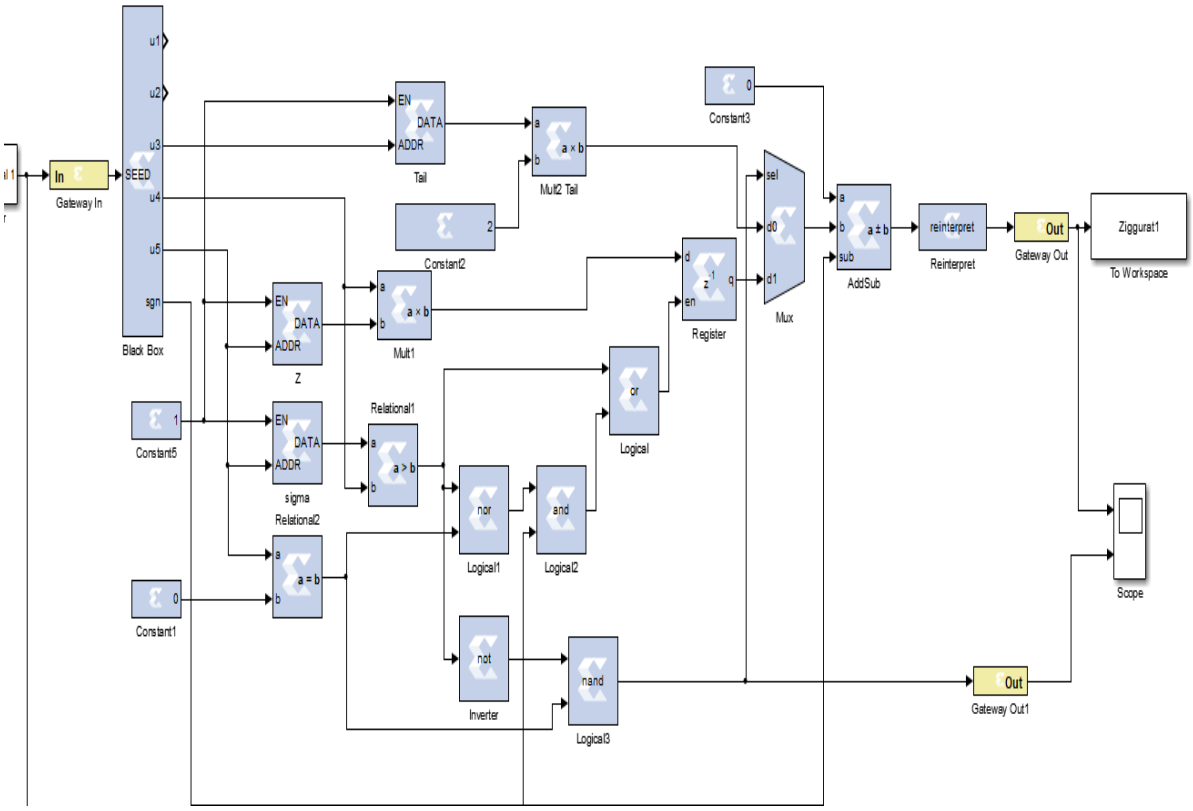

Figure 4.13*: L' architecture de générateur de nombre aléatoire gaussien par la méthode de Ziggourat*

Remarque : L'architecture qui est en gris en utilisée pour faire la simulation dans le Matlab

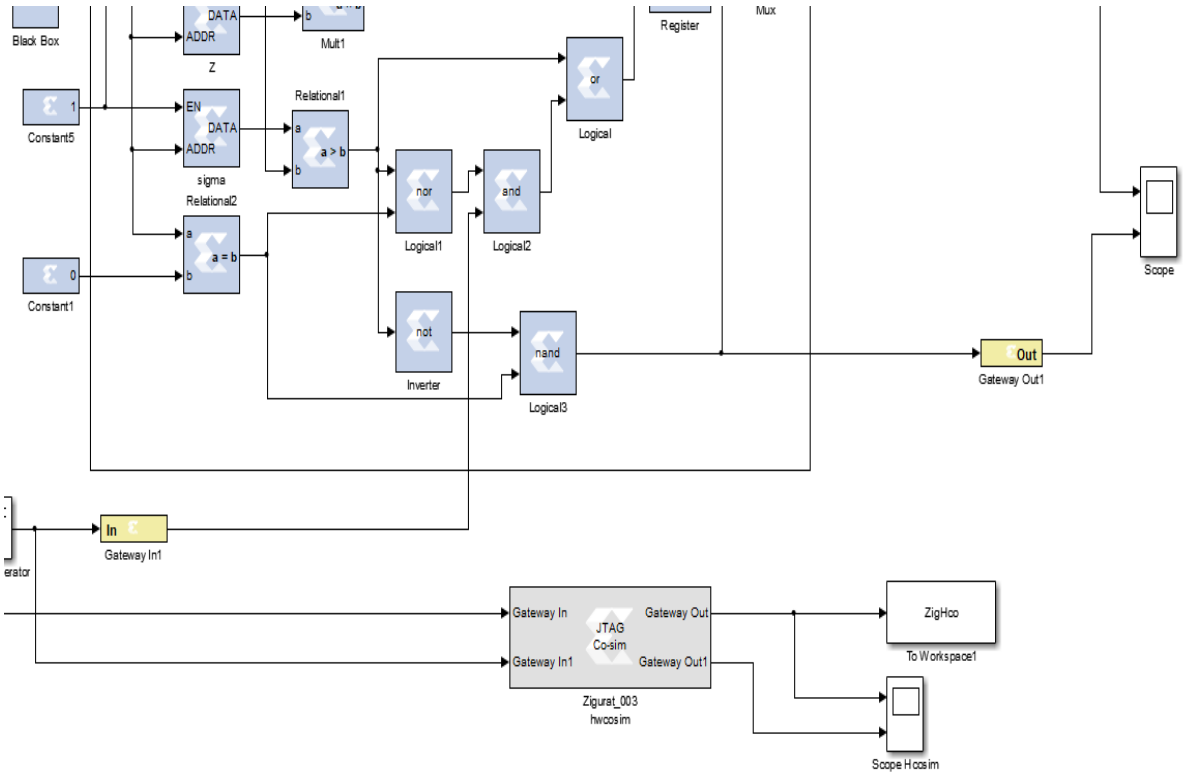

***Figure 4.14**: L' architecture de générateur de nombre aléatoire gaussien avec la méthode Ziggourat*

Remarque : L'architecture qui est en gris en utilisée pour faire la simulation dans la caret de FPGA

-Nous mettons l'architecture précédée en bleu dans une Bloc qui est en gris

Simulation d'une spectrale dans Système Générateur :

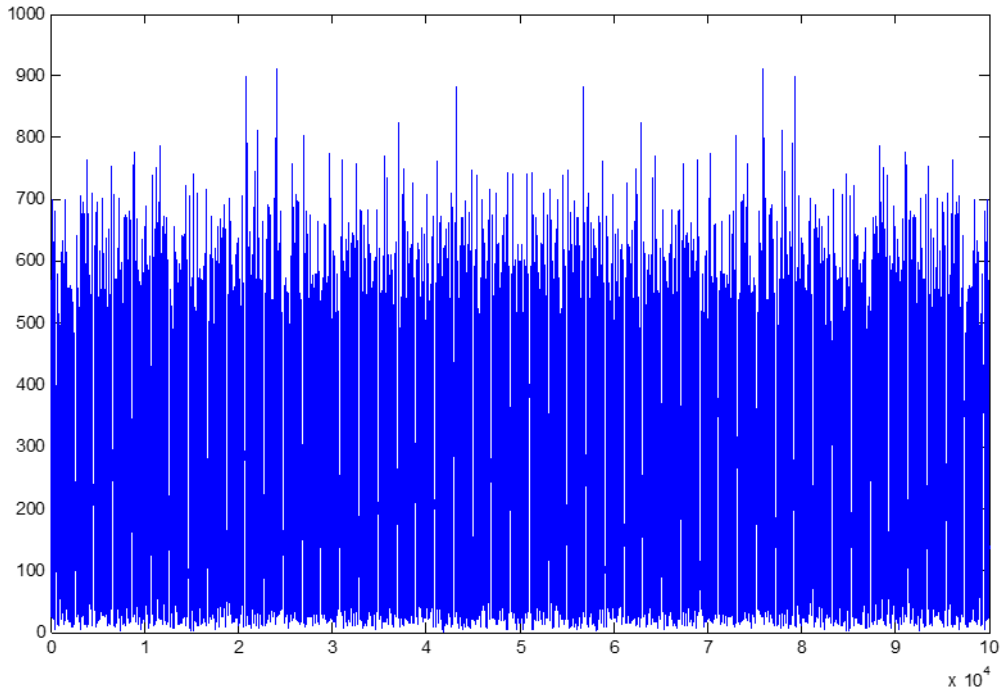

***Figure 4 .15**: Spectrale d'un Ziggourat dans Système Générateur*

- Simulation d'une autocorrélation dans Système Générateur :

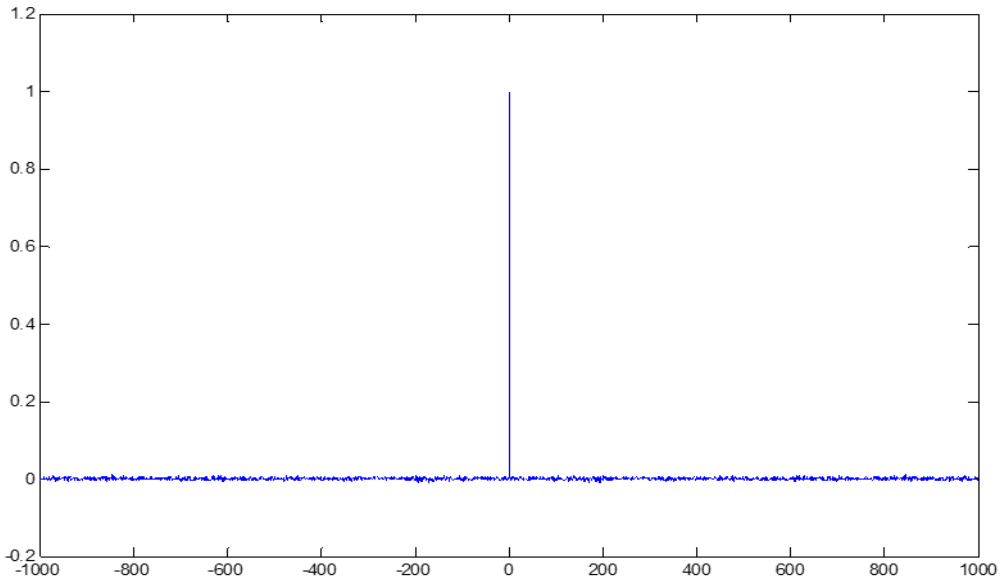

***Figure 4 .16** : Auto-corrélation d'un Ziggourat dans Système Générateur*

Remarque : les deux figure 4 . représentes autocorrélation et spectrale dans Système Générateur donnes les résultats similaire que dans le Matlab .

4.4 Simulations et résultats

Nous avons implémenté le générateur du nombre aléatoire gaussien sur la carte FPGA utilisant système générateur et Matlab

D'après les résultats obtenus on remarque que le bruit blanc gaussien obtenue est de bonne qualité

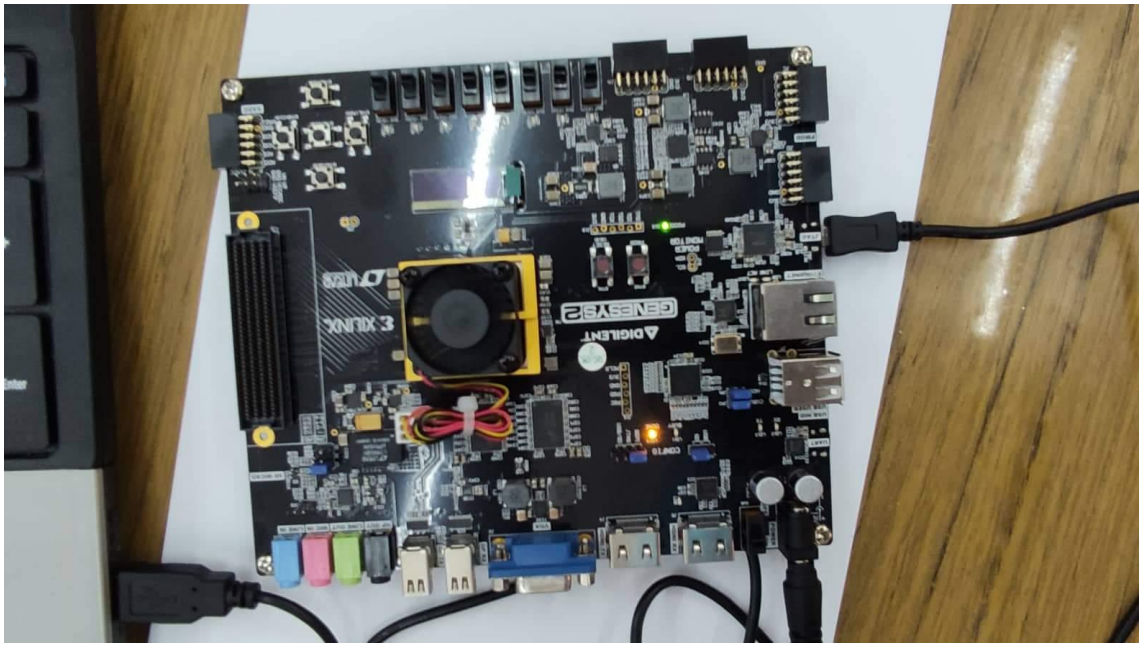

Figure 4.17: *la carte FPGA utilisé dans l'implimentation*

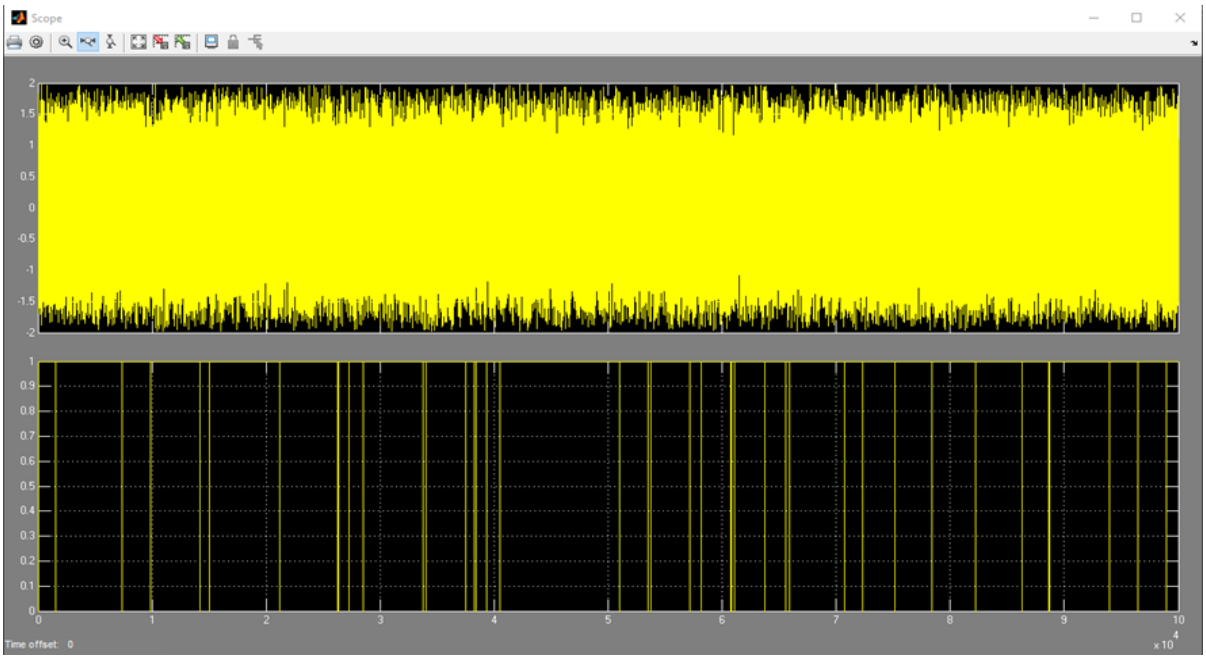

Figure 4. 18 : *Bruit de nombre aléatoire gaussien par FPGA*

Remarque : les résultats obtenus à partir des simulation dans Matlab et FPGA sont semblables .

4.5 Conclusion

Dans ce chapitre, nous avons utilisé le Matlab et l'ISE et le système générateur de XILINX pour simuler, implémentation et tester notre générateur de nombre aléatoire gaussien.

Les résultats obtenus à partir des simulations du modèle à virgule flottants de Matlab, des modèle de virgule fixe du système générateur notre modèle à virgule fixe utilisant la méthode Ziggourat , démontre la supérieurété de notre approche par rapport à l'approche de Système générateur .

Les tests sur carte FPGA confirme parfaitement cette supérieurété.

Conclusion générale

Dans ce projet de fin d'étude, nous avons présenté une nouvelle méthode Ziggourat pour générer les nombres aléatoires gaussien qui donne de bonnes performances en utilisant un chemin d'exécution d' échantillonnage de rejet très simple pour la majorité des points d'échantillonnage générés, mais avec les des calculs plus couteux effectués pour maintenir l'exécution mathématique dans certains cas d'angle qui spécifique représentés par la queue de distribution et les bords éloignés de la Ziggourat , une alternative rapide consiste à utiliser une approximation du pdf gaussien ,en particulier une représentation linéaire par morceaux qui revient à remplacer les rectangles empilés par des trapèze empilés .pour de nombreuses utilisations, en obtiens les résultats plus rapide .

Afin d'utilisé la méthode de Ziggourat pour générer les nombres aléatoires gaussien dans Matlab et en FPGA en trouve que les résultats obtenus dans software (Matlab) plus rapide que dans Hardware (FPGA).

Référence Bibliographie

[1] Altera : Site web altera - http ://www.altera.com/. 9

[2] Amba : Ambaspecification rev2.0. arm, 6, 8 ,70. 42

[3] Ardmahn : Site web ardmahn : Architecture reconfigurable dynamiquement et methodologie pour auto-adaptation home networking - http ://ardmahn.org/. 5

[4] N. Marques : Méthodlogie et architecture adaptative pour le placement efficace des tâches materielles de tailles variables sur des partitions reconfigurables. These de doctorat, Université de Lorraine, 2012. 30

[5] P. Bertin : Memoires actives programmables : conception, realisation et programmation. These de doctorat, These de doctorat, Universitede Paris 07, 1992. 18

[6] J. Mariani : Programmation et Utilisation du FPGA pour la validation et la vérification de circuits électroniques. Electronique. 2011. dumas-00574220

[7] A. Tisserand : Introduction aux circuits FPGA, Présentation Séminaire MIM, 2003 http://www.irisa.fr/prive/Arnaud.Tisserand/docs/semmim-at-fpga.pdf

[8] altera Corporation : Site d'un constructeur de FPGA, Ensemble de documents techniques concernant les FPGA, 2010 http://www.altera.com/literature/lit-index.html

[9] M. Thompson: FPGAsaccelerate time to market for industrials designs, article de l'EETimes concernant les avantages de l'utilisation du FPGA dans l'industries, 2004 http://www.designreuse.com/articles/exit/?id=8190&url=http://www.eetimes.com/showArticle.jhtml;jsessionid=4OJLA20JQAVNSQSNDBGCKHSCJUMEKJVN?articleID=22102798

[10] Les FPGA : introduction. http://proxacutor.free.fr/architecture

[11] M. E. Muller, « A comparaison of methods for generating normal deviates on digital computers » ; J. ACM, vol. 6, no. 3, pp. 376-383, 1959.

[12] D. Arnold, « Curvature in matlab », Maths 50C MultivariableCalculus

[13] L. Colavito and D. Silage, « Efficient FPGA LFSR ImplementationWhitens Pseudo randomNumbers », InternatinalConference on Reconfigurable Computing and FPGAs, 2009.

[14] J. S. Malik, J. N. Malik, A. Hemani, N. D. Gohar, « Generating high tailaccuracy Gaussian randomnumbers in hardwarusing central limittheorem », Conferencepaper, January 2011.

[15] R. B. D'Agostino, M. A. Stephens, « Goodness-of-Fit-techniques (Stactistcs : aSeries of textbooks and Monographs, Vol. 68) », New York : Marcel Dekker, 1986.

[16] C. Wallace, « Fast pseudo randomgenerators for normal and exponentialvariates, » ACM Trans. Math. Softw :, vol. 22, no. 1, pp. 119-127, 1996.

[17] D-U. Lee, W. Luk, J. D. Villasenor, G. Zhang and P. H. W. Leong, « A hardware gaussian noise generatorusing the Wallace method », IEEE transactions on ver large scaleintegration (VLSI) systems, vol. 13, no. 8, 2005.

[18] P. Chu and R. Jones, « Disign techniques of FPGA based randomnumbergenerator, » presentedat the Millitary and aerospace applications of programmable devices and technolgyConf., Laurel, MD, 1999.

[19] P. L'Ecuyer, « MaximallyequidistributedcombinedTausworthegenerators, » Mathematics of computation, vol. 65, no. 213, pp. 203-213, 1973.

[20] J. Chen, J. Moon, and K. Bazargan, « Reconfigurable readback-signal generatorbased on a field-programmable gatearray, » IEE Trans. Magn., vol. 40, no 3, pp. 1744-1750, Mar. 2004.

[21] G. Boooox and M. Muller, « A note on the generation of random normal deviates, » Ann. Math. Statist., vol. 29, pp. 610-611, 1958.

[22] D-U. Lee, W. Luk, J. D. Villasenor and P. H. W. Leong, « A hardware gaussian noise generatorusing the Box-Mulermethod and itserroranalysis», IEEE transactions on ver large scaleintegration (VLSI)systems, vol. 13, no. 8, 2005.

[23] C. P. Robert and G. Casella, « Monte Carlo StatisticalMethods » (Second edition). New York : Springer-Verlag, 2004.

[24] S. F. Ismael, B. S. Mahmood, « Architectural design of randomnumbergenerators and their hardware implementations », Al-Rafidain Engineering, vol. 22, no. 2, 2014.

[25] G. Knuth, seminumericalalgorithms, ser. The Art of Comp. Prog. Addison-Wesley, 1997, vol.2.

[26] S. Moshier, Methods and programs for mathematicalfunctions. Halstedpress, 1989.

[27] G. Marsaglia and W. W. Tsang, « Ziggurat Method for GeneratingRandom Variables » , journal of statistical software, vol. 5, no. 8, 2000.