REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE
UNIVERSITE MOHAMED BOUDIAF - M'SILA

FACULTE : Technologie
DEPARTEMENT : Electronique
N° :2018/ESEM01/87.

DOMAINE : Science et Technologie
FILIERE : Electronique
OPTION : Systèmes embarqués

Mémoire présenté pour l'obtention
Du diplôme de Master Académique

Par: Betka Ibrahim

Intitulé

Etude et Implémentation Pipeline sur FPGA de L'algorithme de Chiffrement AES

Soutenu devant le jury composé de:

Dr.L.LALAOUI	Université de M'sila	Président
Dr.M.BENAHCENE	Université de M'sila	Rapporteur
Dr.M.E.KETFI	Université de M'sila	Examinateur

Année universitaire : 2017 /2018

Dédicace

Je dédie ce modeste travail à tous ceux qui me sont chers:

A l'être le plus cher pour moi dans cette vie, ma mère et,
À l'être que je respecte le plus dans ce monde, mon cher père,

pour leur sacrifice et encouragement afin de finir ce travail. Je leur souhaite une vie plein de joie, de santé et de bonheur, que ALLAH me les garde pour le restant de ma vie.

À mes sœurs et mon frère

A tous mes amis et surtout mon frère Hamza

Remerciements

*Avant tout, nous remercions **Allah** le tout puissant de nous avoir donné la Patience, le courage et la volonté pour pouvoir terminer ce travail.*

Mon sincère remerciement à mon encadreur, M. BENAHCENE MADANI

Mes sincères remerciements à tous mes professeurs dans toute ma carrière académique.

Je voudrais remercier mes professeurs dans le département électronique

Pour tous ceux qui me veulent du bien, merci beaucoup

Pour tous ceux qui m'aiment .. pour tous ceux qui m'ont aidé merci

Résumé

Résumé

L'Institut national des normes et de la technologie (NIST) a lancé un processus d'élaboration d'une norme FIPS (Federal Information Processing Standard) pour la norme AES (Advanced Encryption Standard), spécifiant un algorithme de cryptage avancé pour remplacer le cryptage des données. 1998. Le NIST a sollicité des algorithmes candidats à l'inclusion dans AES, ce qui a donné lieu à quinze algorithmes candidats officiels parmi lesquels Rijndael a été choisi comme Advanced Encryption Standard.

L'Advanced Encryption Standard peut être programmé dans un logiciel ou construit avec du matériel pur. Cependant, les FPGA (Field Programmable Gate Arrays) offrent une solution plus rapide et plus personnalisable. Cette recherche examine l'algorithme AES en ce qui concerne le FPGA et le langage de description de matériel de circuit intégré à très grande vitesse (VHDL). Le logiciel ISE 14.2 est utilisé pour la simulation et l'optimisation du code VHDL synthétisable. Toutes les transformations de Cryptage et de décryptage sont simulées en utilisant une approche de conception itérative afin de minimiser la consommation de matériel en utilisant le simulateur lsim de la plateforme ise 14.2 de XILINX

Summary

The National Institute of Standards and Technology (NIST) has initiated a process to develop a Federal Information Processing Standard (FIPS) for the Advanced Encryption Standard (AES), which specifies an advanced encryption algorithm to replace the data encryption. 1998. NIST solicited candidate algorithms for inclusion in AES, resulting in fifteen official candidate algorithms from which Rijndael was chosen as the Advanced Encryption Standard.

The Advanced Encryption Standard can be programmed into software or built with pure hardware. However, Field Programmable Gate Arrays (FPGAs) offer a faster and more customizable solution. This research examines the AES algorithm for the FPGA and the High Speed Integrated Circuit (VHDL) hardware description language. ISE 14.2 software is used for simulation and optimization of synthesizable VHDL code. All Encryption and decryption transformations are simulated using an iterative design approach to minimize hardware consumption using the XILINX ise 14.2 lsim simulator

Table des matières

Liste des figures

Liste des tableaux

Table des Notations et symboles

AES :	Advanced Encryption Standard
CLB :	Configurable Logic Bloc
DES :	Data Encryption Standard
IP :	Intellectual Property
ISE :	Integrated Synthesis Environment
NIST :	National Institute of Standards and Technology
RSA :	Rivest-Shamir-Adleman
SSI	Small Scale Integration
MSI	Medium Scale Integration
LSI	Large Scale Integration
VLSI	Very Large Scale Integration
ASIC	Application Specific Integrated Circuits
PLA	Programmable Logic Array
PAL	Programmable Array Logic
PLD	Programmable Logic Device
FPLA	Field Programmable Logic Array
SPLD	Simple Programmable Logic Device Signal
OE	Output Enable
CPLD	Complex Programmable Logic Device
FPGA	Field Programmable Gate Array
LUT	Lock UpTable
OTP	One Time Programming
RAM	Random Access Memory
ROM	Read Only Mémory
EPROM	Erasable Programmable Read Only Mémory
MOS	Métal-Oxyde- Semi-conducteur
EEPROM	Electrically Erasable Programmable Read Only Memory
NOVRAM	NOn Volatile RAM
ISP	In Sito Programmation
OLMC	Output Logic MacroCell, dénomination Lattice
SRAM	Static Random Access Memory
VHDL	**V**ery High Speed Integrated Circuit **H**ardware **D**escription **L**angage
DSP	Digital Signal Processor
NRE	Les coûts d'ingénierie non récurrents
OEM	Original Equipment Manufecturer
PFU	Programmable Functional Unit
ECP	Electronic Check Presentment
FPSC	Field Programmable System Chip
CLB	Configurable Logic Block
ECU	Embedded Computational Units
LAB	Logic Array Blocs
CAO	La conception assistée par ordinateur
CTO	Chief Technology Officer
Co- Design	conception conjointe logiciel matériel
PCI	Peripheral Component Interconnect

PLL	Phase Locked Loops
DLL	*Delay Locked Loops*
MAC	Multiplieur Accumulateur
SOPC	System On a Programmable Chip
CPU	Central Processing Unit
ALM	Adaptive Logic Module
LEs	Logic Elements
LABs	Logic Array Blocs
TSMC	Taiwan Semiconductor Manufacturing Company
LUT	Look Up Table
RISC	Reduced instruction set computer
PIO	Parallel Input Output
mppSoC	parametric massive parallel
OFDMA	Orthogonal Frequency Division Multiple Access MIMO multiple-input multiple-output
IGBT	insulated-gate bipolar transistor
GFD	Graphe de Flot de Données
JTAG	Joint Test Action Group
MLI	modulation de largeur d'impulsions
PWM	pulse width modulation
SVpwm	space vector pulse width modulation

Introduction Générale

Introduction Générale

Le monde actuel connait de grands progrès technologiques dans le domaine des réseaux informatiques et des télécommunications. Ce qui a conduit à augmenter le volume d'échanges d'informations par différents opérateurs à travers le monde. Internet est certainement l'outil essentiel utilisé à cette fin par des centaines de millions d'institutions et d'individus. Cependant, face à cette ouverture, certaines questions doivent se poser concernant la sécurité des données. Que se passe-t-il lorsqu'un individu achète un billet d'avion sur Internet et doit rentrer des informations sensibles comme un numéro de carte de crédit ? Comment les mots de passe d'un utilisateur sont-ils enregistrés dans les différents sites auxquels il est inscrit. En effet, après les scandales liés à Wikileaks la divulgation d'informations sensibles du Gouvernement Américain par Edward Snowden ou encore le vol de données confidentielles de l'entreprise Sony, il est normal de se demander comment les données sont protégées ? Les différents gouvernements et les grandes entreprises dépensent énormément d'argent pour assurer la sécurité de leurs données. Malgré cela, les données arrivent à être dérobées.[1]

La cryptographie constitue un outil important pour la protection des données échangées contre les attaques et diminue le risque de vol de l'information. Elle rend l'information inintelligible et inexploitable par ses deux types de cryptographies symétrique et asymétrique. La cryptographie symétrique utilise une même clé de petite taille (128 à 256 bits) pour chiffrer et déchiffrer l'information ce qui la rend rapide et utilise des opérations simples tels que : des décalages, des permutations et des additions. Par contre la cryptographie asymétrique utilise une paire de clés de (1024 à 2048) bits : une clé publique pour le chiffrage et une clé privée pour le déchiffrage, ce qui entraine un très grand volume de calcul.

L'AES (Advanced Encryption Standard) est le protocole de chiffrement symétrique le plus utilisé actuellement. Il consiste en l'exécution de 10, 12 ou 14 rounds séquentiellement en utilisant des clés de 128, 192 ou 256 bits sur des blocs de données de 128 bits. Chaque round est constitué de 4 opérations SubBytes, ShiftRows, MixColumn et AddRoundKey. En pratique, il existe deux types d'implémentation de l'AES à savoir des méthodes softwares sur processeurs DSP ou microcontrôleurs ou des

méthodes hardwares sur circuits programmables CPLD ou FPGA. Accélérer l'AES revient à accélérer ces opérations en les implémentant sur matériel tel que l'utilisation des circuits programmables comme les FPGAs vu les avantages qu'ils offrent : reprogrammation, flexibilité et ressources matérielles. Encore, l'AES peut être implémenté sur matériel en utilisant les architectures Série/Série, Parallèle/Série où Parallèle/Pipeline offrant des niveaux de performances : faible, moyen et élevé.

Dans ce projet dont l'intitulé est: " implémentation sur circuit FPGA de l'algorithme de cryptage AES et", nous avons étudié les différentes fonctions de l'algorithme de chiffrement AES et les avons programmé en langage VHDL et implanté sur circuits FPGA de Xilinx.

Le but de ce projet est d'effectuer des simulations avec le simulateur lsim de la plate-forme ISE 14.2 de XILINX et Modelsim pour effectuer le choix du composant cible qui réalise un bon compromis entre la surface occupée par notre AES IP sur circuit FPGA et son temps d'exécution. Ce mémoire est structuré en quatre chapitres :

Le premier chapitre traite les concepts et les principes de base de la cryptographie où ses deux types : symétrique et asymétrique ont été exposés.

Le deuxième chapitre donne une description détaillée de l'algorithme AES ainsi que de ces transformations.

Le troisième chapitre décrit les circuits programmables FPGA et les outils nécessaires pour implémenter l'algorithme AES sur ces circuits.

Le dernier chapitre présente les résultats de simulation et d'implémentation de notre conception sur un circuit FPGA de Xilinx en utilisant l'environnement de conception ISE.

Enfin, notre rapport se termine par une conclusion générale et quelques perspectives.

Chapitre I

Notions de Cryptographie

1.1 Introduction

Actuellement, le réseau Internet est omniprésent. Que ce soit sur un smartphone, un ordinateur ou même sur un objet comme une montre, Internet est partout. Il possède des caractéristiques fascinantes qui n'ont pas besoin d'être décrites tellement son utilisation est rentrée dans les mœurs.

Toutefois, face à cette ouverture, certaines questions doivent se poser concernant la sécurité des données. Que se passe-t-il lorsqu'un individu achète un billet d'avion sur Internet et doit rentrer des informations sensibles comme un numéro de carte de crédit ? Comment les mots de passe d'un utilisateur sont-ils enregistrés dans les différents sites auxquels il est inscrit. En effet, après les scandales liés à Wikileaks la divulgation d'informations sensibles du Gouvernement Américain par Edward Snowden ou encore le vol de données confidentielles de l'entreprise Sony , il est normal de se demander comment les données sont protégées ? Les différents gouvernements et les grandes entreprises dépensent énormément d'argent pour assurer la sécurité de leurs données. Malgré cela, les données arrivent à être dérobées.[1]

Au vu de cette problématique, la cryptographie va être l'objet de ce travail. Si des données sont interceptées mais qu'elles sont chiffrées et donc incompréhensibles, alors ces données sont inutilisables. En revanche, si une méthode de cryptage pas très robuste a été utilisée, une personne malveillante pourrait retrouver les données d'origine. En partant du principe qu'Internet est un réseau ouvert où n'importe quelle communication peut être interceptée, la cryptographie prend alors tout son sens.[2]

Ce chapiter va tout d'abord parler de l'histoire de la cryptographie, afin de voir son évolution. La partie suivante va porter sur l'étude de différentes méthodes de cryptage. Pour finir, certains algorithmes vont être programmés afin de voir plus en détail leur fonctionnement.[3]

1.2Définition

Le terme cryptographie provient des deux mots grecs anciens « Kruptos » qui signifie « cacher » et « graphein » qui signifie « écrire » . Ce qui signifie littéralement, « cacher l'écriture ». Le Petit Larousse donne la définition suivante : « *Ensemble des techniques de chiffrement qui assurent l'inviolabilité de textes et, en informatique, de données.* » [4]

1.3Histoire de la cryptographie

L'être humain a toujours eu le besoin de cacher des informations. Que ce soit un secret ne devant pas être divulgué dans son entourage qui compromettrait des individus ou encore d'informations tactiques lors des différentes batailles et guerres ayant marqué l'Histoire. Dissimuler des informations a toujours été une nécessité. Voici une liste non exhaustive de différentes techniques utilisées au fil des siècles qui marquent l'évolution de la cryptographie à travers les âges.

Les premières traces de cryptographie remontent à l'Antiquité, entre le X^{ème} et le VII^{ème} siècle avant J.-C., les Grecs utilisaient des scytales, des sortes de bâtons en bois. Quand l'émetteur voulait communiquer, il enroulait une bande de cuir sur la scytale et y inscrivait le message (une lettre par bout de bande). Une fois la bande déroulée, les lettres n'étaient plus ordonnées et n'avaient donc plus aucun sens. Le seul moyen de pouvoir comprendre le message était d'enrouler la bande sur une scytale de même diamètre pour que les lettres puissent s'aligner correctement. [5] [6]

Figure 1.1 : Une scytale

Au I^{er} siècle avant J.-C., le cryptage de César faisait son apparition. Ce chiffrement était utilisé par Jules César pour communiquer de façon secrète. C'est un des premiers chiffrements par substitution. Son principe est simple, il suffit de substituer chaque caractère du message d'origine par un autre dans l'alphabet, qui se trouve toujours à une distance fixe. [7]

Au fur et à mesure que le temps passe, comme les méthodes utilisées précédemment deviennent de plus en plus faciles à comprendre et à détourner, d'autres mesures ont dû être prises afin de pouvoir empêcher le décryptage des messages. Au XVI^{ème} siècle, le chiffre de Vigenère fit son apparition.

Il s'agit également d'un chiffrement par substitution, mais il est plus avancé que le chiffrage de César. Plutôt que d'utiliser un décalage fixe, le chiffre de Vigenère se base sur une clé qui va déterminer le décalage pour chaque caractère. [8]

A partir du siècle dernier, la cryptographie a joué un rôle clé lors des différentes guerres. La technologie a commencé à être utilisée avec par exemple, les messages radio ou les télégrammes. Les informations pouvaient être beaucoup plus facilement interceptées que dans le passé. Ce fut le cas du ministre des affaires étrangères allemand Arthur Zimmermann, le 16 janvier 1917. Il a envoyé un télégramme à l'ambassadeur allemand au Mexique pour lui proposer un complot afin de garder les Etats-Unis hors de la guerre. Cependant, le message a été intercepté par les Britanniques car les Allemands ont utilisé le « code diplomatique 0075 » qui était un code de substitutions comprenant 1000 substitutions possibles (voir Annexe 1). [9]

Après la première Guerre Mondiale, la machine Enigma a été créée et les Allemands, ayant compris l'importance que peuvent avoir les informations sensibles, ont investi dans une version militaire plus complexe de cette machine. Toutefois, bien que le fonctionnement de la machine soit complexe, des chercheurs polonais ont étudié le fonctionnement de la machine pour tenter de décrypter les messages. Par la suite, le célèbre mathématicien, cryptologue et informaticien britannique Alan Turing a collaboré au décryptage des messages codés par la machine. Grâce à cela, les Britanniques ont pu déchiffrer des messages, ce qui a été un sérieux avantage qui leur a permis de gagner la guerre. Il a
même été estimé que la découverte de Turing a raccourci la deuxième guerre mondiale de deux ans[10].

Figure 1.2 : Machine Enigma [11]

En 1977, le standard de chiffrement DES est proposé comme standard par le NIST. C'est un chiffrement par bloc de textes de 56 bit (en réalité 64, mais un bit est utilisé pour contrôler la parité). Il est resté comme standard jusqu'à la fin des années 1990.
En effet, à la fin des années 70, les ordinateurs n'étaient pas suffisamment puissants pour déchiffrer un message codé avec DES, mais avec la multiplication de leur puissance, le déchiffrage est devenu possible dans un temps raisonnable. Une information chiffrée en DES peut être déchiffrée en 30h avec 1000 PC en parallèle cadencés à 1GHz. C'est pourquoi, à la fin des années 90, le NIST a lancé un concours pour remplacer le DES et définir un nouveau standard de cryptage qui deviendra AES[3]. Les gagnants sont les deux chercheurs belges Joan Daemen et Vincent Rijmen avec leur chiffrement Rijndael. [12]

A l'époque du DES, plus précisément en 1976, les deux cryptologues Whitfield Diffie et Martin Hellman publient un article « New Directions in Cryptography » [13] qui propose une nouvelle façon de crypter les données. C'est à partir de leur article que le cryptage asymétrique est né avec le chiffrement de Diffie-Hellman. Jusque-là, tous les cryptosystèmes étaient symétriques. Bien que ce type de cryptage fonctionne bien, garder la clé secrète ou bien tout simplement la communiquer peut devenir contraignant. Avec leur système, le problème de clé secrète n'est plus un problème car le fonctionnement se base sur une clé pouvant être connue de tout le monde.

1.4. Définition de la clé

Une ***clé*** est un paramètre utilisé en entrée d'une opération cryptographique (chiffrement, déchiffrement, signature numérique, vérification de signature). Elle doit être variable et maintenue en secret (sauf dans certains algorithmes où une part de la clé reste exposée). Elle peut se présenter sous plusieurs formes : mots ou phrases, procédure pour préparer une machine de chiffrement (connexions, câblage, etc.), données codées sous une forme binaire (cryptologie moderne). La protection apportée par un algorithme de chiffrement est liée à la longueur de la clé, qui peut s'exprimer en bits. Les clés les plus grandes resteront crypto graphiquement sûres pour une plus longue période. Si ce que l'on chiffre doit rester caché pendant de nombreuses années, il faut utiliser une clé très grande.

1.5. Différents types de cryptographie

Les techniques cryptographiques se découpent en deux grandes parties :

- ✓ La cryptographie à clés secrètes ou cryptographie symétrique.
- ✓ La cryptographie à clés publiques ou cryptographie asymétrique.

Ils ont tous deux leurs avantages et leurs inconvénients. La différence qui existe entre ces deux types se situe au niveau de la clé.

1.6.La cryptographie

1.6.1 Cryptographie symétrique

La cryptographie symétrique (ou cryptographie à clé secrète) est la forme la plus ancienne de cryptographie [14]. Ce chiffrement fonctionne en principe avec une clé secrète, bien qu'il existe certains chiffrements symétriques qui n'utilisent pas de clé, comme par exemple le chiffre de César. Dans le cas des chiffrements avec clé, le principe est le suivant : L'émetteur du message chiffre les données grâce à une clé. Cette clé est généralement une chaîne de caractères. Le message est chiffré et sans la clé il est quasi impossible (le niveau d'impossibilité dépend du niveau de protection du chiffrement utilisé ainsi que de la complexité de la clé utilisée) de retrouver le message d'origine. L'émetteur doit donc transmettre la clé aux personnes à qui il désire transmettre le message s'il veut que son message puisse être lu.

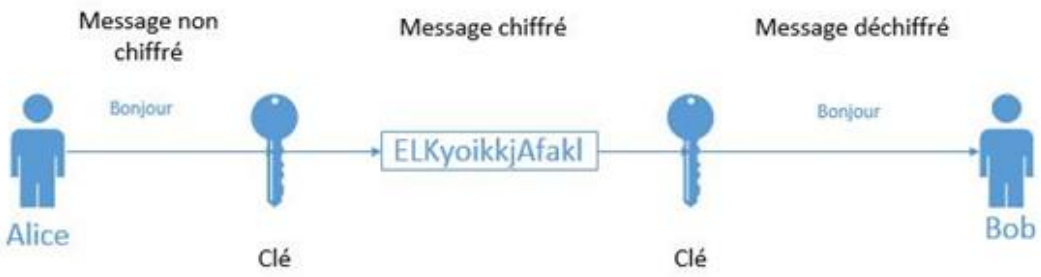


Figure 1.3 : Schéma de fonctionnement de la cryptographie symétrique

1.2.2 Cryptographie asymétrique

La cryptographie asymétrique ou cryptographie à clé publique fonctionne de façon totalement différente à la cryptographie symétrique. Si l'on peut comparer la cryptographie symétrique à un coffre-fort auquel seules les personnes possédant la clé peuvent accéder, la cryptographie asymétrique pourrait être comparée à une boite aux lettres dans laquelle on peut déposer des informations, et seule la personne possédant la clé peut accéder au contenu de la boîte. La boite aux lettres serait la clé publique (donc accessible à tout le monde), alors que la clé pour ouvrir la boite serait la clé privée. En effet, dans la cryptographie asymétrique, il y a une clé publique et une clé privée.[15]

Les nombres premier sont l'élément clé pour rendre les algorithmes de cryptographie asymétrique indéchiffrables (ou presque). En prenant comme exemple la multiplication de 5 x 7, il est assez facile de répondre instantanément 35. L'opération inverse, à savoir, retrouver 35 à partir de facteurs premiers est assez facile. Cependant, factoriser 1591 par exemple, est beaucoup plus compliqué, alors que calculer 37 x 43 se fait très facilement. C'est sur cette difficulté de factorisation que se reposent les algorithmes asymétriques. [15]

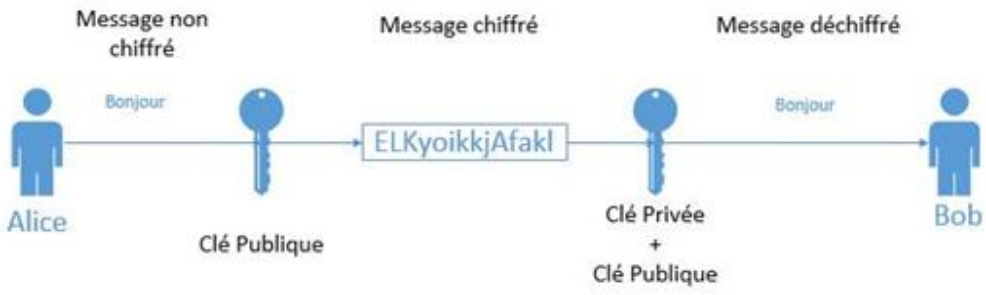


Figure 1.4 : Schéma de fonctionnement de la cryptographie asymétrique

1.7. Comparaison entre les chiffrements symétrique et asymétrique

Les chiffrements symétriques et asymétriques présentent chacun des avantages et des inconvénients.

La principale difficulté des algorithmes de chiffrement symétrique ou à clés secrètes réside dans la sécurité de l'échange des clés. Un autre inconvénient est que tout couple d'utilisateurs doit au préalable s'entendre sur une clé commune. La gestion des clés devient vite problématique. Toutefois, les systèmes symétriques sont très efficaces.

Ils demeurent sûrs, rapides et peuvent chiffrer et déchiffrer une grande quantité de données en des temps records.

Les crypto systèmes asymétriques actuels sont beaucoup plus lents que leurs homologues symétriques, et nécessitent de plus longues clés. Leur avantage principal réside dans la simplicité de la gestion des clés : le secret n'est pas partagé, les clés publiques peuvent être publiées dans l'annuaire, et le nombre de clés est bien inférieur lorsque plusieurs personnes veulent communiquer entre elles de façon confidentielle .

Le tableau 1 récapitule les avantages et inconvénients des chiffrements symétrique et asymétrique.

	Symétrique	Asymétrique
Avantages	- Rapidité, jusqu'à 100 fois plus rapide. - Facilité d'implémentation sur Hardware. - Taille de clés 128 bits (implique 16caractères mémorisable).	Distribution des clés facilitée :pas d'authentification. - Permet de signer des messages facilement. - Nombre de clés à distribuer est réduit par rapport aux clés symétriques.
Inconvénients	-Nombre de clés à gérer. - Distribution des clés (authentification, confidentialité). - Certaines propriétés sont difficiles à réaliser	- Taille des clés. -Vitess de chiffrement.

Tableau 1.1. Avantages et Inconvénients des chiffrements symétrique et Asymétrique [16].

1.8. Méthodes de chiffrement

1.8.1. Chiffrement par flot

Les algorithmes basés sur le principe de chiffrement par flot chiffrent ou déchiffrent un message à la volée. Leur fonctionnement se base sur un générateur de nombres pseudo-aléatoires et un mécanisme de substitution bit à bit. Les algorithmes se basant sur ce principe sont réputés rapides. [12]

1.8.2Chiffrement par blocs

Le chiffrement par blocs fonctionne différemment. Au lieu de prendre chaque bit un par un, les messages sont découpés en blocs (la taille des blocs dépend de la clé).

Ensuite, chaque bloc est additionné à la clé et un traitement de type permutation, opération XOR ou autre est appliqué à chaque bloc. [12]

1.9 Hachage

Les fonctions de hachage permettent de chiffrer un message sous la forme d'une chaîne de caractères de taille fixe, peu importe la taille du message d'origine, généralement entre 128 et 512 bits. Elles sont comparables à une empreinte, car un message aura toujours la même empreinte en appliquant une fonction de hachage. Elles sont à sens unique, ce qui signifie qu'il est facile de hacher un message, mais qu'il n'est en principe pas possible de calculer son inverse, à moins d'utiliser une méthode de force brute. Pour qu'une fonction de hachage soit sûre, elle doit être résistante aux collisions. En partant du principe qu'il y a une infinité de messages possibles pouvant être chiffrés, il est évident que plusieurs messages vont donner le même résultat une fois hachés. La résistance est le fait que les collisions ne puissent pas être retrouvées. [18]

1.10. Cryptographie quantique

La cryptographie quantique se base sur les propriétés de la mécanique quantique. Plutôt que d'essayer de chiffrer des données afin qu'elles soient incompréhensibles, la communication est arrêtée si un message a été intercepté. C'est donc de cette façon que fonctionne la cryptographie quantique. Selon le principe d'incertitude d'Heisenberg, si un objet quantique (dans ce cas, un photon circulant dans une fibre optique) a été observé alors il a été modifié et donc la communication est coupée. [19]

Le papier « La cyptographie quantique en bref » [19] résume ce type de cryptage avec un exemple : un joueur de tennis lance avec sa raquette, des balles contenant chacune un message. Si une personne souhaite intercepter les balles, elle pourrait le faire. Mais si au lieu de lancer des balles, le joueur de tennis lançait des bulles en savon ? Dans ce cas-là, si quelqu'un tente de les intercepter, elles exploseraient sur le coup car elles sont fragiles.

Figure 1.5: Illustration du fonctionnement de la cryptographie quantique [19]

1.11Conclusion

La complexité des algorithmes de cryptage n'a cessé d'augmenter à travers les âges. A ses débuts, un bâton permettait de chiffrer des données. Bien qu'actuellement ce procédé paraisse rudimentaire, il fonctionnait très bien car à cette époque, beaucoup de personnes étaient illettrées (ce qui était déjà une barrière pour déchiffrer un message codé) et la cryptographie était une science non connue. Par la suite, des méthodes manipulant les caractères d'un message ont commencé à prendre place. D'abord avec des chiffrements monoalphabétiques et par la suite avec des chiffrements polyalphabétiques. Toutefois, bien que certaines méthodes polyalphabétiques soient jugées encore sûres de nos jours, la cryptographie a pris un sérieux tournant avec le début de la technologie. Les décalages de caractères des anciennes méthodes ont laissé place à des méthodes ingénieuses combinant les caractères dans tous les sens. Toutefois, certaines de ces méthodes modernes ont atteint leurs limites. Le code Enigma paraissait indéchiffrable et pourtant il a été déchiffré. Le chiffrement DES a aussi connu le même sort. Il paraissait indéchiffrable et pourtant le NIST a dû lancer le concours AES afin de trouver un successeur suffisamment sûr.

La technologie étant de plus en plus présente, les experts en cryptographie ont réfléchi une nouvelle manière de pouvoir communiquer des clés de chiffrement. En effet, transmettre une clé à une personne peut être assez facile. Transmettre une clé à beaucoup de personnes est plus compliqué. De plus, les messages peuvent être interceptés. C'est pour cette raison que des protocoles tels que Diffie-Hellman ont fait leur apparition. Il est à noter que contrairement aux autres algorithmes symétriques cités plus haut dans cette conclusion, cette nouvelle façon, asymétrique, de chiffrer les données a été dès le départ beaucoup plus complexe que les scytales jadis utilisées. Ces cryptologues avaient déjà la connaissance du domaine et ont cherché à combler le problème de transmission de clés à grande échelle.

Actuellement, AES et RSA sont des standards en termes de cryptographie. Cependant, bien qu'ils soient jugés sûrs, l'évolution de la cryptographie a prouvé que les algorithmes ont toujours fini par atteindre leurs limites avec l'évolution de la technologie. Donc finalement, si la technologie permet d'augmenter la complexité des méthodes de chiffrement, mais que parallèlement elle permet de casser des messages, est-ce que la technologie est-elle son propre ennemi ?

Chapitre II

L’Algorithme de Cryptage et Décryptage AES

Chapitre II

L'Algorithme de Cryptage et Décryptage AES

2.1. INTRODUCTION

L'Institut national des normes et de la technologie (NIST) a sollicité des propositions pour l'Advanced Encryption Standard (AES). L'AES est une norme fédérale de traitement de l'information (FIPS), qui est un algorithme cryptographique utilisé pour protéger les données électroniques. L'algorithme AES est un chiffrement de bloc symétrique qui peut crypter, (chiffrer) et décrypter (déchiffrer) des informations. Le cryptage convertit les données en une forme inintelligible appelée texte chiffré. Le décryptage du texte chiffré convertit les données dans leur forme d'origine, appelée texte en clair. L'algorithme AES est capable d'utiliser des clés cryptographiques de 128, 192 et 256 bits pour crypter et décrypter des données dans des blocs de 128 bits.

De nombreux algorithmes ont été présentés à l'origine par des chercheurs de douze pays différents. Quinze (15) algorithmes ont été sélectionnés à partir du premier ensemble de soumissions. Après un processus d'étude et de sélection, cinq (5) ont été choisis comme finalistes. Les cinq algorithmes sélectionnés étaient MARS, RC6, RIJNDAEL, SERPENT et TWOFISH. La conclusion était que les cinq concurrents présentaient des caractéristiques similaires. Le 2 octobre 2000, le NIST a annoncé que l'algorithme Rijndael était le gagnant du concours. L'algorithme de Rijndael a été choisi car il a obtenu les meilleurs scores globaux en matière de sécurité, de performance, d'efficacité, de capacité d'implémentation et de flexibilité. L'algorithme de Rijndael était développé par Joan Daemen de Proton World International et Vincent Fijmen de l'Université Katholieke de Louvain.

L'algorithme de Rijndael est un chiffrement de bloc symétrique qui peut traiter des blocs de données de 128 bits en utilisant des clés de chiffrement de 128, 192 et 256 bits. L'algorithme de Rijndael a également été conçu pour gérer des tailles de blocs et des longueurs de clé supplémentaires. Cependant, les fonctionnalités supplémentaires n'ont pas été adoptées dans l'AES. L'implémentation matérielle de l'algorithme de Rijndael peut fournir des performances élevées ou des coûts réduits pour des applications spécifiques. Dans les canaux de communication backbone ou les serveurs fortement chargés, il n'est pas possible de perdre la vitesse de traitement, ce qui réduit l'efficacité du système global tout en exécutant des algorithmes de cryptographie dans le logiciel. De l'autre côté, un faible coût et une petite conception peuvent être utilisés dans

les applications de cartes à puce, ce qui permet à un large éventail d'équipements de fonctionner en toute sécurité. [21]

2.2. Notation et conventions

2.2.1. Entrées et sorties

L'entrée et la sortie de l'algorithme AES sont constituées de séquences de 128 bits. Ces séquences sont appelées blocs et les nombres de bits qu'elles contiennent sont appelés leur longueur. La clé de chiffrement pour l'algorithme AES est une séquence de 128, 192 ou 256 bits. Les autres longueurs d'entrée, de sortie et de clé de chiffrement ne sont pas autorisées par cette norme. Les bits à l'intérieur de telles séquences sont numérotés à partir de zéro et se terminant à un de moins que la longueur de la séquence, qui est également appelée longueur de bloc ou longueur de clé. Le nombre "i" attaché à un bit est connu comme son index et sera dans l'une des gammes $0 \leq i < 128$, $0 \leq i < 192$ ou $0 \leq i < 256$ en fonction de la longueur de bloc ou de la longueur de clé spécifiée.[21]

2.2.2. Octets

L'unité de traitement de base dans l'algorithme AES est un octet, qui est une séquence de huit bits traités comme une seule entité. Les séquences de bits d'entrée, de sortie et de clé de chiffrement décrites au paragraphe 1.1 sont traitées comme des tableaux d'octets formés en divisant ces séquences en groupes de huit bits contigus pour former des tableaux d'octets. Pour une entrée, une sortie ou une clé de chiffrement notée a, les octets du tableau résultant sont référencés en utilisant l'une des deux formes, a_n ou a [n], où n sera dans une plage qui dépend de la longueur de la clé. Pour une longueur de clé de 128 bits, n se situe dans la plage $0 \leq n < 16$. Pour une longueur de clé de 192 bits, n se situe dans la plage $0 \leq n < 24$. Pour une longueur de clé de 256 bits, n se trouve dans gamme $0 < n < 32$.

Toutes les valeurs d'octets dans l'algorithme AES sont présentées comme la concaténation des valeurs de bits individuelles, (0 ou 1), entre accolades dans l'ordre $\{b_7, b_6, b_5, b_4, b_3, b_2, b_1, b_0\}$.

Ces octets sont interprétés comme des éléments de corps finis utilisant une représentation polynomiale[22]

$$b_7 x^7 + b_6 x^6 + b_5 x^5 + b_4 x^4 + b_3 x^3 + b_2 x^2 + b_1 x + b_0 = \sum_{i=0}^{7} b_i x^i \quad (1)$$

Par exemple, {01100011} identifie l'élément de champ fini spécifique $x^6 + x^5 + x + 1$. Il est également pratique de désigner des valeurs d'octet en notation hexadécimale avec chacun des deux groupes de quatre bits étant désigné par un seul caractère hexadécimal. Le schéma de notation hexadécimal est représenté sur la Figure.2.1.

Modèle de bits	**caractère**	**Modèle de bits**	caractère	**Modèle de bits**	Caractère	**Modèle de bits**	caractère
0000	0		4		8		c
0001	1		5		9		d
0010	2		6		A		e
	3		7		B		F

Tableau 2.1. Représentation hexadécimale des formes de bits [22]

Par conséquent, l'élément {01100011} peut être représenté par {63}, où le caractère désignant le groupe de quatre bits contenant les bits les plus élevés est de nouveau à gauche. Certaines opérations de champ finis impliquent un bit supplémentaire {b8} à gauche d'un octet de 8 bits. Lorsque le bit b8 est présent, il apparaît comme {01} précédant immédiatement l'octet de 8 bits. Par exemple, une séquence de 9 bits est présentée comme {01} {1b}.

2.2.3. Tableaux d'octets

Les tableaux d'octets sont représentés sous la forme $a_0\ a_1\ a_2$ ••• a_{15}. Les octets et l'ordre des bits dans les octets sont dérivés de la séquence d'entrée de 128 bits, input0input1input2 ••• entrée126 entrée127 comme a_0 = {entrée0, entrée1, •••, entrée7}, a_1 = {entrée8, entrée9, •••, entrée15} avec le motif se prolongeant jusqu'à . a_{15}={entrée120, entrée121, •••, entrée127 } Le motif peut être étendu à des séquences plus longues associées à des clés 192 et 256 bits. Engénéral,

a_n = {input8n, input8n + 1, •••, input8n + 7}.

Un exemple de désignation d'octet et de numérotation à l'intérieur d'octets pour une séquence d'entrée donnée est présenté à la tableau 2.2.

séquences de bits d'entrée	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	15.	17..	18..	19..	20..	21..	22..	23.	..
numéro d'octet	0								1								2								
nombre de bits en octets	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	

Tableau 2.2. Indices pour les octets et les bits [22]

2.2.4. L'état

En interne, les opérations de l'algorithme AES sont effectuées sur un tableau bidimensionnel d'octets appelé State. L'état se compose de quatre rangées d'octets. Chaque ligne d'un état contient N_b nombres d'octets, où N_b est la longueur de bloc divisée par 32. [24]Dans le tableau d'états, qui est désigné par le symbole **S**, chaque octet individuel a deux indices. Le premier indice d'octet est le numéro de ligne r, qui se situe dans la plage $0 \leq r \leq 3$ et le second octet index est le numéro de colonne c, qui se situe dans la plage $0 \leq c \leq N_b\text{-}1$. Une telle indexation permet de désigner un octet individuel de l'État comme Sr, c ou S [r, c]. Pour l'AES $N_b = 4$, ce qui signifie que $0 \leq c \leq 3$. Au début du cryptage et du décryptage, l'entrée, qui est le tableau d'octets symbolisé par in0 in_1 ••• in_{15}, est copiée dans le tableau State. Cette activité est illustrée à la Figure 2.1. Les opérations de cryptage ou de décryptage sont effectuées sur le tableau d'états. Une fois la manipulation du tableau d'états terminée, sa valeur finale est copiée dans la sortie, qui est un tableau d'octets symbolisé par out $_0$$out_1$ ••• out_{15}.

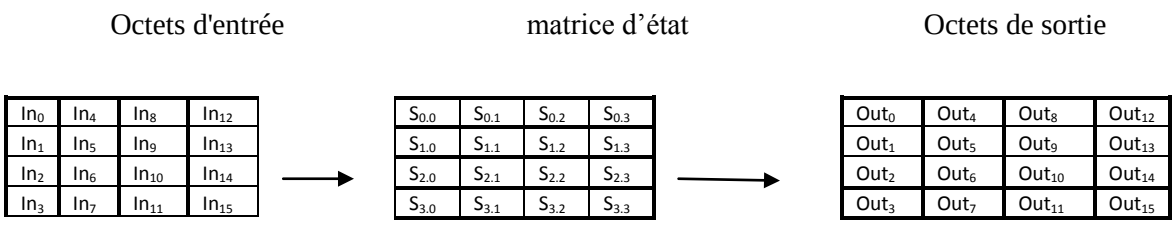


Figure 2.1. Entrée et sortie du tableau d'état [22]

Au début du cryptage ou du déchiffrement, le tableau d'entrée est copié dans l'état tableau avec

$$S[r, c] = in[r + 4c]$$

Où $0 \leq r \leq 3$ et $0 \leq c \leq Nb-1$ À la fin du cryptage et du décryptage, l'État est copié dans le tableau de sortie avec

$$out[r + 4c] = S_{[r,c]}$$

Où $0 \leq r \leq 3$ et $0 \leq c \leq N_{b-1}$

2.2.5. L'État en tant que tableau de colonnes

Les quatre octets dans chaque colonne de l'état forment des mots de 32 bits, où la ligne ,le nombre "r" fournit un index pour les quatre octets dans chaque mot. Par conséquent, l'état peut être interprété comme un tableau unidimensionnel de mots de 32 bits, qui est symbolisé par w_0 ... w_3.[22][23] Le numéro de colonne **c** fournit un index dans ce tableau d'états linéaire. Considérant l'État représenté à la figure 3, l'État peut être considéré comme un tableau de quatre mots

$$w_0 = S_{0,0}\ S_{1,0}\ S_{2,0}\ S_{3,0},$$
$$w_1 = S_{0,1}\ S_{1,1}\ S_{2,1}\ S_{3,1},$$
$$w_2 = S_{0,2}\ S_{1,2}\ S_{2,2}\ S_{3,2}$$

et

$$w_3 = S_{0,3}\ S_{1,3}\ S_{2,3}\ S_{3,3}.$$

2.3. Contexte mathématique

Chaque octet de l'algorithme AES est interprété comme un élément de corps fini utilisant le notation introduite dans la section.1.1.2. Tous les éléments de champ finis peuvent être ajoutés et multiplié. Cependant, ces opérations diffèrent de celles utilisées pour les nombres et leur utilisation nécessite une enquête.

2.3.1. Une addition

L'addition de deux éléments dans un corps fini est obtenue en "ajoutant" coefficients pour les puissances correspondantes dans les polynômes pour les deux éléments. Le l'addition est effectuée en utilisant l'opération XOR, qui est indiquée par le symbole de l'opérateur $\oplus$. Une telle addition est réalisée modulo-2. En addition modulo-2

$$1 \oplus 1 = 0,$$
$$1 \oplus 0 = 1,$$

$$0 \oplus 1 = 1$$

Et

$$0 \oplus 0 = 0.$$

Par conséquent, la soustraction de polynômes est identique à l'addition de polynômes. En variante, l'addition d'éléments de corps finis peut être décrite comme l'addition modulo-2 de bits correspondants dans l'octet. Pour deux octets {a7a6a5a4a3a2a1a0} et {b7b6b5b4b3b2b1b0}, la somme est {c7c6c5c4c3c2c1c0}, où chaque ci = $a_i \oplus b_i$ où i représente les bits correspondants. Par exemple, les expressions suivantes sont équivalentes entre elles.[22][23]

$(x^6 + x^4 + x^2 + x + 1) + (x^7 + x + 1) = x^7 + x^6 + x^4 + x^2$ (Notation polynomiale)

{01010111} ⊕ {10000011} = {11010100}(Notation binaire)

{57} ⊕ {83} = {*d* 4}(Notation hexadécimale)

2.3.2. Multiplication

Dans la représentation polynomiale, la multiplication dans le champ de Galois GF (2^8) (notée par •) correspond à la multiplication de polynômes modulo un polynôme irréductible de degré 8. Un polynôme est irréductible si ses seuls diviseurs sont un et lui-même.[22][23] Pour l'algorithme AES, ce polynôme irréductible est donné par l'équation (2).

$$m(x) = x^8 + x^4 + x^3 + x + 1 \qquad (2)$$

Par exemple, {57}•{83} = {*c*1} parce que

$$(x^6 + x^4 + x^2 + x + 1)(x^7 + x + 1) = x^{13} + x^{11} + x^9 + x^8 + x^7 + x^7 + x^5 + x^3 + x^2 + x + x^6 + x^4 + x^2 + x + 1 = x^{13} + x^{11} + x^9 + x^8 + x^6 + x^5 + x^4 + x^3 + 1$$

$$x^{13} + x^{11} + x^9 + x^8 + x^6 + x^5 + x^4 + x^3 + 1 \text{ Modulo } (x^8 + x^4 + x^3 + x + 1) = x^7 + x^6 + 1.$$

La réduction modulaire de m (x) garantit que le résultat sera un polynôme binaire de degré inférieur à 8, qui peut être représenté par un octet. Contrairement à l'addition, il n'y a pas d'opération simple au niveau des octets qui corresponde à cette multiplication. La multiplication définie ci-dessus est associative et l'élément {01} est l'identité multiplicative. Pour tout polynôme binaire b (x) non nul de degré inférieur à 8, l'inverse multiplicatif de b(x), noté b^{-1} (x), peut être trouvé. L'inverse est trouvé à traversutilisation de l'algorithme euclidien étendu pour calculer les polynômes a (x) et c(x) tels que

$$b(x)a(x) + m(x)c(x) = 1. \qquad (3)$$

Par conséquent, a (x) • b (x) mod m (x) = 1, ce qui signifie

$$b^{-1}(x) = a(x) \bmod m(x) \qquad (4)$$

De plus, pour tout a (x), b (x) et c (x) dans le domaine, il est vrai que

a (x) • (b (x) + c (x)) = a (x) • b (x) + a (x) • c (x)

Il s'ensuit que l'ensemble des 256 valeurs d'octets possibles, avec XOR utilisé comme addition et multiplication définies comme ci-dessus, a la structure du corps fini GF (2^8).[22][23]

2.3.3. Multiplication par x

Multiplier le polynôme binaire défini dans l'équation (1) avec le polynôme x résulte en

$$b_7 x^8 + b_6 x^7 + b_5 x^6 + b_4 x^5 + b_3 x^4 + b_2 x^3 + b_1 x^2 + b_0 x. \quad (6)$$

Le résultat x • b (x) est obtenu en réduisant le résultat ci-dessus modulo m (x). Si b_7 est égal à zéro, le résultat est déjà réduit. Si b_7 est égal à un, la réduction est obtenue en soustrayant le polynôme m (x). Il s'ensuit que la multiplication par x, qui est représentée par {00000010} ou {02}, peut être implémentée au niveau de l'octet comme un décalage vers la gauche et un XOR conditionnel subséquent avec {1b}. Cette opération sur bytes est notée xtime ().[22][23] La multiplication par des puissances plus élevées de x peut être implémentée par une application répétée de xtime (). Grâce à l'addition de résultats intermédiaires, la multiplication par n'importe quelle constante peut être implémentée. [23]

Par exemple, {57} • {13} = {fe} parce que

{57} • {02} = xtime ({57}) = {ae}

{57} • {04} = xtime ({ae}) = {47}

{57} • {08} = xtime ({47}) = {8e}

{57} • {10} = xtime ({8e}) = {07},

Ainsi

{57} • {13} = {57} • ({01} • {02} • {10})

= {57} • {ae} • {07}
= {fe}.

2.3.4. Polynômes avec coefficients dans GF (2^8)

Les polynômes à quatre termes peuvent être définis avec des coefficients qui sont des éléments de corps finis comme l'équation suivante (7)

$$a(x) = a_3x^3 + a_2x^2 + a_1x + a_0 \qquad (7)$$

qui sera noté comme un mot sous la forme [a0, a1, a2, a3]. Notez que les polynômes de cette section se comportent un peu différemment des polynômes utilisés dans la définition des éléments de corps finis, même si les deux types de polynômes utilisent le même indéterminé, x. Les coefficients de cette section sont eux-mêmes des éléments de corps finis, c'est-à-dire des octets, au lieu de bits; De plus, la multiplication des polynômes à quatre termes utilise un polynôme de réduction différent, défini ci-dessous. Pour illustrer les opérations d'addition et de multiplication,[22][23]

$$b(x) = b_3\,x^3 + b_2\,x^2 + b_1\,x + b \qquad (8)$$

définir un deuxième polynôme à quatre termes. L'addition est effectuée en ajoutant les coefficients de corps finis de puissances similaires de x. Cette addition correspond à une opération XOR entre les octets correspondants dans chacun des mots - en d'autres termes, le XOR des valeurs de mots complètes Ainsi, en utilisant les équations de (7) et (8),

$$a(x) + b(x) = (a_3 \oplus b_3)x^3 + (a_2 \oplus b_2)x^2 + (a_1 \oplus b_1)x + (a_0 \oplus b_0) \qquad (9)$$

La multiplication est réalisée en deux étapes. Dans la première étape, le produit polynomial c (x) = a (x) • b (x) est algébriquement étendu, et comme les pouvoirs sont recueillis pour donner

$$c(x) = c_6x^6 + c_5x^5 + c_4x^4 + c_3x^3 + c_2x^2 + c_1x + c_0 \qquad (10)$$

Où

$$c_0 = a_0 \bullet b_0$$

$$c_1 = a_1 \bullet b_0 \oplus a_0 \bullet b_1$$

$$c_2 = a_2 \bullet b_0 \oplus a_1 \bullet b_1 \oplus a_0 \bullet b_2$$

$$c_3 = a_3 \bullet b_0 \oplus a_2 \bullet b_1 \oplus a_1 \bullet b_2 \oplus a_0 \bullet b_3$$

$$c_4 = a_3 \bullet b_1 \oplus a_2 \bullet b_2 \oplus a_1 \bullet b_3$$

$$c_5 = a_3 \bullet b_2 \oplus a_2 \bullet b_3$$

$$c_6 = a_3 \bullet b_3$$

Le résultat, c (x), ne représente pas un mot de quatre octets. Par conséquent, la deuxième étape de la multiplication est de réduire c (x) modulo un polynôme de degré 4; le résultat peut être réduit à un polynôme de degré inférieur à 4.[22][23] Pour l'algorithme AES, ceci est accompli avec le polynôme $x^4 + 1$, de sorte que

$$X^i \bmod (x^4+1) = x^{i \bmod 4} \qquad (11)$$

Le produit modulaire de a (x) et b (x), noté a (x) • b (x), est donné par les quatre termes polynôme d (x), défini comme suit

$$d(x) = d_3 x^3 + d_2 x^2 + d_1 x + d_0 \qquad (12)$$

avec

$$d_0 = (a_0 \bullet b_0) \oplus (a_3 \bullet b_1) \oplus (a_2 \cdot b_2) \oplus (a_1 \cdot b_3)$$

$$d_1 = (a_1 \bullet b_0) \oplus (a_0 \bullet b_1) \oplus (a_3 \bullet b_2) \oplus (a_2 \bullet b_3)$$

$$d_2 = (a_2 \bullet b_0) \oplus (a_1 \bullet b_1) \oplus (a_0 \bullet b_2) \oplus (a_3 \bullet b_3)$$

$$d_3 = (a_3 \bullet b_0) \oplus (a_2 \bullet b_1) \oplus (a_1 \bullet b_2) \oplus (a_0 \bullet b_3)$$

Lorsque a (x) est un polynôme fixe, l'opération définie dans l'équation (12) peut être écrite sous forme matricielle comme l'équation suivante (13).

$$\begin{bmatrix} d_0 \\ d_1 \\ d_2 \\ d_3 \end{bmatrix} \begin{bmatrix} a_0 & a_3 & a_2 & a_1 \\ a_1 & a_0 & a_3 & a_2 \\ a_2 & a_1 & a_0 & a_3 \\ a_3 & a_2 & a_1 & a_0 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} \qquad (13)$$

Parce que $x^4 + 1$ n'est pas un polynôme irréductible sur GF (2^8), la multiplication par un polynôme fixe à quatre termes n'est pas nécessairement inversible.[22][23] Cependant, l'algorithme AES spécifie un polynôme fixe à quatre termes qui a un inverse est donné par

$$a(x) = \{03\}x^3 + \{01\}x^2 + \{01\}x + \{02\} \qquad (14)$$

$$a^{-1}(x) = \{0b\}x^3 + \{0d\}x^2 + \{09\}x + \{0e\} \qquad (15)$$

Un autre polynôme utilisé dans l'algorithme AES a a0 = a1 = a2 = {00} et a3 = {01}, qui est le polynôme x^3.[23][22] L'inspection de l'équation (13) ci-dessus montrera que son effet est de former le mot de sortie en faisant tourner des octets dans le mot d'entrée. Cela signifie que

[b0, b1, b2, b3] est transformé en [b1, b2, b3, b0].

2.4.Chiffrement et déchiffrement avec l'AES

Dans ce qui suit, nous détaillons l'AES-128, où les 128 bits de données sont répartis en 16 blocs de 8 bits (8 bits = 1 octet), eux-mêmes « dispatchés » dans un tableau 4×4. Même les 128 bits de la clé sont organisés sous forme matricielle [BAB 12].Pour d'évidentes raisons de taille, les valeurs binaires seront notées sous forme hexadécimale.

La première étape de chiffrement consiste à combiner la matrice State (le bloc de texteclair) avec la clé. Cette opération s'appelle *AddRound* Key. À chaque tour, quatre transformations sont appliquées *SubBytes, ShiftRows, MixColumn* et *AddRoundKey* sauf pour le dernier tour, l'opération *MixColumns* n'est pas effectuée. Chaque tour utilise sa propre sousclé qui est générée par l'opération Key Expansion à partir de la clé maitresse [25].

Le déchiffrement est l'opération inverse du chiffrement et les transformations se réalisent dans le sens inverse.

La figure 2.2 montre les différentes opérations effectuées dans chaque round pendant le processus de chiffrement et de déchiffrement de l'AES.

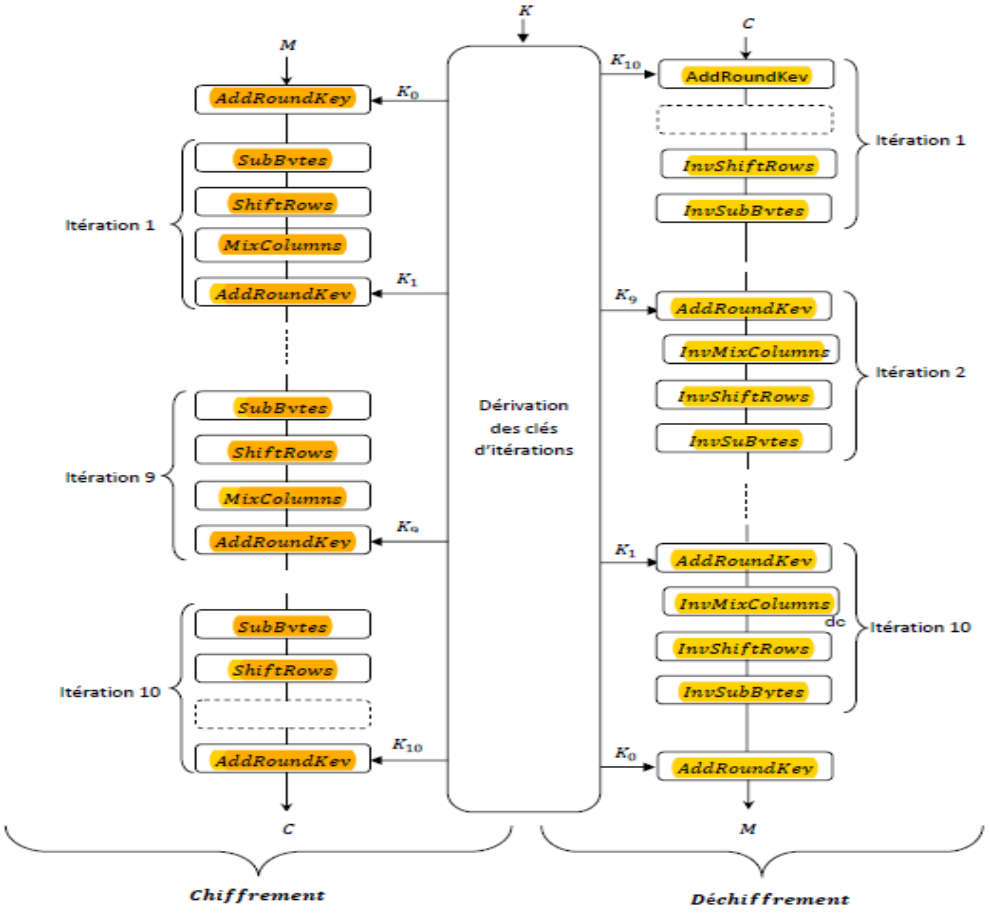


Figure 2.2 Etapes de l'algorithme AES

2.5 CHIFFRAGE

2.5.1. Processus de chiffrement

Le processus de cryptage de l'algorithmeAES(Advanced Encryption Standard)est présenté ci-dessous, dans la figure 2.3.

Les opérations de chiffrement s'appuient sur quatre fonctions prédéfinies : AddRoundKey, SubBytes, ShiftRows et MixColumns. Chacune de ces fonctions est exécutée sur le tableau d'états. Le cycle de chiffrement comprend une transformation initiale, des tours intermédiaires et un tour final.

La transformation initiale consiste à appliquer la fonction AddRoundKey au tableau d'états. Les tours intermédiaires exécutent, dans l'ordre, les fonctions SubBytes, ShiftRows, MixColumns et AddRoundKey sur le tableau d'états. Le tour final diffère des tours intermédiaires, par la suppression de la fonction MixColumns dans le cycle des transformations[21][24].

Le nombre de tour dans l'AES est dépendant de la taille de la clé. Ainsi, pour une clé de 128 bits, le nombre de tours est 10. De même, nous avons 12 tours pour une clé de 192 bits et 14 tours pour une clé de 256 bits.

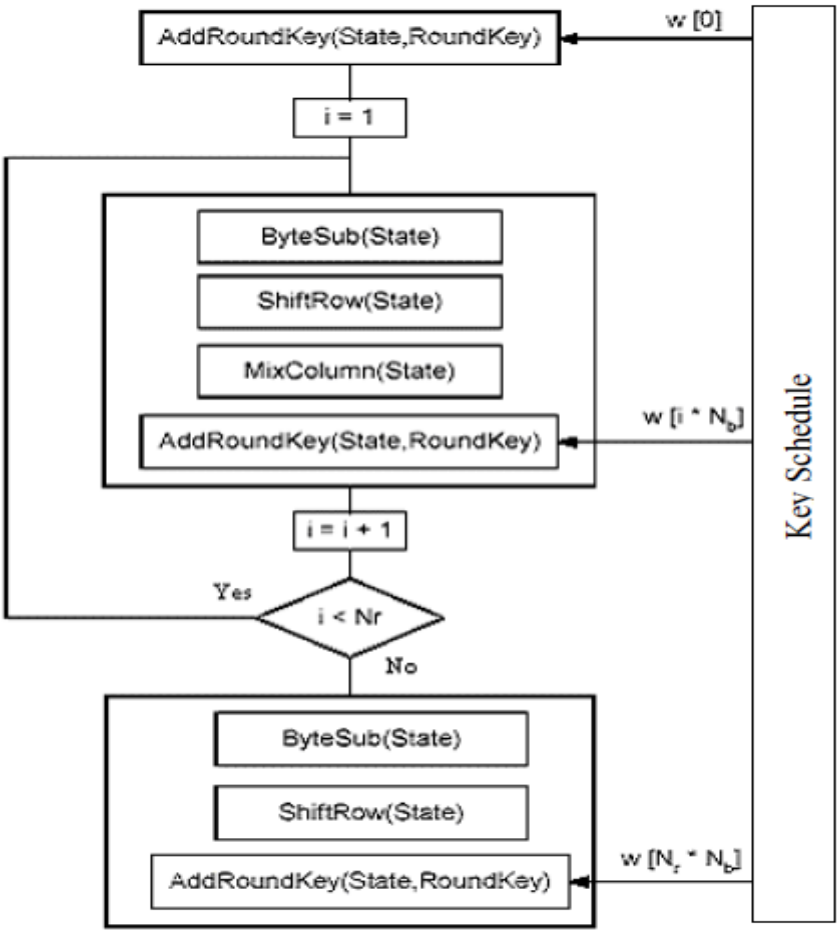


Figure 2.3. Le processus de chiffrement de l'AES[26]

Ce diagramme est générique pour les spécifications AES. Il consiste en un nombre de transformations différentes appliquées consécutivement sur les bits du bloc de données, dans un nombre fixe d'itérations, appelées rounds. Le nombre de tours dépend de la longueur de la clé utilisée pour le processus de cryptage[21][24].

2.5.2. LaTransformation subBytes

La transformation de substitution d'octets subBytes (état) est une substitution non

linéaire d'octets qui fonctionne indépendamment sur chaque octet de l'état en utilisant une table de substitution (S-box) présentée dans la figure2.6. Cette S-box qui est inversible, est construite en composant deux transformations

1. Prendre l'inverse multiplicatif dans le champ fini GF (28), décrit dans la section 1.3.2. L'élément {00} est mappé sur lui-même.
2. Appliquez la transformation affine suivante (sur GF (2))

$$b'_i = b_i \oplus b_{(i+4) \bmod 8} \oplus b_{(i+5) \bmod 8} \oplus b_{(i+6) \bmod 8} \oplus b_{(i+7) \bmod 8} \oplus c_i \quad (16)$$

pour $0 \leq i \leq 8$, où bi est le ième bit de l'octet, et ci est le ième bit d'un octet c avec la valeur {63} ou {01100011}. Ici et ailleurs, un nombre premier sur une variable (par exemple, b ') indique que la variable doit être mise à jour avec la valeur sur la droite. Sous forme matricielle, l'élément de transformation affine de l a boîte S peut être exprimé comme

$$\begin{bmatrix} b'_0 \\ b'_1 \\ b'_2 \\ b'_3 \\ b'_4 \\ b_5 \\ b'_6 \\ b'_7 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} \pm \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

Figure 2.4. Notation matricielle de s-box[22]

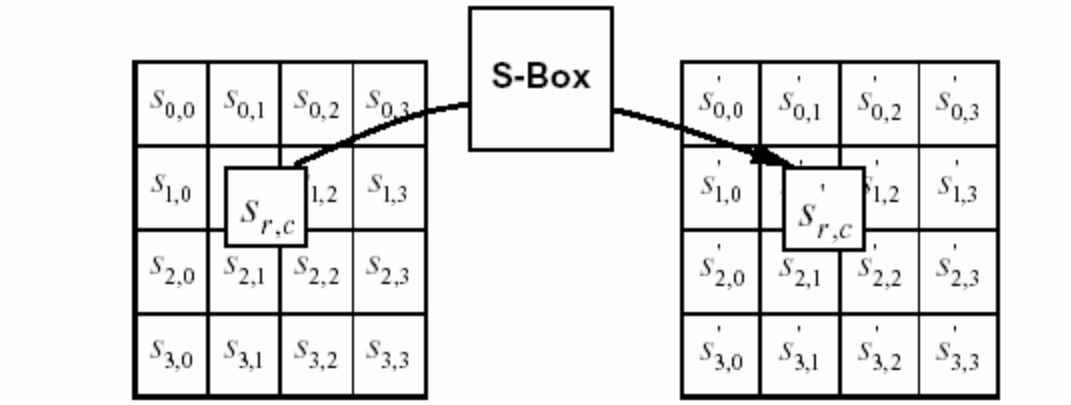


Figure 2.5. Application de S-box à chaque Byte de l'État [22]

Le S-box utilisé dans la transformation Sub Bytes est présenté sous forme hexadécimale dans la figure 2.6. Par exemple, if = S1,1 = {53}, alors la valeur de substitution serait déterminée par l'intersection de la ligne avec l'index '5' et la colonne avec l'index '3' sur la figure 2.6. Cela donnerait à $S'_{1,1}$ une valeur de {ed}.

		y														
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
x	0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
	1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
	2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
	3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
	4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
	5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
	6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
	7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
	8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
	9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
	a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
	b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
	c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
	d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
	e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
	f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

Figure 2.6. Valeurs S-box pour toutes les 256 combinaisons en format hexadécimal [22]

2.5.3.La transformation ShiftRows

Dans la transformation ShiftRows Shift Row (), les octets dans les trois dernières lignes de l'état sont décalés de manière cyclique sur différents nombres d'octets (offsets).

La première rangée, r = 0, n'est pas décalée. Plus précisément, la transformation ShiftRows () se déroule comme suit

$$s'_{r,c} = s_{r,(c+shift(r,Nb))} \bmod Nb \quad \text{for } 0< r < 4 \text{ and } 0 \leq c \leq Nb,$$

Lorsque le décalage de valeur de décalage (r, Nb) dépend du numéro de ligne, r, comme suit (Nb = 4)

$$Shift(1,4) = 1: Shift(2,4) = 2; Shift(3,4) = 3.$$

a pour effet de déplacer les octets vers des positions "inférieures" dans la rangée (c.-à-d., des valeurs inférieures de **c** dans une rangée donnée), tandis que les octets "les plus bas" s'enroulent dans le "haut" de la rangée ((c'est-à-dire des valeurs supérieures de c dans une ligne donnée.). La figure 2.6 illustre la transformation ShiftRows ().[20][24].

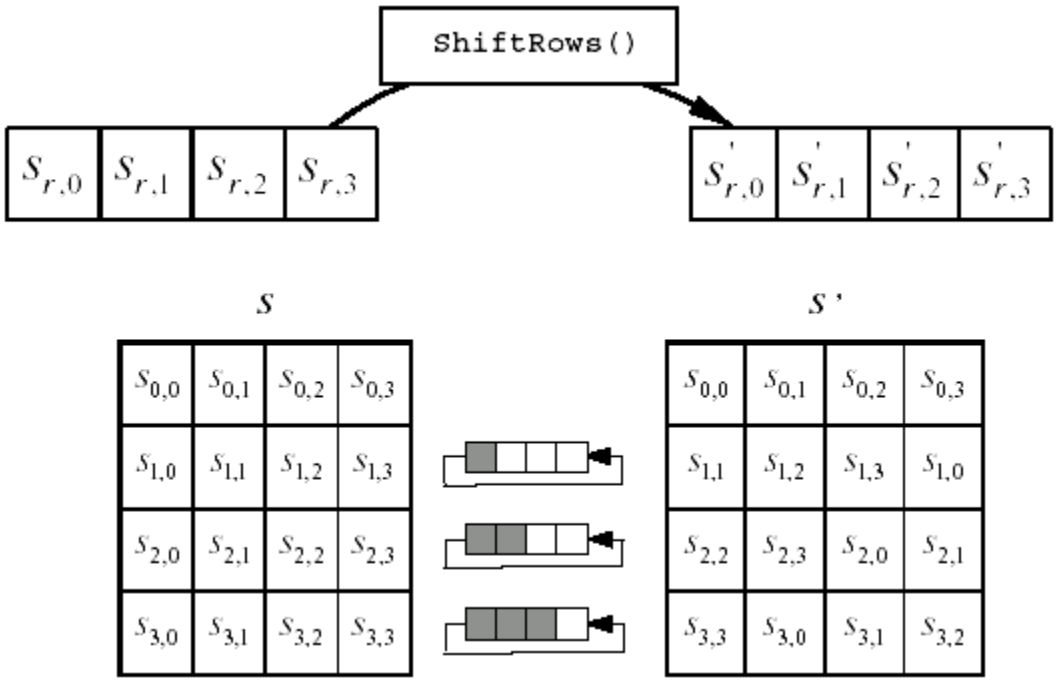


Figure 2.7. Décalage cyclique des trois dernières rangées de l'État [22]

2.5.4. La transformation MixColumns

Cette transformation est basée sur la multiplication du champ de Galois. Chaque octet d'une colonne est remplacé par une autre valeur qui est une fonction de tous les quatre octets dans la colonne donnée. La transformation MixColumns () fonctionne sur l'état colonne par colonne, en traitant chaque colonne comme un polynôme à quatre termes, comme décrit dans la section 1.3.4. Les colonnes sont considérées comme des polynômes sur GF (2^8) et multipliées modulo x4 + 1 avec un polynôme fixe a (x), donné par l'équation suivante[22].

$$a(x) = \{03\}x^3 + \{01\}x^2 + \{01\}x + \{02\}.$$

Comme décrit dans la section. 1.3.4, ceci peut être écrit comme une multiplication matricielle. Laisser

$$S'(x) = a(x) \otimes S(x)$$

$$\begin{bmatrix} S'_{0,c} \\ S'_{0,c} \\ S'_{0,c} \\ S'_{0,c} \end{bmatrix} \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} S'_{0,c} \\ S'_{0,c} \\ S'_{0,c} \\ S'_{0,c} \end{bmatrix} \qquad \text{Pour } 0 \leq c < Nb.$$

À la suite de cette multiplication, les quatre octets d'une colonne sont remplacés par les suivants

$$S'_{0,c} = (\{02\} \bullet S_{0,c}) \oplus (\{03\} \bullet S_{1,c}) \oplus S_{2,c} \oplus S_{3,c}$$

$$S'_{1,c} = S_{0,c} \oplus (\{02\} \bullet S_{1,c}) \oplus (\{03\} \bullet S_{2,c}) \oplus S_{3,c}$$

$$S'_{2,c} = S_{0,c} \oplus S_{1,c} \oplus (\{02\} \bullet S_{2,c}) \oplus (\{03\} \bullet S_{3,c})$$

$$S'_{3,c} = (\{03\} \bullet S_{0,c}) \oplus S_{1,c} \oplus S_{2,c} \oplus (\{02\} \bullet S_{3,c})$$

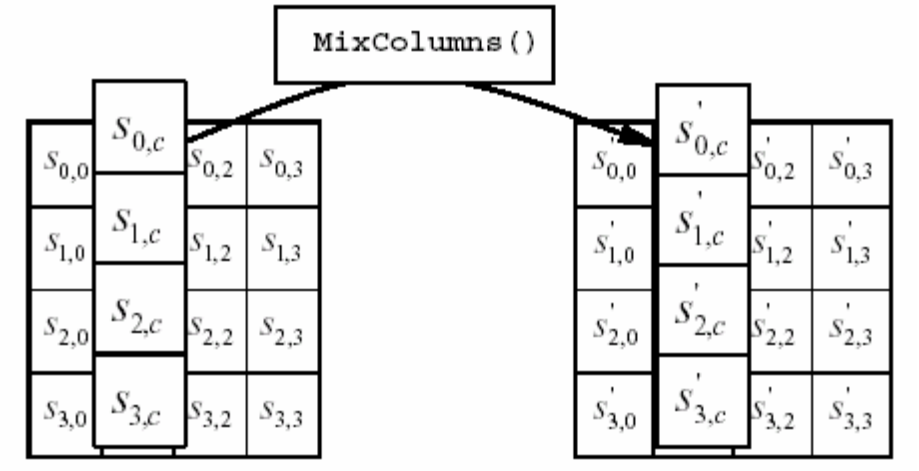


Figure 2.8. Mélange de colonnes de l'État [22]

2.5.5. Addition de la transformation de clé ronde

Dans Addition de la transformation de clé ronde AddRoundKey (), une clé ronde est ajouté à l'état par une simple opération XOR bit à bit. Chaque clé ronde est composée de N_b les mots issus de la génération d'horaires clés (décrits dans la section 2.6 ci-dessous). Ceux Nb les mots sont chacun ajoutés dans les colonnes de l'Etat, de telle sorte que

$$[S'_{0,c}, S'_{1,c}, S'_{2,c}, S'_{3,c},] = [S_{0,c}, S_{1,c}, S_{2,c}, S_{3,c}] \oplus [W_{round*Nb}] \qquad \text{pour } 0 \leq c \leq Nb$$

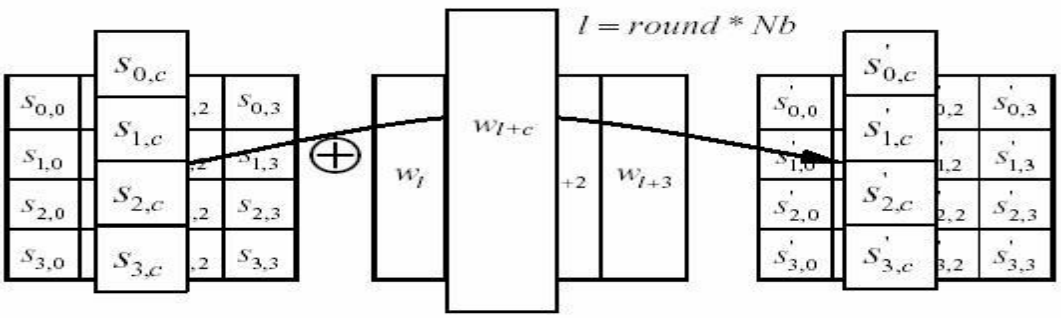


Figure 2.9. Fonctionnement OU exclusif des mots clés d'état et de chiffrement [22]

où [wi] sont les mots de génération de clé décrits dans le chapitre 3, et round est une valeur dans la plage dans le cryptage, l'addition initiale de clé ronde se produit quand round = 0, avant la première application de la fonction round. L'application de la transformation AddRoundKey () aux Nr tours du cryptage se produit lorsque 1 ≤ rond ≤ Nr. L'action de cette transformation est illustrée sur la figure 2.9, où l = round * Nb. L'adresse d'octet dans les mots de l'horaire clé a été décrite à la section 1.2.2.

2.5.6. Key schedule Expansion

Chaque clé ronde est un tableau de 4 mots (128 bits) généré comme un produit de la clé ronde précédente, une constante qui change chaque tour, et une série de recherches S-Box (figure2.5) pour chaque mot de 32 bits du clé. La première touche est la même que l'entrée d'origine de l'utilisateur. Chaque octet (w0 - w3) de la clé initiale est XOR'd avec une constante qui dépend du tour en cours, et le résultat de la recherche S-Box pour wi, pour former la prochaine clé ronde. Le nombre de tours requis pour trois longueurs de clé différentes est présenté dans tableau 2.3.

	Longueur de clé (Nk mots)	Taille de bloc (Nb mots)	Nombre de Rondes (Nr)
AES-128	4	4	10
AES-192	6	4	12
AES-256	8	4	14

Tableau 2.3. Bloc-clé- Combinaisons rondes [22]

Key Expansion génère un total de Nb (Nr + 1) mots: l'algorithme nécessite un ensemble initial de mots Nb, et chacun des Nr tours nécessite Nb mots de données clés.

Le programme de clés résultant est constitué d'un tableau linéaire de mots de 4 octets, noté [wi], avec i dans l'intervalle 0 ≤ i <Nb (Nr + 1).

2.6.DÉCRYPTAGE

2.6.1. Processus de décryptage

Le processus de déchiffrement de l'algorithmeAES(Advanced Encryption Standard) est présenté ci-dessous, dans la figure2.10.

Le déchiffrement est réalisé en effectuant les opérations inverses des quatre fonctions de chiffrement, dans l'ordre inverse.

Ainsi, chaque fonction utilisée dans les opérations de chiffrement dispose de safonction inverse, utilisée pour le déchiffrement : InvShiftRows, InvSubBytes et InvMixColumns. La fonction AddRoundKey reste inchangée. À l'instar du chiffrement,

le processus de déchiffrement comprend une transformation initiale, des tours intermédiaires et un tour final.

La transformation initiale consiste à appliquer la fonction AddRoundKey au tableaud'états.

Les tours intermédiaires exécutent, dans l'ordre, les fonctions InvShiftRows, InvSubBytes, AddRoundKey et InvMixColumns sur le tableau d'états.

Le tour final diffère des tours intermédiaires, par la suppression de la fonction InvMixColumns dans le cycle des transformations.

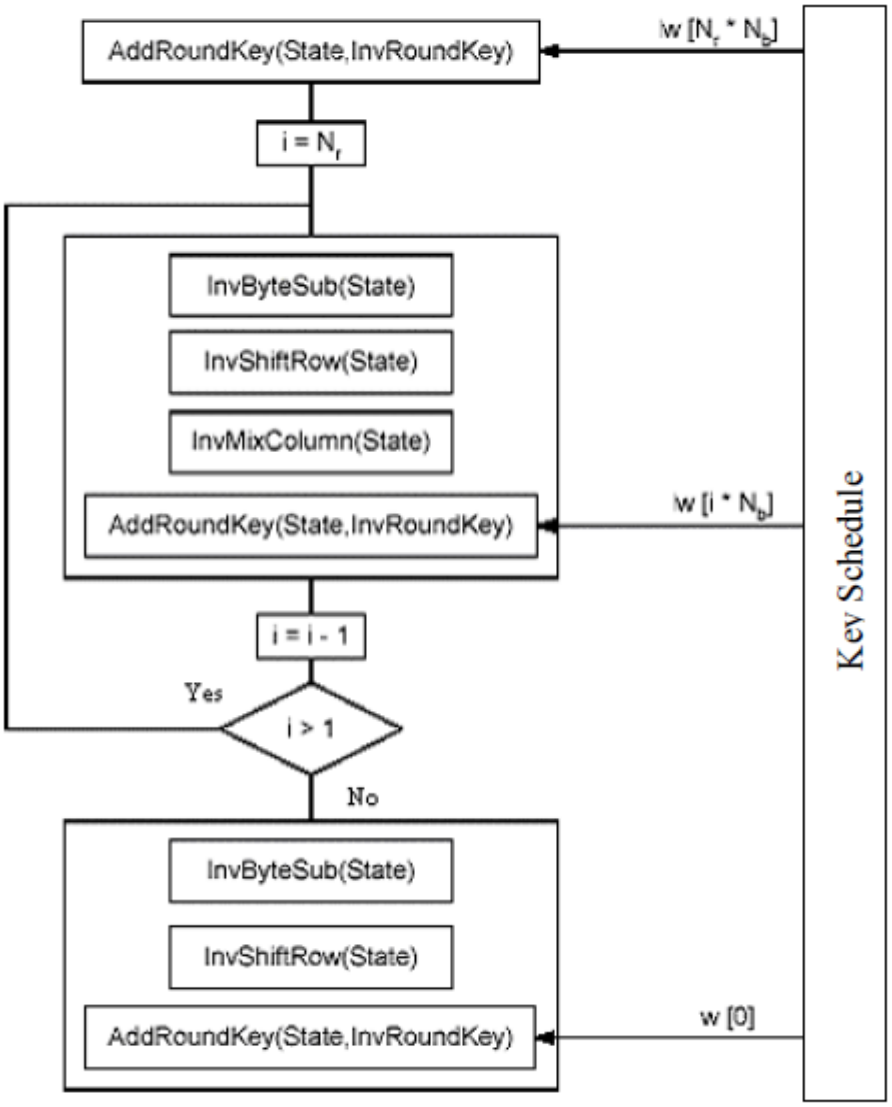


Figure 2.10. Processus de Décryptage[26]

Ce processus est l'inverse direct du processus de cryptage . Toutes les transformations appliquées dans le processus de cryptage sont inversement appliquées à ce processus. Par conséquent, les dernières valeurs arrondies des données et de la clé sont des entrées de premier tour pour le processus de déchiffrement et suivent dans l'ordre décroissant.

2.6.2. Transformation InvSubBytes

La Transformation InvSubBytes () est l'inverse de la transformation de substitution d'octets, dans laquelle la S-Box inverse (figure2.12) est appliquée à chaque octet de l'état. Ceci est obtenu en appliquant l'inverse de la transformation affine à l'équation (16) suivie de l'inverse multiplicatif dans GF (2^8).

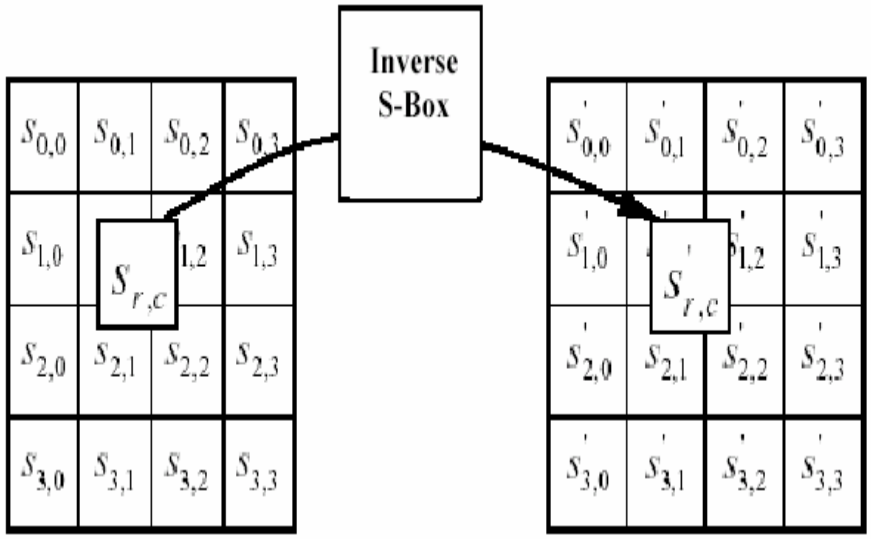


Figure 2.11. Application de la boîte S inverse à chaque octet de l'État [22]

x \ y	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	52	09	6a	d5	30	36	a5	38	bf	40	a3	9e	81	f3	d7	fb
1	7c	e3	39	82	9b	2f	ff	87	34	8e	43	44	c4	de	e9	cb
2	54	7b	94	32	a6	c2	23	3d	ee	4c	95	0b	42	fa	c3	4e
3	08	2e	a1	66	28	d9	24	b2	76	5b	a2	49	6d	8b	d1	25
4	72	f8	f6	64	86	68	98	16	d4	a4	5c	cc	5d	65	b6	92
5	6c	70	48	50	fd	ed	b9	da	5e	15	46	57	a7	8d	9d	84
6	90	d8	ab	00	8c	bc	d3	0a	f7	e4	58	05	b8	b3	45	06
7	d0	2c	1e	8f	ca	3f	0f	02	c1	af	bd	03	01	13	8a	6b
8	3a	91	11	41	4f	67	dc	ea	97	f2	cf	ce	f0	b4	e6	73
9	96	ac	74	22	e7	ad	35	85	e2	f9	37	e8	1c	75	df	6e
a	47	f1	1a	71	1d	29	c5	89	6f	b7	62	0e	aa	18	be	1b
b	fc	56	3e	4b	c6	d2	79	20	9a	db	c0	fe	78	cd	5a	f4
c	1f	dd	a8	33	88	07	c7	31	b1	12	10	59	27	80	ec	5f
d	60	51	7f	a9	19	b5	4a	0d	2d	e5	7a	9f	93	c9	9c	ef
e	a0	e0	3b	4d	ae	2a	f5	b0	c8	eb	bb	3c	83	53	99	61
f	17	2b	04	7e	ba	77	d6	26	e1	69	14	63	55	21	0c	7d

Figure 2.12. Valeurs S-box inverses pour toutes les 256 combinaisons en format hexadécimal [26]

2.6.3. Transformation InvShiftRows

Les octets dans les trois dernières lignes de l'état sont décalés cycliquement sur différents nombres d'octets. La première rangée, r = 0, n'est pas décalée. Les trois rangées inférieures sont décalées de manière cyclique par octets de décalage Nb-shift (r, Nb), où le décalage de valeur de décalage (r, Nb) dépend du numéro de ligne et est

expliqué dans la section.2.3. Plus précisément, la transformation InvShiftRows () se déroule comme suit

$$S'_{r,(c+shift(r,Nb)) \bmod Nb} = S_{r,c} \quad \text{pour } 0 \le r<4 \text{ and } 0 \le c<Nb$$

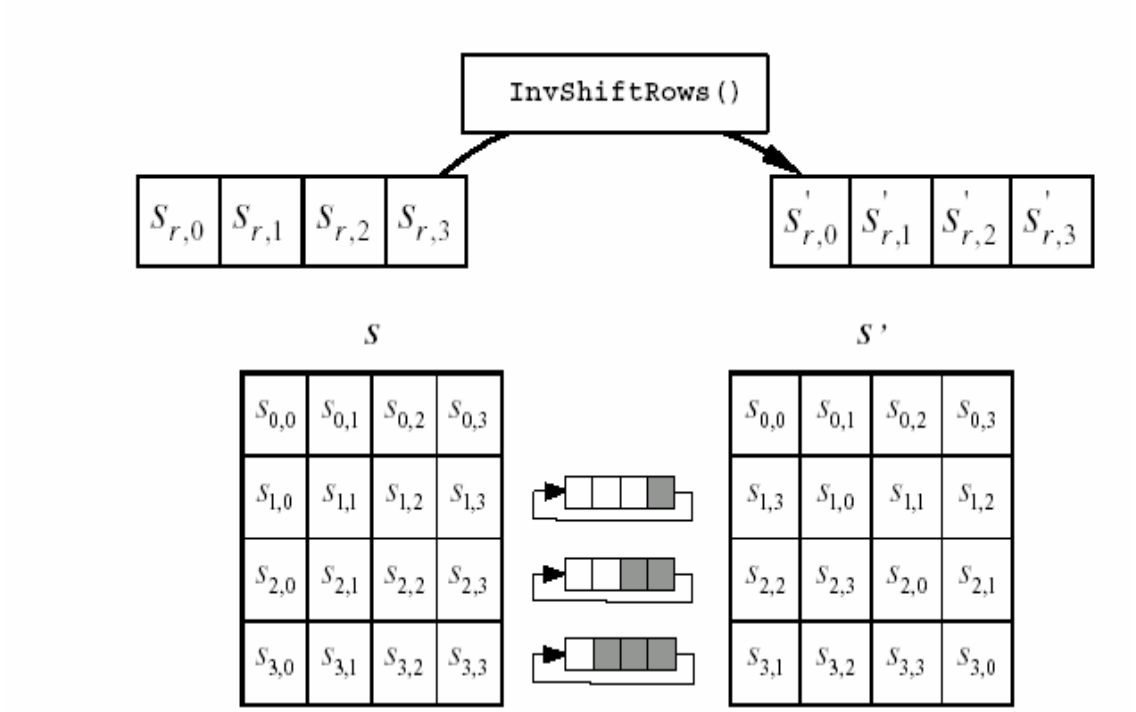


Figure 2.13. Décalage cyclique inverse des trois dernières rangées de l'État [22]

2.6.4. Transformation InvMixColumns

InvMixColumns opère sur l'état colonne par colonne, en traitant chaque colonne comme un polynôme à quatre termes comme décrit dans la section.1.3.4. Les colonnes sont considérées comme des polynômes sur GF (2^8) et multipliées modulo $x^4 + 1$ avec un polynôme fixe $a^{-1}(x)$, donné par

$$a^{-1}(x) = \{0b\}x^3 + \{0d\}x^2 + \{09\}x + \{0e\}.$$

Comme décrit dans la section 1.3.4, ceci peut être écrit comme une multiplication matricielle. Laisser $S'(x) = a^{-1}(x) \otimes S(x)$

$$\begin{bmatrix} S'_{0,c} \\ S'_{0,c} \\ S'_{0;c} \\ S'_{0,c} \end{bmatrix} = \begin{bmatrix} 0e & 0b & 0d & 09 \\ 09 & 0e & 0b & 0d \\ 0d & 09 & 0e & 0b \\ 0b & 0d & 09 & 0e \end{bmatrix} \begin{bmatrix} S_{0,c} \\ S_{0,c} \\ S_{0,c} \\ S_{0,c} \end{bmatrix} \quad \text{Pour } 0 \le c \le Nb$$

À la suite de cette multiplication, les quatre octets d'une colonne sont remplacés par les équations suivantes.

$$S'_{0,c} = (\{0e\} \bullet S_{0,c}) \oplus (\{0b\} \bullet S_{1,c}) \oplus (\{0d\} \bullet S_{2,c}) \oplus (\{09\} \bullet S_{3,c})$$

$$S'_{1,c} = (\{09\} \bullet S_{0,c}) \oplus (\{0e\} \bullet S_{1,c}) \oplus (\{0b\} \bullet S_{2,c}) \oplus (\{0d\} \bullet S_{3,c})$$

$$S'_{2,c} = (\{0d\} \bullet S_{0,c}) \oplus (\{09\} \bullet S_{1,c}) \oplus (\{0e\} \bullet S_{2,c}) \oplus (\{0b\} \bullet S_{3,c})$$

$$S'_{3,c} = (\{0b\} \bullet S_{0,c}) \oplus (\{0d\} \bullet S_{1,c}) \oplus (\{09\} \bullet S_{2,c}) \oplus (\{0e\} \bullet S_{3,c})$$

Par conséquent, l'État peut être représenté

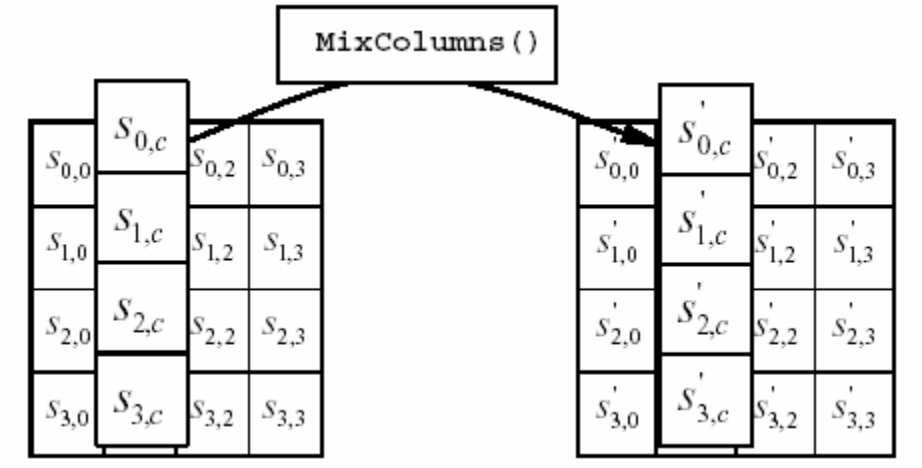


Figure 2.14. Opération de Inverse Mix Column sur l'état [22]

2.7. Conclusion :

Dans ce chapitre nous avons détaillé le fonctionnement de l'algorithme AES (Advanced Encryption Standard) dans les deux modes chiffrement et déchiffrement. Nous avons identifié les étapes des deux algorithmes de chiffrement et de déchiffrement et constaté que pour une clé de 128 bits il existe 10 tours (rounds) et chaque round consiste en une fonction non linéaire de substitution SubBytes() et des fonctions linéaires de décalage et permutation ShiftRows() et MixColumns() et enfin se termine par la fonction AddRoundKey() qui est un simple ou exclusif avec les colonne de la matrice state. La fonction KeyExpansion() permet de diversifier la clef de chiffrement. Le déchiffrement s'exécute en inversant les fonctions cites précédemment. Nous allons donc aborder dans le chapitre prochain les circuits programmables FPGA sur lesquels on va programmer les fonctions de chiffrement AES en un langage approprié connu sous le nom VHDL.

Chapitre III :

LES CIRCUITS FPGA

3 .1.INTRODUCTION

Les FPGA, sigle anglais qui signifie « Field Programmable Gates Arrays » traduit en français par réseau de portes programmables, sont des circuits intégrés reprogrammables. Ils offrent la possibilité de réaliser des fonctions numériques plus ou moins complexes, tout comme leurs homologues figés : les ASIC [27].

Les FPGA sont des circuits numériques matériels configurables dédiés à l'électronique numériques [28].

A l'état initial ils ne peuvent rien faire mais disposent d'une importante quantité (dépendant de la technologie utilisée) de ressources matérielles opérationnelles dont on peut configurer la fonction. Ces ressources sont, principalement, des blocs élémentaires logiques (pour réaliser des fonctions booléennes), des mémoires RAM, des opérateurs arithmétiques (qui travaillent en virgule fixe), des ressources de routage interne et des entrées/sorties. Ces ressources configurables sont reliées par un réseau dense de lignes de routage et de lignes de transport des horloges. Ces lignes de routage sont aussi configurables. En plus de ces ressources, un FPGA est composé d'une mémoire interne de configuration. Chaque point de cette mémoire correspond à la configuration d'un élément d'une des ressources opérationnelles. Cette mémoire est, dans la plupart des cas, réalisée avec une des trois technologies suivantes : ANTIFUSIBLE (la plus ancienne, configurable une seule fois), FLASH (non-volatile) ou SRAM (volatile, la plus utilisée, représente plus de 80 % du marché). Pour réaliser une application avec un FPGA il faut décrire le circuit électronique à réaliser avec un langage de description matérielle comme le VHDL (Very High Speed Integrated Circuit Hardware Description Langage). Puis il faut synthétiser cette description en circuit électronique. Cette étape et les suivantes peuvent se faire avec des logiciels gratuits fournies par le fabricant de circuit. Enfin après une étape de placement et routage qui prend en compte l'architecture du FPGA, un fichier de configuration appelé bitstream est généré. Celui-ci permet de spécifier au FPGA lors de la configuration la position des points de la mémoire de configuration [28].

La technologie FPGA (Field-Programmable Gate Array) continue de gagner du terrain : selon les prévisions, le marché mondial du FPGA devrait passer d'1,9 milliard de dollars en 2005 à 2,75 milliards d'ici 2010 [29]. Depuis leur invention par Xilinx en 1984, les FPGA sont partis d'un simple rôle d'« interfaçage d'appoint » pour arriver à véritablement remplacer les ASIC (circuits intégrés à application spécifique) et les

processeurs personnalisés dans des applications de contrôle et de traitement de signaux. Afin d'expliquer ce succès, cet chapitre propose une introduction à la technologie FPGA et met en évidence quelques-uns des nombreux avantages que les FPGA sont aujourd'hui les seuls à offrir [30].

3.2.Description de la composent FPGA

Un FPGA est un circuit en silicium reprogrammable. À l'aide de blocs logiques préconstruits et de ressources de routage programmables, vous pouvez configurer ce circuit afin de mettre en œuvre des fonctionnalités matérielles personnalisées, sans avoir jamais besoin d'utiliser une maquette ou un fer à souder. Il vous suffit de développer des tâches de traitement numérique par logiciel et de les compiler sous forme de fichier de configuration ou de flux de bits contenant des informations sur la manière dont les composants doivent être reliés. En outre, les FPGA sont totalement reconfigurables et peuvent adopter instantanément une nouvelle « personnalité » si vous recompilez une nouvelle configuration de circuits. Jusqu'à présent, seuls des ingénieurs particulièrement expérimentés en matière de conception de matériel numérique pouvaient utiliser la technologie FPGA.

Si les FPGA rencontrent un tel succès dans tous les secteurs, c'est parce qu'ils réunissent le meilleur des ASIC et des systèmes basés processeur. Ainsi, ils offrent un cadencement par matériel qui leur assure vitesse et fiabilité, mais sont plus rentables que les ASIC personnalisés. Les circuits reprogrammables jouissent également de la même souplesse d'exécution logicielle qu'un système basé processeur, mais ils ne sont pas limités par le nombre de cœurs de traitement disponibles. Contrairement aux processeurs, les FPGA sont vraiment parallèles par nature, de sorte que plusieurs opérations de traitement différentes ne se trouvent pas en concurrence pour l'utilisation des ressources. Chaque tâche de traitement indépendante est affectée à une section spécifique du circuit, et peut donc s'exécuter en toute autonomie sans dépendre aucunement des autres blocs logiques. En conséquence, vous pouvez accroître le volume de traitement effectué sans que les performances d'une partie de l'application n'en soient affectées pour autant [30,31.32].

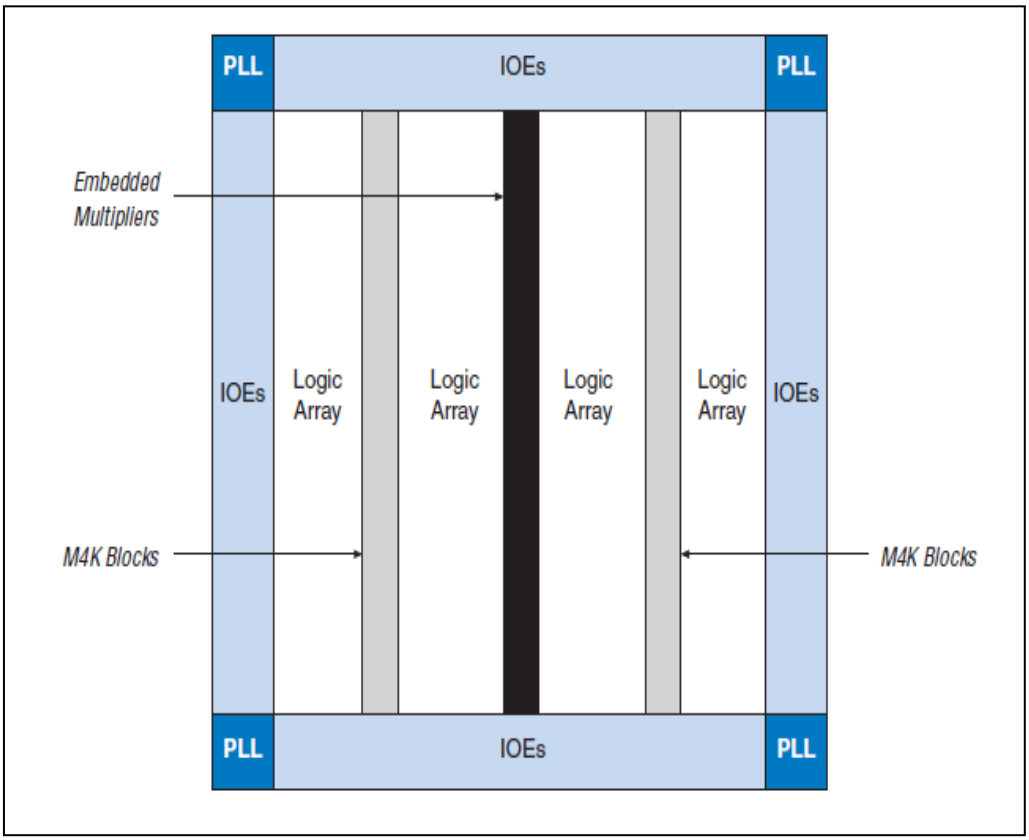


Figure 3.1 : Architecture interne du FPGA fabriqué ALTERA (Cyclone II EP2C20) [33].

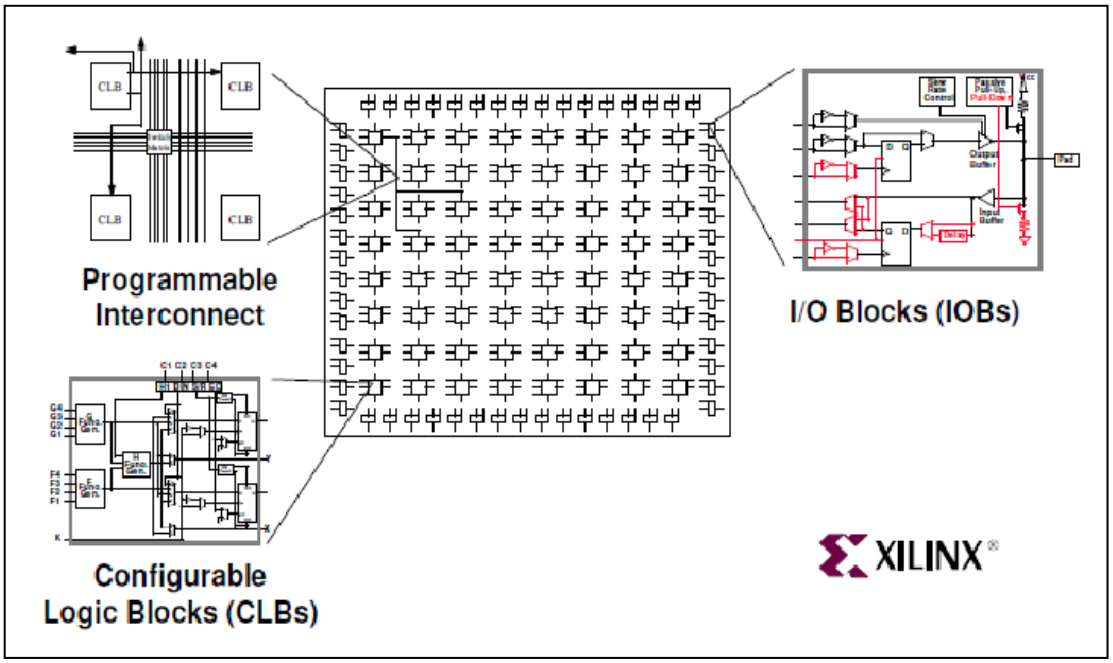


Figure 3.2 : Architecture interne du FPGA fabriqué Xilinx : XC4000 [31].

3.3 Les cinq principaux atouts de la technologie FPGA

1. Performances
2. Temps de mise sur le marché
3. Coût
4. Fiabilité
5. Maintenance à long terme

3.3.1. Performances

Comme ils tirent parti du parallélisme matériel, les FPGA offrent une puissance de calcul supérieure à celle des processeurs de signaux numériques (DSP), car ils s'affranchissent du modèle d'exécution séquentielle et exécutent plus d'opérations par cycle d'horloge. BDTI, une importante société d'analyse et de « *benchmarking* », a publié des études montrant que les FPGA peuvent offrir une puissance de traitement par dollar plusieurs fois supérieure à celle d'une solution DSP dans certaines applications. Contrôler les entrées et sorties (E/S) au niveau matériel permet d'obtenir des temps de réponse plus courts ainsi que des fonctionnalités spécifiques, qui répondent mieux aux besoins de l'application.

3.3.2. Temps de mise sur le marché

Face à des préoccupations croissantes concernant les temps de mise sur le marché, la technologie FPGA représente une solution souple offrant des capacités de prototypage rapide. Ainsi, vous pouvez tester une idée ou un concept, puis le vérifier sur du matériel sans avoir à passer par le long processus de fabrication d'un ASIC personnalisé [28]. Par la suite, vous pourrez apporter les éventuelles modifications nécessaires à votre FPGA, en quelques heures au lieu de quelques semaines. Le matériel « sur étagère » actuellement commercialisé propose également différents types d'E/S déjà connectées à un circuit FPGA programmable par l'utilisateur. La multiplication des outils logiciels de haut niveau disponibles sur le marché permet de réduire le temps d'apprentissage avec les couches d'abstraction. Ces outils comprennent souvent des cœurs de propriété intellectuelle (fonctions précompilées) utiles pour le contrôle avancé et le traitement de signaux.

3.3.3. Coût

Les coûts d'ingénierie non récurrents (NRE) des ASIC personnalisés sont bien supérieurs à ceux des solutions matérielles basées sur du FPGA. L'important investissement de départ que requièrent les ASIC se justifie largement pour les OEM, par exemple, qui peuvent livrer des circuits par milliers chaque année. Cependant, la plupart des utilisateurs finaux ont besoin de matériels personnalisés pour quelques dizaines ou quelques centaines de systèmes en développement. Par nature, les circuits programmables n'impliquent ni coût de fabrication, ni longs délais d'assemblage. Les besoins de la plupart des systèmes évoluent avec le temps ; or la modification progressive d'un FPGA représente un coût négligeable comparé à la dépense considérable qu'exige la reconception d'un ASIC.

3.3.4. Fiabilité

Tandis que les outils logiciels fournissent l'environnement de programmation, les circuits FPGA sont une véritable implémentation matérielle de l'exécution logicielle. Les systèmes basés processeur comprennent souvent plusieurs couches d'abstraction, pour aider à la planification des tâches et à la répartition des ressources entre les différents processus. La couche de driver contrôle les ressources matérielles et le système d'exploitation gère la mémoire et la bande passante du processeur. Sur chaque cœur de processeur, une seule instruction peut s'exécuter à la fois ; c'est pourquoi les systèmes basés processeur risquent toujours de voir des tâches prioritaires entrer en conflit. Les FPGA, qui n'utilisent pas de système d'exploitation, minimisent les problèmes de fiabilité car ils assurent une exécution véritablement parallèle et un matériel déterministe dédié à chaque tâche.

3.3.5. Maintenance à long terme

Comme nous l'avons vu, les circuits FPGA sont évolutifs et vous épargnent donc la dépense de temps et d'argent qu'implique la reconception des ASIC. Les spécifications des protocoles de communication numériques, par exemple, évoluent avec le temps. Or les interfaces basées sur ASIC peuvent poser des problèmes de maintenance et de compatibilité. Comme ils sont reconfigurables, les circuits FPGA sont capables de s'adapter aux modifications éventuellement nécessaires. À mesure qu'un produit ou qu'un système évolue, vous pouvez y intégrer des améliorations fonctionnelles sans

perdre de temps à reconcevoir le matériel ou à modifier l'implantation du circuit [30].

3.4.Fabricants

Ayant présenté les caractéristiques globales des FPGAS, nous allons rapidement passer en revue les produits proposés par les différents fabricants, à savoir, dans l'ordre alphabétique, Actel, Altera, Atmel, Lattice Semiconductor, QuickLogic et Xilinx .

- **Xilinx**

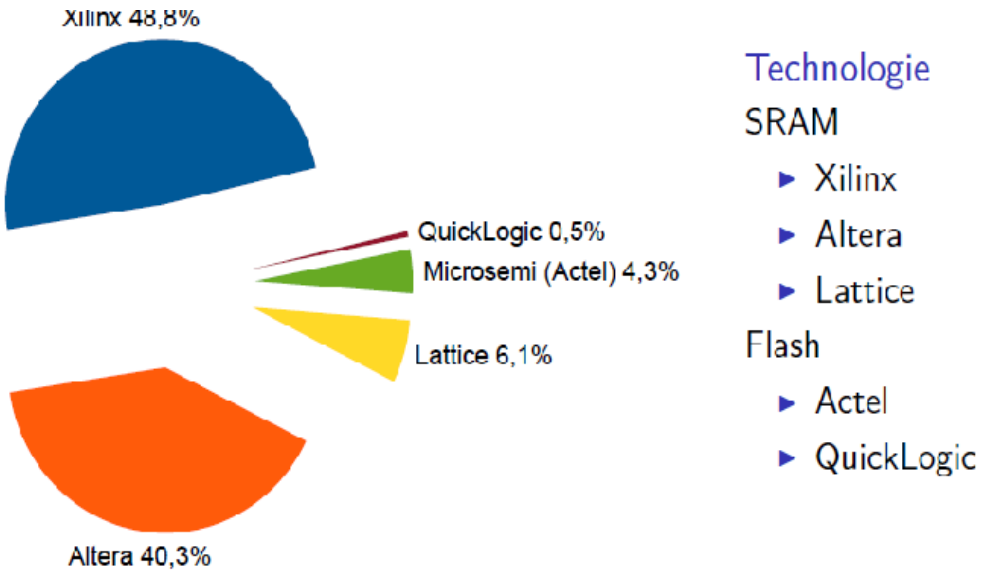
A l'heure actuelle, Xilinx, le premier fabricant de FPGAs, propose principalement deux familles, Virtex et Spartan, toutes deux de type SRAM, basées sur une architecture LUT. La différence entre les deux familles est minime, et tient principalement du nombre d'éléments proposés ainsi que du type de process utilisé, les Spartan étant positionnées bas coût en comparaison des Virtex. L'élément de base, le CLB (Configurable Logic Block), est composé de deux Slices, eux-mêmes comprenant deux 4-LUT et deux bascules. Les versions Virtex-II, Virtex-4 et Spartan-3 diffèrent toutefois, le CLB y contenant quatre Slices. Les deux familles contiennent des blocs de RAM, pouvant être utilisés en single ou dual port.

Alors que les séries Spartan-3, Virtex-II et Virtex-4 sont presque identiques en terme d'architecture, la série Virtex-II Pro introduit un ou deux processeurs de type PowerPC. Contrairement à l'échec de la solution d'Altera faisant intervenir un ARM, Xilinx a su imposer son produit, et la série Virtex-II Pro se trouve très bien positionnée dans la gamme de ses produits.

Le plus imposant de la série Spartan-3 propose 5M de portes équivalentes, dont 104 multiplicateurs de 18x18 bits et 1'872 K bits de RAM. En comparaison, les plus gros Virtex sont le XC4VLX200, de la famille Virtex-4LX et le XC4VFX140, un Virtex-FX. Le premier contient 178'176 éléments logiques, un élément logique étant une LUT à 4 entrées et une bascule. A ceci s'ajoutent 6'048 Kbits de RAM configurables, ainsi que 96 multiplicateurs 18x18 bits, pour un total de 15M de portes équivalentes (chiffre calculé). Le deuxième, le XC4VFX140, est composé de 126'336 éléments logiques, de 9'936 KBits de RAM, 192 multiplicateurs, et 2 processeurs de type PowerPC.

Nous pouvons finalement noter que Xilinx, à l'instar d'Altera, propose une solution appelée EasyPath, permettant la réalisation en ASIC d'un design implémenté sur un

Virtex-II, Virtex- II Pro ou un Spartan-3 [32].



.

Figure 3.3 : Parts de marché en termes de CA en 2010 [34].

Le tableau II-1 de Comparaison des caractéristiques des différentes FPGAs résume les principales caractéristiques des FPGAs disponibles sur le marché. Nous pouvons observer une complexification constante de ces circuits qui se voient ajouter de plus en plus de nouvelles caractéristiques. Il devient en effet quasiment impossible d'en trouver un ne possédant pas de blocs de mémoire RAM, et la présence de multiplicateurs devient presque systématique.

Sur le plan des structures de routage, la tendance est à l'optimisation. Les technologies utilisées devenant de plus en plus petites, le délai des portes se réduit d'autant, alors que celui des fils ne diminue pas. Dès lors une nouvelle attention est donnée à cette partie du circuit, de manière à assurer un fonctionnement correct à haute fréquence.

Concernant la complexité du routage, nous pouvons noter que les réseaux de routage des gros FPGAs ne se bornent pas à une simple grille de switchboxs, mais qu'ils font intervenir plusieurs niveaux. La série Startix d'Altera, par exemple, est composés de Logic Array Blocs (LABs) contenant 10 Logic Elements (LEs), chaque LE étant une LUT, une bascule et quelques portes logiques. Le LAB y a une connectique interne, et une connectique externe

pour des chemins de longue distance entre les LABs. Altera voit ce réseau comme une structure à plusieurs dimensions, l'ensemble du réseau étant vu comme une grille de sous- réseaux interconnectés, eux-mêmes étant une grille de sous-réseaux, etc [32].

Vendeur	Famille	Technologie	Portes Equivalentes	Nb Flip-flops	Mémoire Bloc (Kbits)	Multiplicateur 18×18	Processeur Processeur
Actel	Axcelerator	anti-fusible	2M	21'504	294	-	-
Actel	ProASIC Plus	FLASH	1M	56'320	198	-	-
Altera	Cyclone	SRAM	1M	20'060	295	-	-
Altera	Stratix	SRAM	4M (calcul)	79'040	7'427	176	-
Altera	Stratix II	SRAM	9M (calcul)	143'520	9'383	384	-
Altera	Excalibur	SRAM	1.7M	38'400	256	-	1 ARM922T
Atmel	AT40	SRAM	50K (Usable gates)	3'048	18.4	-	-
Atmel	AT6000	SRAM	30K (Usable gates)	6'400	-	-	-
Atmel	FPSLIC	SRAM	50K (déduit)	2'962	18.4	-	8-bit AVR
Lattice	ORCA4	SRAM	899K	18'216	148	-	-
Lattice	ispXPGA	E^2CMOS	1.25M	30'700	414	-	-
Lattice	ECP	SRAM	1M (calcul)	46'080	645	40	-
QuickLogic	Eclipse II	anti-fusible	320K	4'002	55	-	-
QuickLogic	pASIC	anti-fusible	75K	2'692	-	-	-
QuickLogic	QuickRAM	anti-fusible	176K (90K Usable PLD gates)	2'692	25	-	-
QuickLogic	QuickMIPS	anti-fusible	457K (115K Usable PLD gates)	4'032	83	18	MIPS32 4Kc
Xilinx	Virtex-II	SRAM	8M	93'184	3'024	168	-
Xilinx	Virtex-II Pro	SRAM	8M	88'192	7'992	444	2 PowerPC
Xilinx	Virtex-4LX	SRAM	15M (calcul)	178'176	6'048	96	-
Xilinx	Virtex-4FX	SRAM	11M (calcul)	126'336	9'936	192	2 PowerPC
Xilinx	Spartan-3	SRAM	5M	66'560	1'971	104	-

Tableau 3.1 : Comparaison des caractéristiques des différentes FPGAs [32].

3.5.Structure interne de FPGA

3.5.1.Architectures des FPGA

L'architecture d'un FPGA est principalement décrite par la topologie des ressources de routages et des éléments logiques configurables de base. Il existe deux architectures classiques, l'architecture îlot de calcul (initialement utilisée dans les composants Xilinx) et l'architecture hiérarchique (initialement utilisée dans les composants Altera).

Cependant, une tendance apparait avec les dernières générations de circuits, les architectures sont principalement de style îlots de calculs avec une légère hiérarchique (un ou deux niveaux de cluster hiérarchique). Les sections suivantes donnent quelques détails sur ces architectures. Dans le passé, d'autres architectures ont étaient utilisées. On peut citer l'architecture de routage logarithmique utilisée par Xilinx pour son circuit XC6000. Malheureusement, les outils de placement routage n'étaient pas adaptés à cette architecture pour permettre au concepteur d'en tirer pleinement partie.

3.5.2Architecture îlot de calcul

L'architecture la plus communément utilisée pour réaliser ces circuits est de type *îlot de calcul*. Dans ce cas les ressources configurables sont disposées sous formes de matrice, comme on peut le voir sur la figure 3.4. Des lignes de routage sont disposées horizontalement et verticalement autour des ressources configurables. Des blocs de connexion relient les ressources configurables aux lignes de connexion. Des matrices de connexion relient les lignes de routage horizontales et verticales.

L'utilisation de matrices de connexions configurables est indispensable pour assurer la connectivité des modules, mais les matrices de connexions configurables dégradent les caractéristiques des signaux, diminuent les performances (fréquence de fonctionnement et consommation de puissance) et nécessitent des outils de placement-routage efficaces. Sur la figure 3.4, on peut voir en gras des liaisons point à point entre deux éléments configurables. Ces liaisons utilisent : les ports d'entrées/sorties des éléments configurables, les connexions configurables qui permettent la connexion des éléments configurables au réseau de routage, les lignes de routages et les matrices de connexions configurables. Autant d'éléments parcourus qui dégradent les performances du circuit mais qui permettent une flexibilité importante.

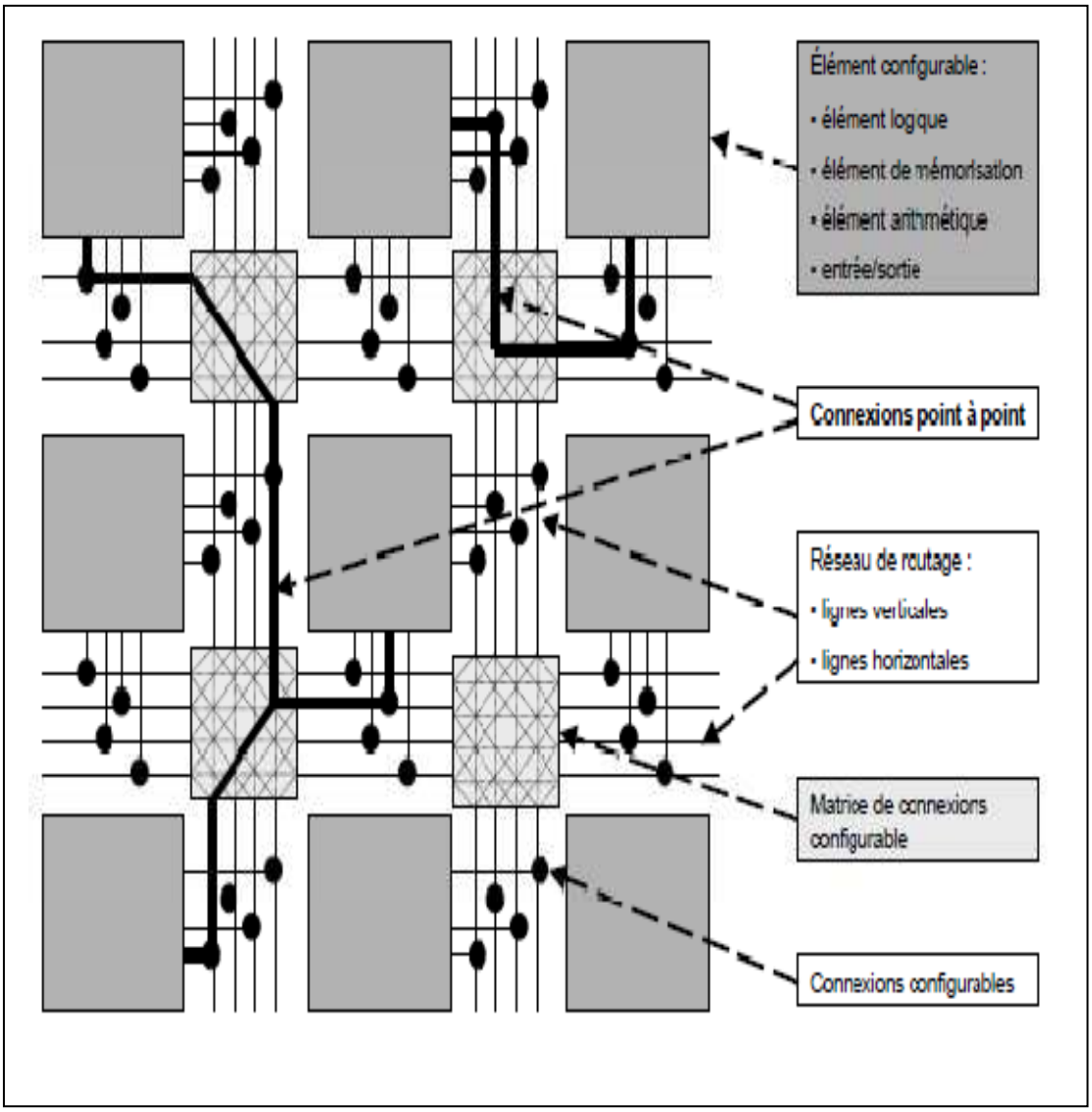


Figure 3.4 : Architecture îlot de calcul, typique des FPGA actuels [35].

3.5.3.Architecture hiérarchique

L'architecture hiérarchique couramment utilisée pour les circuits FPGA est constituée de quatre ou trois niveaux. A chacun de ces niveaux des ressources de routage sont disponibles pour communiquer entre les éléments propres du circuit. La figure3.5 schématise une architecture hiérarchique à quatre niveaux en utilisant un exemple

d'architecture couramment rencontrée dans les FPGA ACTEL. Au niveau le plus haut de la hiérarchique, le circuit est constitué de tuiles agencées matricielle ment. Les tuiles sont constituées de clusters logiques et de bancs de mémoires. Enfin, les clusters logiques regroupent les éléments logiques (et/ou arithmétiques) configurables. Ce style d'architecture peut être très efficace énergétiquement car elle permet de localiser les communications intenses et limite l'utilisation de longues lignes de routage.

Cependant, il est nécessaire pour les algorithmes de placement et routage de prendre en compte les caractéristiques Spécifiques de ces architectures.

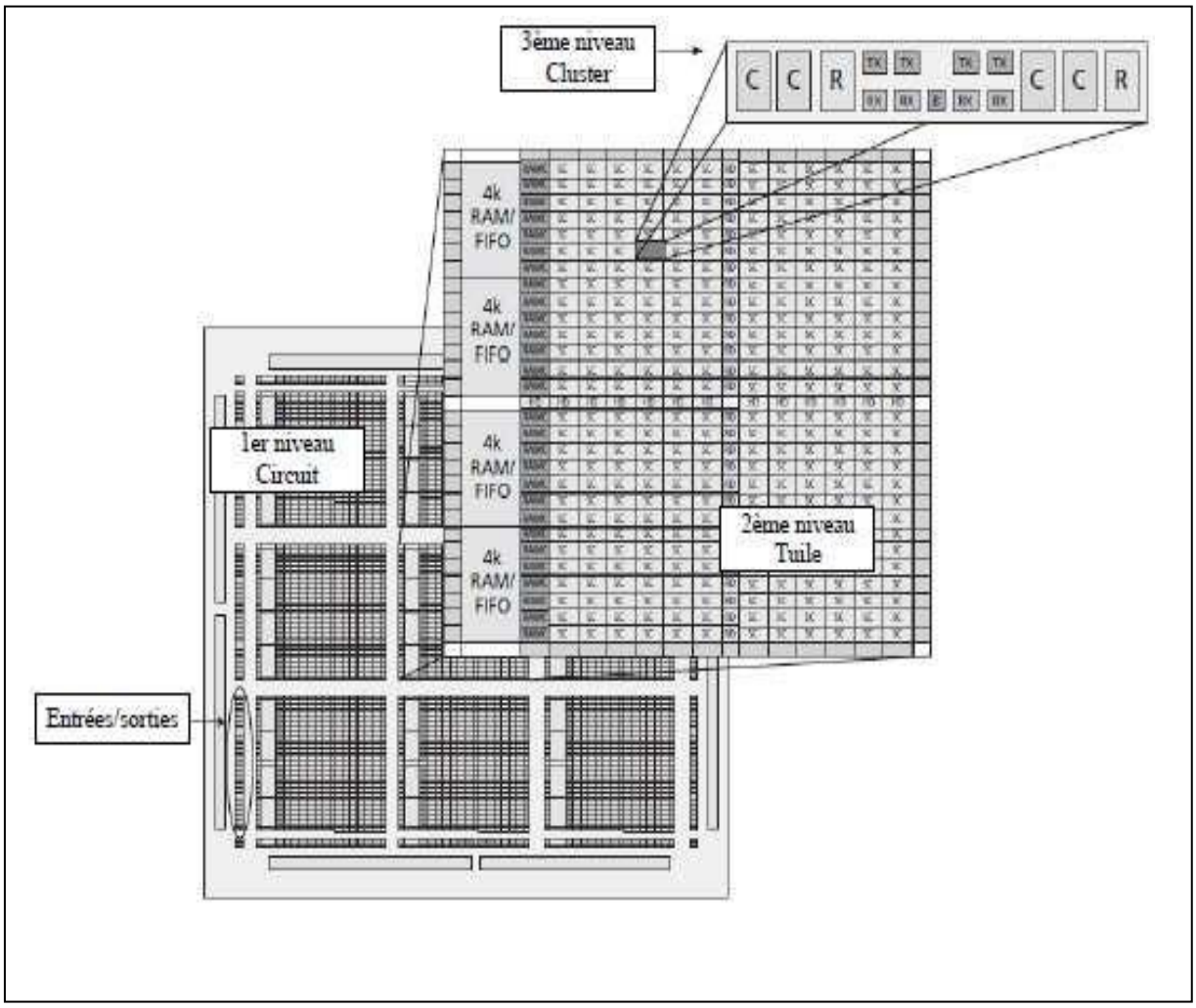


Figure 3.5 : Exemple d'Architecture hiérarchique à quatre niveaux (circuit, tuiles, clusters, éléments configurables) que l'on rencontre fréquemment dans les circuits ACTEL [35].

3.5.4.Architecture de type mer de portes

Elle est composée hiérarchiquement et le routage est de type logarithmique. Ce type de topologie fut utilisé par Xilinx pour sa série 6000. Mais ces composants n'ont pas eu de succès, commercialement parlant, par manque d'outils de CAO capables de les exploiter correctement. Peut être, reverrons-nous un jour des architectures de ce type ressortir sur le marché [28].

3.5.5.Architecture Spacetime

La société Tabula vient, de son côté, de concrétiser pour la première fois son architecture Spacetime avec sa famille de réseaux logiques programmables (3PLD) baptisée Abax. Spacetime revendique l'appellation 3D en prenant le temps comme troisième dimension pour optimiser l'utilisation des ressources sur la puce et ainsi accroître la densité de logique disponible. La société donne du volume à la matrice matérielle en la reconfigurant très rapidement : 1,6 milliard de fois par seconde (1,6 GHz), soit environ un million de fois plus rapidement que les FPGA actuels qui chargent la configuration à partir d'une mémoire externe. Pour faciliter la gestion de cette structure temporelle et permettre une reconfiguration

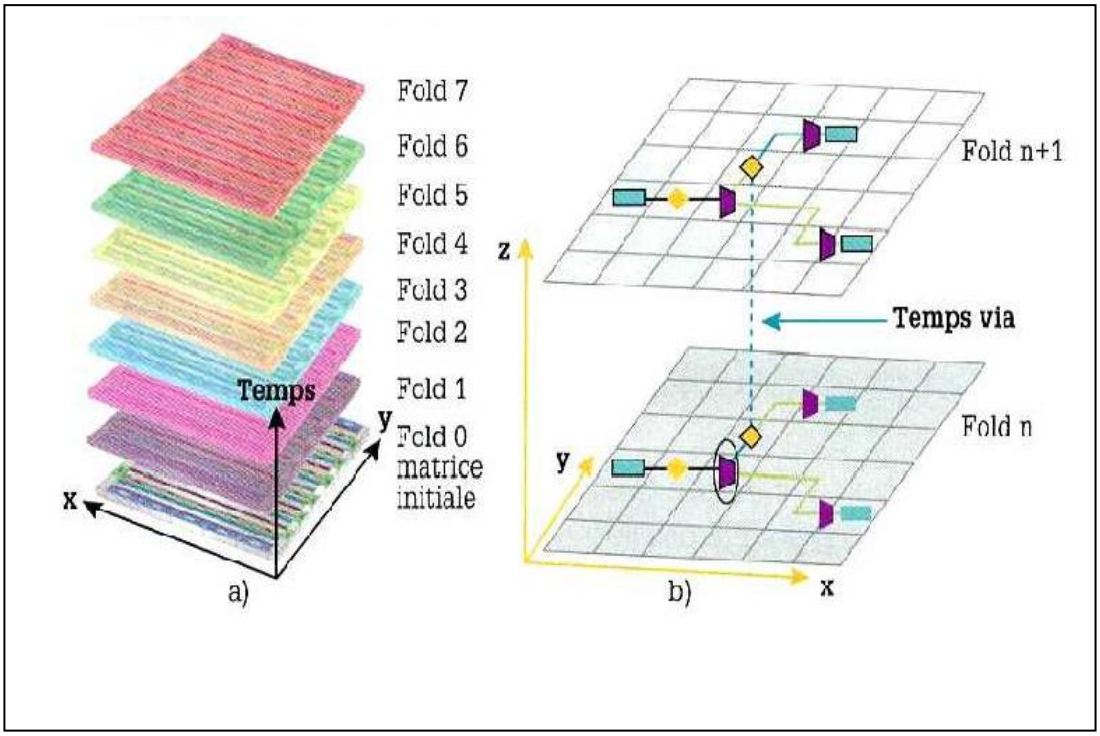


Figure 3.6 : Architecture Spacetime (société Tabula) [36].

ultra rapide de la logique, la succession de configurations est stockée localement sur la puce. Par rapport aux solutions conventionnelles, la société annonce pour ses FPGA fabriqués dans le procédé 40 nm de TSMC, des gains respectifs de 2,5 pour la densité, de 2 pour la capacité mémoire et de 3,7 pour les performances DSP. Tabula précise également que cette reconfiguration est transparente pour l'utilisateur, celui-ci pourra donc suivre une méthodologie de conception classique. Dans l'architecture Spacetime, la matrice matérielle est découpée en plusieurs zones de blocs logiques basés sur des LUT, alternant avec des bancs de mémoire Ram et des multiplexeurs pour distribuer les signaux entre ces diverses entités. Suivant le même principe que celui de la logique programmable, l'axe des temps donne un effet de volume à la mémoire, multipliant sa capacité initiale ainsi que le nombre de ports. Lorsqu'en 2003, Steve Teig, président et CTO de Tabula, mit au point le concept de cette architecture Spacetime, le premier travail de son équipe fut d'étudier la faisabilité d'outils de CAO de type classique pour « animer » cette architecture. Les premiers membres de la famille Abax (A1EC02,-03,-04,-06) embarquent entre 220 000 et 630 000 LUT par FPGA. Ils possèdent tous 5,5 Mo de Ram, 920 E/S parallèles, 44 PLL ainsi que 48 SerDes opérant entre 55 Mbits/ s et 6,5 Gbits/ s. Le FPGA A1EC06 bénéficie en plus de 1 280 blocs MAC (multiplier/accumulate). L'échantillonnage de ces FPGA est prévu en juillet, pour une

production en volume en octobre.

Quelle que soit l'architecture choisie, les éléments constitutifs d'un FPGA sont toujours à peu près les mêmes. Chaque fabricant ayant ses variantes par rapport à un autre. Nous pouvons citer un certain nombre de ces éléments [36].

3.6.Ressources fonctionnelles configurables

Dans la plupart des cas l'élément logique configurable de base des FPGA se compose d'une LUT (Look Up Table5) avec un nombre d'entrées allant de 4 à 8 pour les dernières générations, d'une chaîne de propagation rapide de la retenue et d'un registre de sortie afin d'assurer la synchronisation des signaux (très utile pour l'implémentation de calculs pipelinés). Ces éléments configurables peuvent être rassemblés en clusters hiérarchiques afin de favoriser une connectivité locale et rapide (cas des composants Altera). Les éléments logiques configurables appelé SLICE chez Xilinx. Entre deux générations successives de FPGA Xilinx, qu'un SLICE Virtex4 ne correspond pas à un SLICE Virtex6.

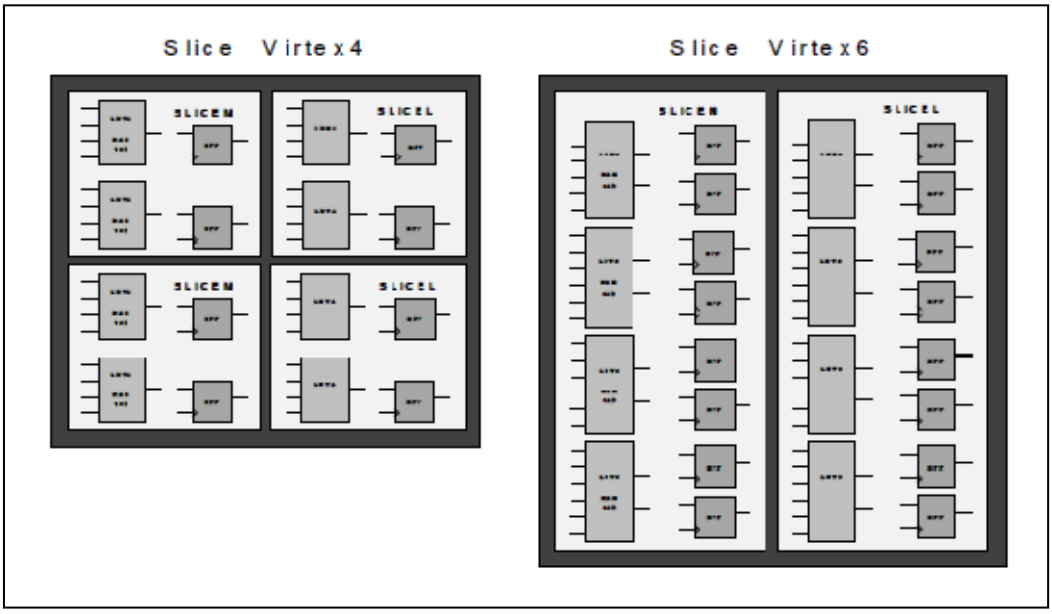


Figure3.7 : Eléments logique configurables(circuits Xilinx Virtex4 et virtex4[36]

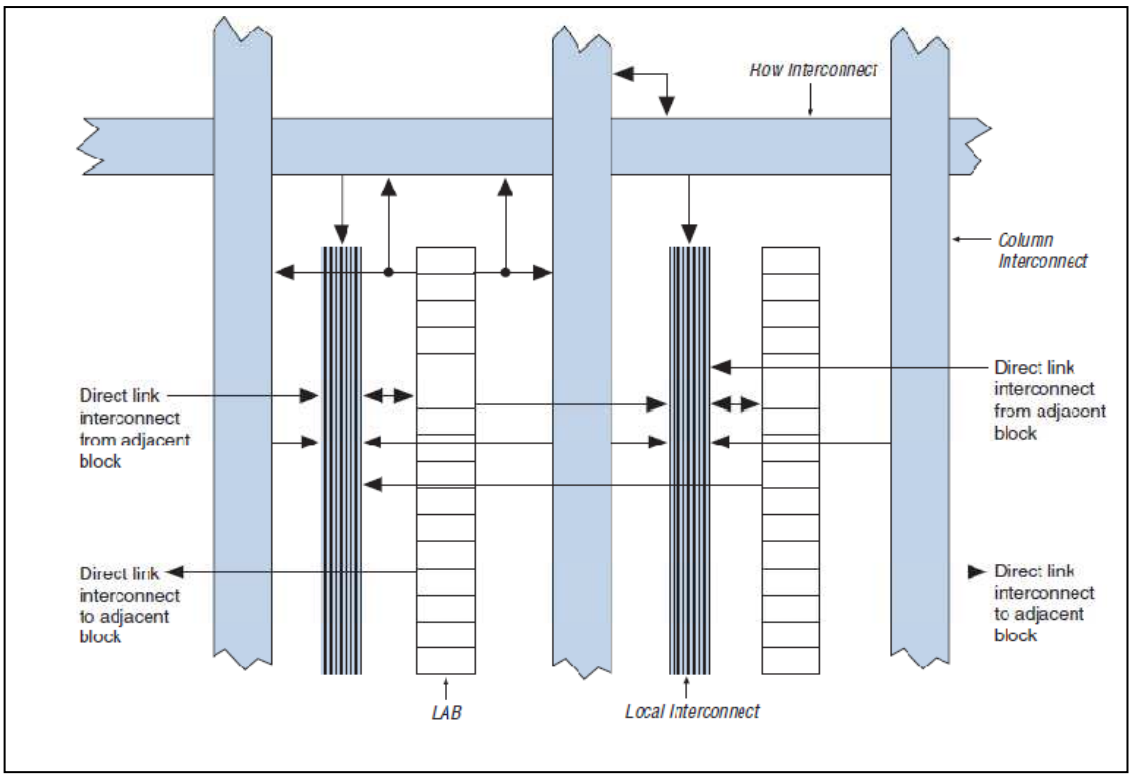


Figure 3.8 : Eléments logiques (LAB) configurables (Logic Blocks, Cyclone II) d'Altera [33].

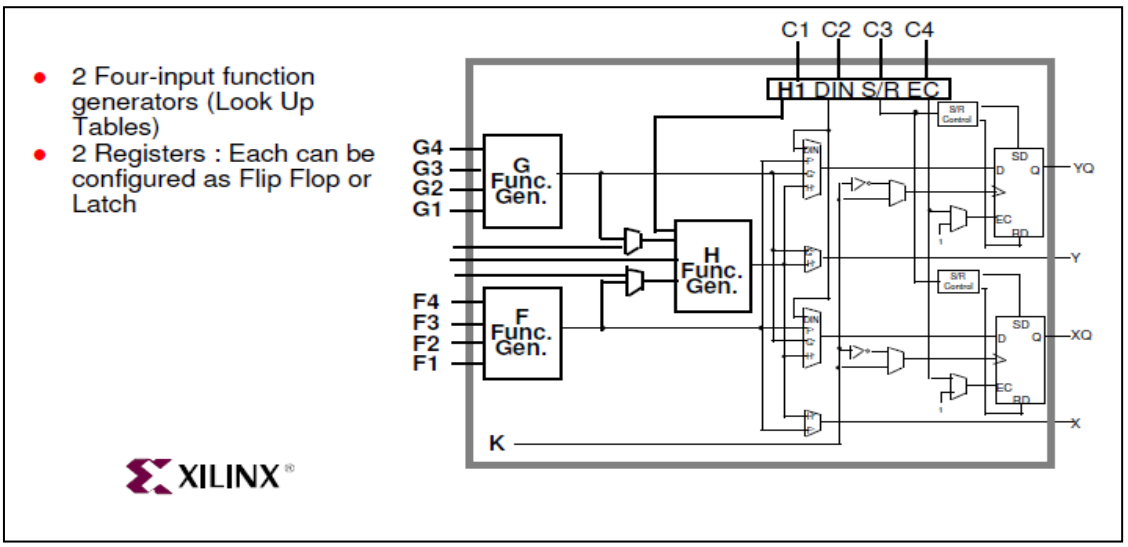


Figure 3.9 : Eléments logiques configurables (simplifiés) des circuits XC4000 Configurable (LAB) Logic Blocks de XILINX [31].

Remarque : Une LUT peut être considérée comme une petite mémoire RAM dans laquelle on mémorise la table de vérité d'une fonction logique. Une LUT à 4 entrées et 1 sortie, classiquement utilisée dans les FPGA, est donc équivalente à une RAM 16 bits. Quelques rares circuits ne comportent pas de LUT et sont constitués de cellules configurables basées sur des multiplexeurs. Ce fut le cas du circuit Xilinx XC6000 qui n'est plus disponible. C'est le cas pour certains composants ACTEL comme les familles FPGA de technologie FLASH ProASIC-3 et Fusion ainsi que la famille de FPGA de technologie anti-fusible Axcelerator.

Rapidement, afin de réaliser complètement des applications modernes, les FPGA ont dû se doter d'éléments configurables de mémorisation (apparition en 1999 dans les composants Virtex et Apex de Xilinx et Altera). Sans ceux-ci la mémoire synthétisée doit être distribuée sur les LUT, ce qui laisse peu de place pour les traitements [28,31,37].

Ce type de circuit permet de profiter du parallélisme de calcul offert par l'architecture matérielle et du contrôle séquentiel efficace offert par le système programmable (microprocesseur). Aussi en tirant parti des propriétés respectives des systèmes programmables et des systèmes reconfigurables il est possible d'améliorer l'adéquation du système global avec l'application développée. Dans ce cas l'utilisation de méthodes de conception conjointe logicielle/matérielle est indispensable et demande un effort important en développement d'outils.

Plusieurs architectures existent aujourd'hui, la figure 3.10 illustre ces différentes possibilités. Dans certains circuits, la partie matérielle configurable et la partie programmable sont séparées par un bus spécifique. La partie programmable comprend le système à microprocesseur dans son ensemble : cœur de processeur, mémoires caches, périphériques, interface etc … Ce fut le cas du premier circuit commercial embarquant un cœur de processeur, le circuit Altera Excalibur qui regroupait une matrice FPGA APEX 20KE et un cœur de processeur ARM9 (32 bits) fonctionnement à 100MHz accompagné de deux fois 8kilo-octets de mémoire cache (instructions + données). Malheureusement, au moment de la sortie de ce composant, les outils n'étaient pas matures pour permettre une utilisation simple et efficace dans un contexte industriel.

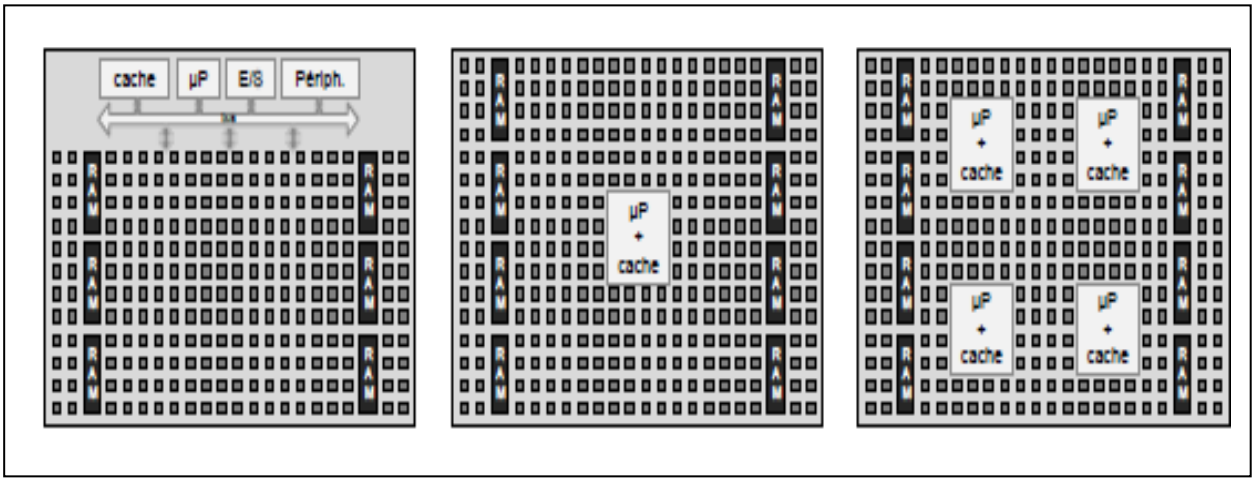


Figure 3.10 : Trois architectures possibles de circuits mixtes FPGA microprocesseur(s) [35].

Certains circuits embarquent *profondément* au cœur de la matrice configurable le ou les processeurs. Ceux-ci ne sont pas nécessairement accompagnés de leur système, mais des outils logiciels permettent de configurer une partie de la logique pour constituer le système complet. Cela permet ainsi une plus grande flexibilité dans le choix du système programmable. C'est cette solution qui fut choisie par Xilinx pour son premier composant mixte Virtex-II Pro. Celui-ci était composé d'une matrice Virtex-II Pro et de un à quatre cœurs IBM PowerPC 405 (32 bits) cadencés à 400MHz disposant de deux fois 16 kilo-octets de mémoire cache (instructions et données). Cette même architecture est toujours utilisée pour les composants Virtex de dernière génération.
Il faut noter qu'il est possible de réaliser un circuit mixte sans disposer physiquement d'un cœur de processeur embarqué dans le circuit FPGA. Dans ce cas l'utilisation d'un cœur synthétisable (dit *cœur soft*), fourni gratuitement par le fabricant de circuit, est une solution très efficace. Les cœurs synthétisables 32 bits Altera NIOS et Xilinx MicroBlaze sont par exemple très utilisés. Bien entendu, les performances de ces cœurs sont inférieures à celles des cœurs embarqués, mais ils permettent une plus grande flexibilité de configuration [28].

3.7.Tendances

Quelques tendances apparaissent ces dernières années dont l'objectif est d'augmenter l'efficacité des FPGA. Sans vouloir être exhaustif nous pouvons en décrire quelques unes. Au niveau de la mémoire de configuration, les densités de configuration et la maitrise des technologies ont permis l'émergence de composants de technologie mixte FLASH-SRAM, tel que les composants Lattice XP2 et Xilinx Spartan AN. L'offre commerciale a évolué en proposant de plus en plus des séries spécialisées à l'intérieur d'une même famille de composants

3.8.Conclusion

La part de marché des FPGA dans le marché globale des circuits matériels pour l'électronique numérique ne cesse d'augmenter. Les évolutions technologiques et architecturales qui ont eu successivement lieu depuis le début des années 2000 ont fait de ces circuits de réels et rentables alternatives aux classiques ASIC. Avec ces évolutions c'est tout un nouveau domaine de l'électronique numérique qui s'est ouvert. Aujourd'hui les FPGA sont utilisés dans tous les domaines, des systèmes embarqués aux systèmes de communications, ils sont au cœur d'un important champ de recherche académique et industrielle.

Chapitre IV :
SYNTHESE ET IMPLEMENTATION

4.1Introduction

L'implémentationdenotrearchitecturesuruncircuit FPGA consisteenla programmationducircuitàl'aidedulangagededescriptiondehautniveau VHDL. Nousallonsdoncdécrirelecircuitsouslaformed'algorithme VHDL,etce,après avoir choisi les contraintes d'implémentation. La seconde étape consiste à effectuer une simulation fonctionnelle du circuit a l'aide du simulateurlsim de la plateforme ise 14.2 de XILINX

4.2 Introduction auparallélisme

4.2.1 Définition:

Lesordinateursparallèlessontdesmachinesquicomportentunearchitecture parallèle,constituéedeplusieursprocesseursidentiques,ounon,quiconcourent autraitementd'uneapplication.

Laperformanced'unearchitectureparallèleestlacombinaisondesperformances desesressourcesetdeleuragencement.(Latence,débit).

4.2.2Le parallélisme est la conséquence:

- Besoin desapplications
- Calculscientifique
- Traitementd'images
- Bases dedonnées
- QuidemandentdesressourcesenCPUetentempsdecalculdeplusenplus importantes
- Limitesdesarchitecturesséquentielles
- Performance
- Capacitéd'accèsàlamémoire
- Tolérance auxpannes

4.3 Architectures matérielles de l'AES

En général, il existe trois types d'architectures pour l'implémentation du protocole de chiffrement et déchiffrement AES :

- **L'architecture série-série,** où les quatre blocs de 32 bits constituant le message d'un round sont exécutés en série et les rounds en séquentiel.
- **L'architecture parallèle-série**, où les quatre blocs de 32 bits constituant le messaged'un round sont exécutés en parallèle et les rounds en séquentiel.
- **L'architecture parallèle-pipeline,** où les quatre blocs de 32 bits constituant le message sont exécutés en parallèle et les rounds en pipeline [25].

Concernant le choix de l'architecture adéquate à la conception de notre crypto-système AES, nous constatons que l'architecture série-série occupe moins de ressources par rapport aux autres architectures, mais en revanche un temps très important est perdu en exécutant les quatre blocs de 32 bits séquentiellement. De même, l'architecture parallèle-pipeline est dédiée aux applications fortes débit, mais elle nécessite beaucoup de ressources matérielles.
Ce raisonnement nous a conduit de choisir une architecture parallèle-série pour la conception de notre système. Cette architecture offre un bon compromis entre la rapidité d'exécution et les ressources utilisées. En effet, la matrice d'entrée « state » de 128 bits sera découpée en quatre blocs de 32 bits qui seront traités en parallèle. L'exécution d'un cycle a besoin de 4 blocs de 32 bits pour la transformation *MixColumns*et 16 *Sboxs*implémentés en parallèle, travaillant avec des données indépendantes, ce qui augmente le débit de cryptage/décryptage.
La figure 4.1 illustre l'architecture parallèle-série globale d'un round AES.

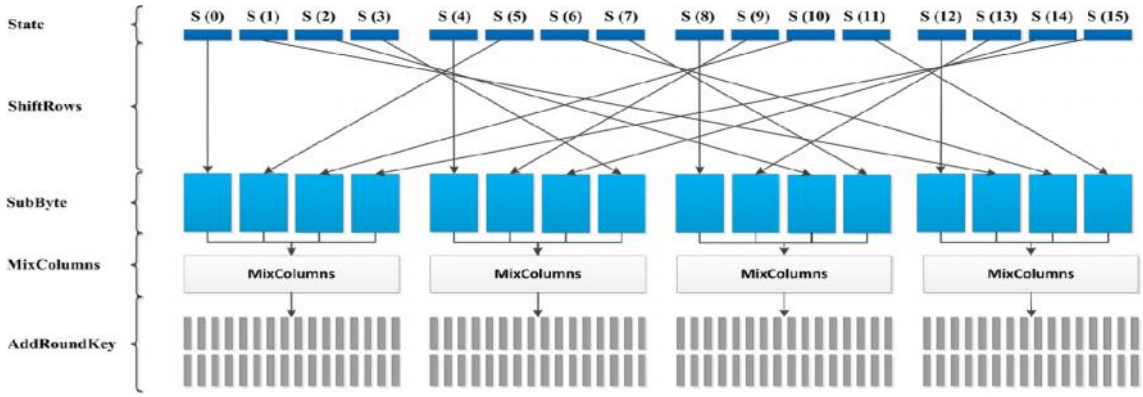


Figure 4.1. Architecture parallèle-série d'un cycle AES.

4.4. Présentation de l'architectureproposée

4.4.1 .Architecture de base:

L'idée générale dans cette architecture porte des registres supplémentaires sont ajoutéeaumilieudelalogiquecombinatoire,desorteque plusieursblocsde donnéespeuventêtretraitésparlechiffredanslemêmetemps,exécutélesdeux processus(générationdecléetgénérationdutextecrypter(DATA))defaçon parallèle.

4.4.2Architecture pipelined'AES

Diverstypesd'architecturesmatériellespourl'algorithmeAESsontpossibles.La meilleurearchitectureenestunequiestd'avoirlemeilleurcompromisentre vitessed'horloge(débitdedonnées)etoccupationmoinslesressourcesmatériel.

DanscetteconceptionChaquetourAESestd'avoir4stages(niveaux)etchaque stagecontientlesdeuxprocessus:transformationdedonnées(data)etextension de la clé(Key) en pipeline, au sein de chaque cycle (pour 10 rondes de l'algorithmeAESde128bitsetlesdonnéescléde128bits).

L'utilisateurestinvitéàentrerletextequidoitêtrechiffréetaussilaclépar lequel il sera

capable de déchiffrer ce texte crypté retour au texte brut. Aprèsl'élargissementdelacléesterminée,maintenantprocessusdechiffrement commence.Dansleprocessusdecryptage,toutd'abord,uncycleprédoitêtre effectuéquiXORl'originalde128bitsdesdonnéesd'entréeà128bitsdelaclé originaleetlerésultatintermédiaireeststockédanslabanquederegistre.Une foislecyclepréesterminé,alorslesépreuvesrestantesdefonctionnementsont effectuées.Cescyclesonttraitéesdelamanièresuivante:

Premièrement,lesdonnées(Data)delabanquederegistreintermédiairesont luesetenvoyéàl'entréedelaS-box.Celapermettradetransformerlesdonnéesà leursdonnéescorrespondantesettransmettreàdeuxièmestagepourun traitementultérieur,etpourleprocessus(Key)justelesélémentsdeWi-1sont sélectionnépourl'opérationdeSubstitution(Subyte)alasortiedesdonnéeson faireuneaddition(Xor)entrelesélémentsdeMot(Word)(wi-4)etleséléments de Opération de substitution et rotation de mot (SubRotWord) (wi-1) les résultatsdecetteopérationsontstockédansregistrepourprochainstage.

Dansledeuxièmestageselonleprincipededécalage(ShiftRow)lesdonnées (data)sontdécaléetenregistrerpourletroisièmeniveau,pourleprocessusde extensiondelacléoneffectuéeuneadditionentrelepremierélémentdela matriceprédéfinis(RCon)Solonalgorithmeetdanscetteétapeonaconstruirele premiervecteurdelaclédepremierrondeMot(Wi).

Etpourletroisièmestage,Laopérationdemixagedescolonnes(MixCol)obtient cesdonnéesetselonl'algorithmemultiplie-lesdonnéesavecunematrice standardpourproduireunesortiedesdonnéesstocké.etpourobtientlesautre vecteursdelacléonfaireuneaddition(Xor)entrelesélémentsderegistre.

OrdanslequantièmeniveaulesdeuxsortiessontXORedetsontstockédansla banqueintermédiaireRegistreencorepourleprochaintour.

Dansledernierтour,OpérationdeMixcolonneestignoréeetlerésultatde donnéeseststockédanslamémoiredechiffrementpourl'affichagedutexte crypté. Lechoixjudicieuxdumoduleetlechemindedonnéespourunesérieparticulière sefaitparuneautomateétatfinie.

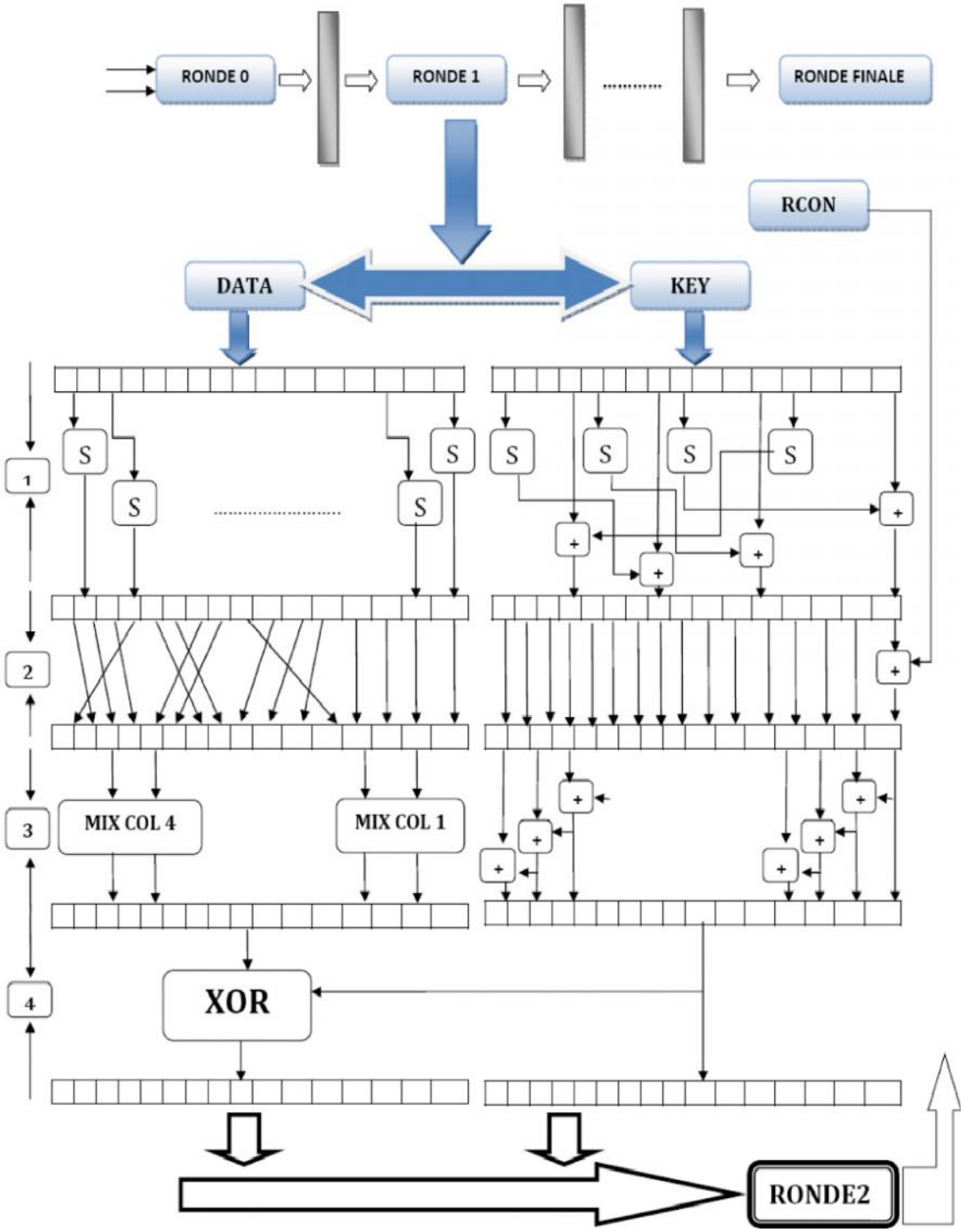


Figure.**4.2** .Architecture pipeline de l'algorithme de chiffrement AES

4.5. Description de l'environnement ISE

ISE (IntegratedSynthesisEnvironment) est le logiciel de programmation des produits Xilinx (CPLD, FPGA Spartan et Virtex…). Cet outil permet de créer des projets comportant plusieurs types de fichiers (HDL, schématique, UCF, EDIF, etc.), de compiler, de créer des contraintes d'implémentation avec des contraintes de timings sur les horloges, de déterminer l'emplacement des broches et de créer des fichiers de stimuli.

Le Navigateur de projet ISE offre un environnement de conception et regroupe tous les outils nécessaires à la conception, la simulation et à l'implémentation d'un projet. Il dispose de :

- Un éditeur de textes, de schémas et diagrammes d'états.
- Un compilateur VHDL et Verilog.
- Un simulateur.
- Des outils pour la gestion des contraintes temporelles.
- Des outils pour la synthèse.
- Des outils pour la vérification.
- Des outils pour l'implémentation sur FPGA et CPLD.

4. 6. Étapes d'implémentation sur un circuit FPGA

a.Spécification : la spécification HDL regroupe les trois modes de création d'un circuit (schematic, diagrammes d'états ou HDL). Elle est synthétisée pour générer un fichier appelé NETLIST qui décrit les interconnexions entre les registres.

b. Vérification : la vérification du design est une étape parallèle où le concepteur observe le comportement du code et vérifie s'il se comporte tel qu'il est supposé. Un simulateur simule le circuit par l'utilisation des vecteurs de test. Les vecteurs de test peuvent se présenter sous plusieurs formes, la plus courante est les **TESTBENCHS** écrits dans un langage de description matériel comme le VHDL pour entrer les instructions au simulateur en appliquant les vecteurs de test sur le code pour que le simulateur fournit les sorties du circuit.

c. Implémentation : une fois la vérification est terminée, le circuit est implémenté sur le composant en spécifiant les références exactes de celui-ci à savoir : le circuit utilisé, la fréquence de travail et les autres options spécifiques à chaque composant. Cette étape se termine par un rapport de tous les sous-programmes exécutés (les erreurs, les I/O utilisées et des données qui permettent de savoir si le composant choisi est le mieux adapté pour l'application ciblée) [38].

4.7. Résultats d'implémentation

4.7.1. Résultats de Simulation

Tout d'abord, nous commençons par vérifier le bon fonctionnement de tous les blocs de l'architecture, par des simulations que nous avons réalisé grâce à l'outil ISIM qui est inclus dans l'environnement ISE.
Nous allons donner dans ce qui suit quelques exemples de simulations des différents blocs nconstituant notre architecture.

4.7.2 .AddRoundKey

La figure 4.3 montre le résultat de simulation de l'opération *AddRoundKey*

Figure4.3. Simulation de l'opération *AddRoundKey*

4.7.3.KeyExpansion

La figure 4.3 montre le résultat de simulation de l'opération*KeyExpansion*.

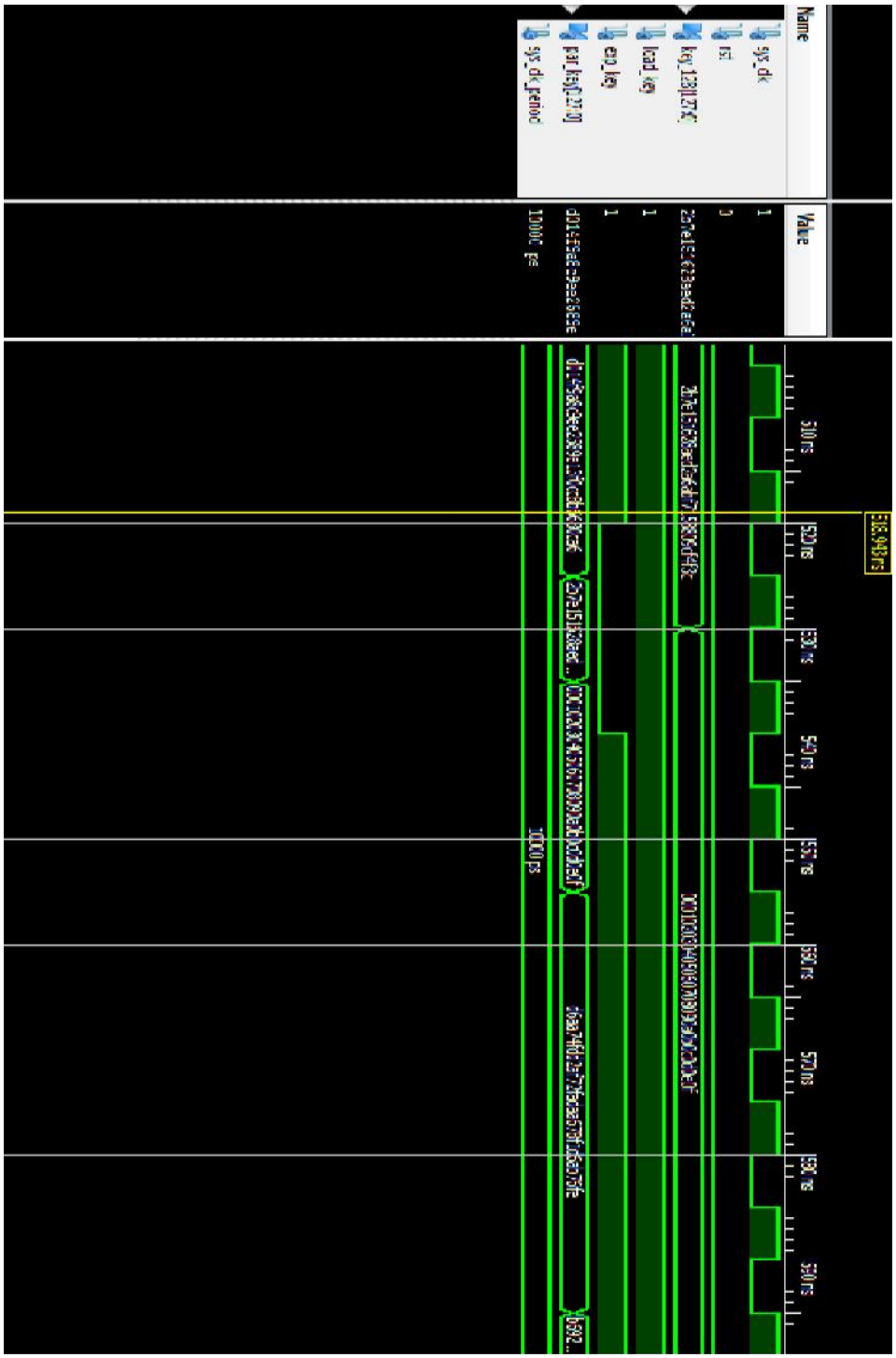


Figure 4.4. Simulation de *KeyExpansion*.

4.7.4. MixColomns

La figure 4.3 montre le résultat de simulation de l'opérationMixColomns

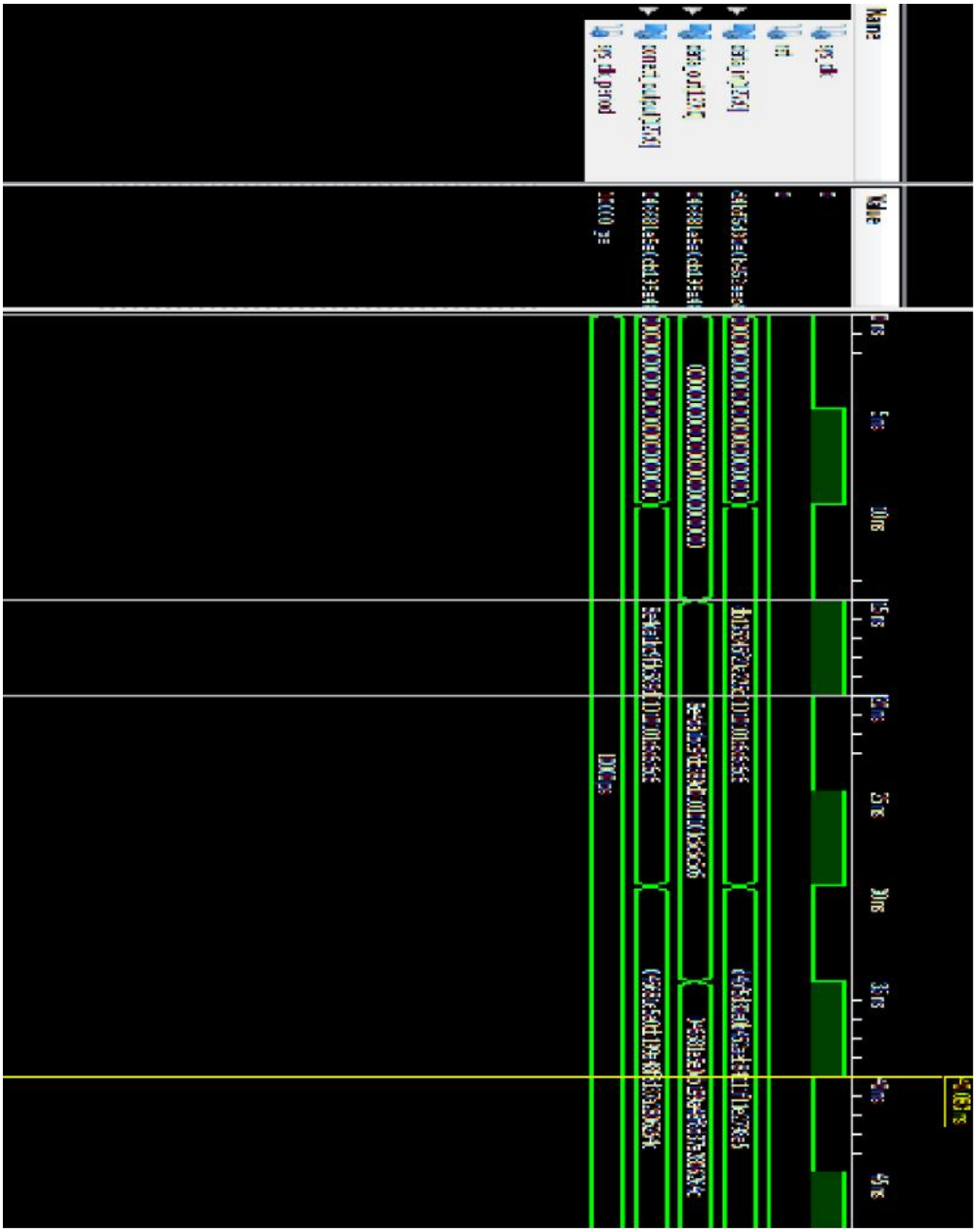


Figure 4.5. Simulation de MixColomns

4.7.5.ShiftRows

La figure 4.3 montre le résultat de simulation l'opération *ShiftRows*.

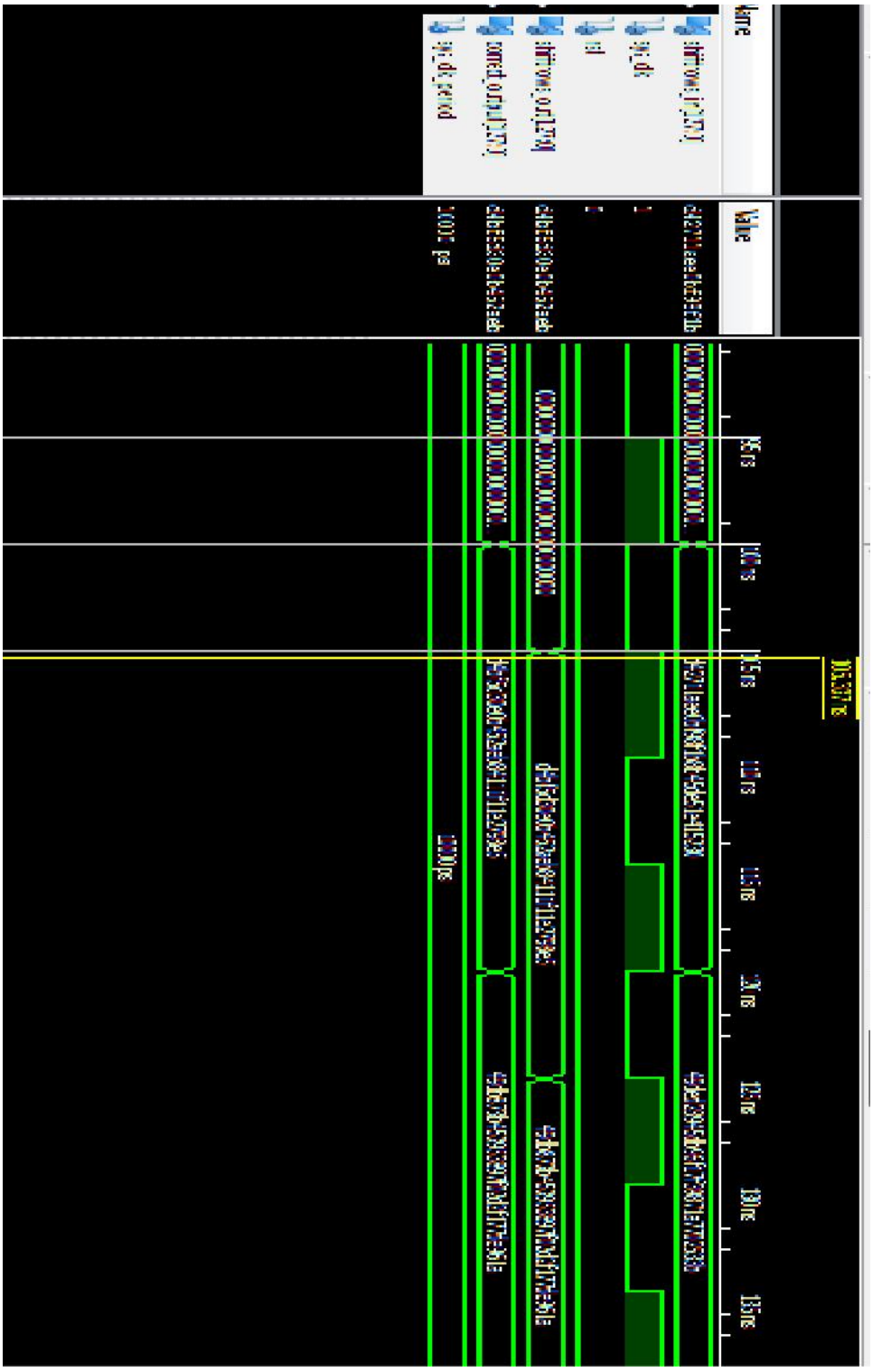


Figure4.6. Simulation de l'*ShiftRows*

4.7.6. *SubByte*

La figure 4.5 montre le résultat de simulation de SBox implémentés en parallèle.

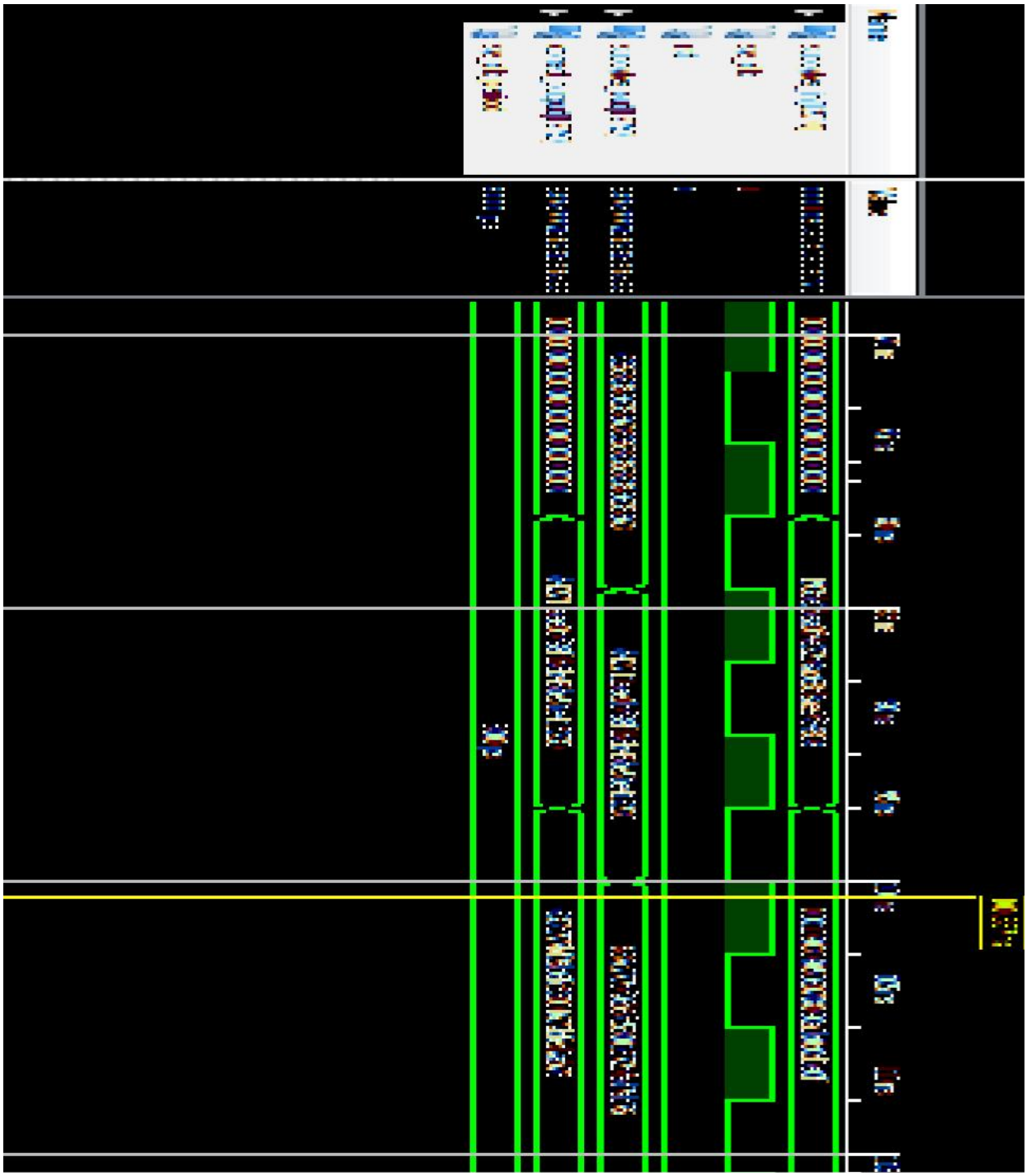


Figure4.7. Simulation de *SubByte*

4.7.7. Chiffrement

Après la vérification du fonctionnement de chaque composant individuellement, nous simulons le crypto système en entier. La figure 4.8 représente le résultat de la simulation duprocessus de chiffrement.

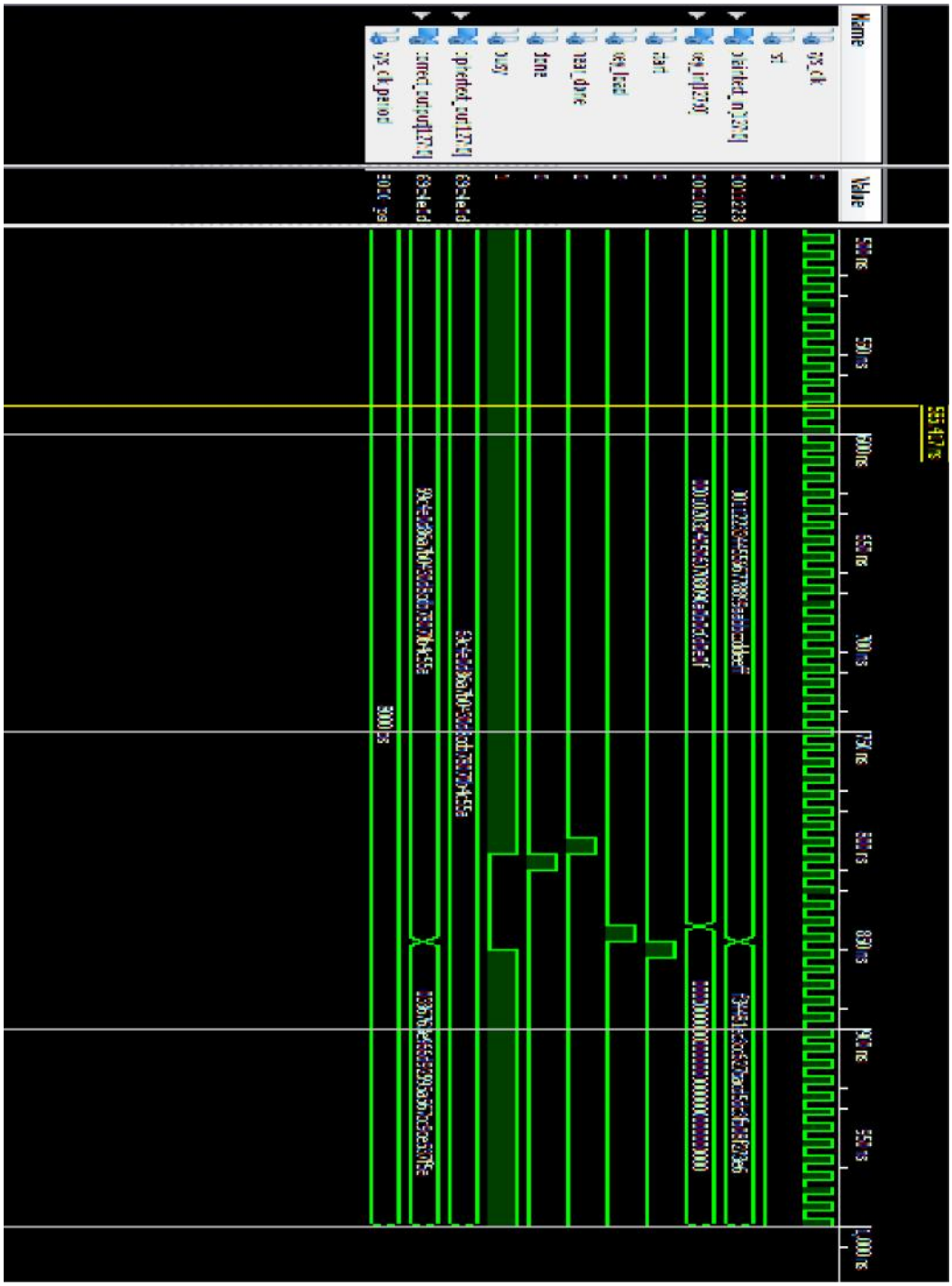


Figure 4.8. Résultat de simulation de chiffrement.

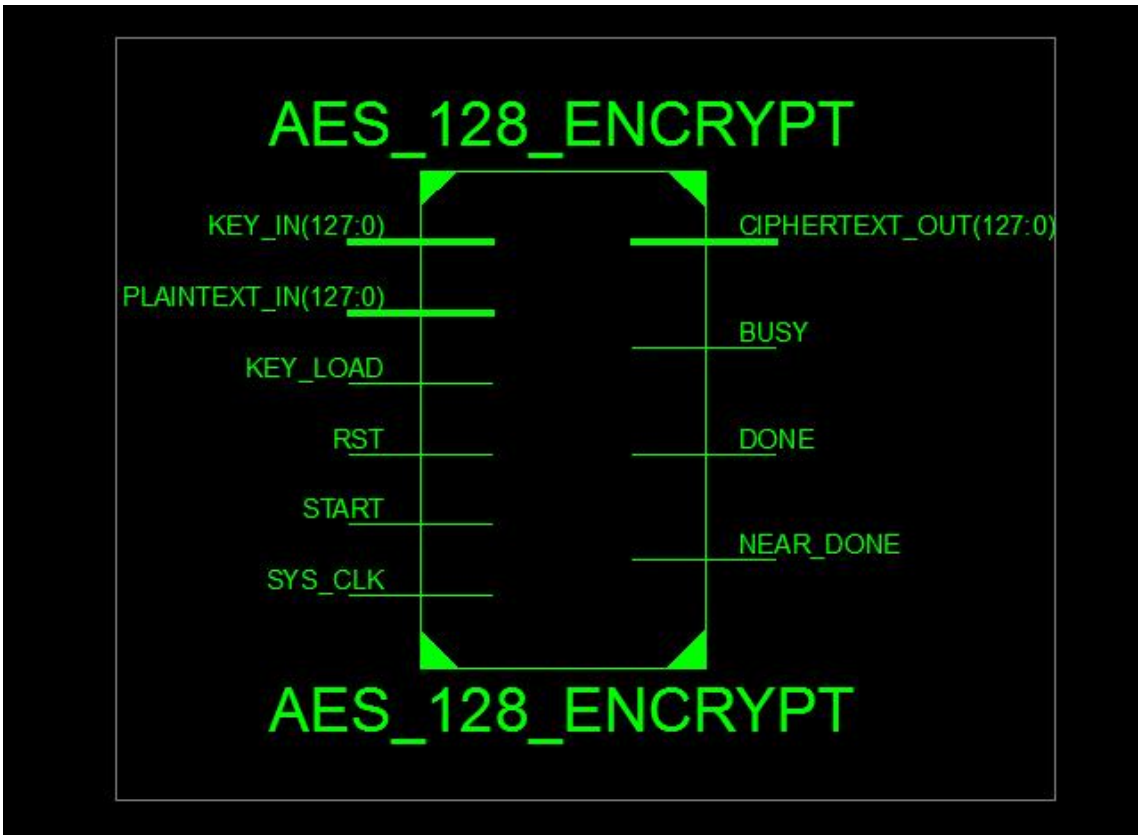


Figure 4.9. Schéma RTL de l'AES

4.8.Conclusion

Nous avons introduit ce chapitre par une brève description des différents outils nécessaires pour le développement de notre crypto-système en présentant les différentes étapesd'implémentation sur un circuit programmable, et en décrivant le langage de descriptionmatérielle utilisé.
Par la suite, nous avons exposé nos résultats de simulations de chaque bloc fonctionnel pour attester leurs bons fonctionnements, en les validant par un exemple Nous l'avons suggéré

Conclusion Générale

Au cours de ce travail, nous avons décrit la cryptographie depuis sa naissance jusqu'à l'heure actuelle en mettant en relief les deux types les plus utilisés dans les temps modernes à savoir la cryptographie symétrique et asymétrique. L'AES (Advanced Encryption Standard) s'est imposé comme étant l'algorithme symétrique le plus sécurisé qui a été choisi par le NIST (National Institutes of Standards and Technology). L'intérêt de l'algorithme AES dans le domaine de la cryptographie n'est plus à démontrer, et son Intégration sur circuit numérique (FPGA) est la cible d'un grand nombre de travaux, notamment dans les domaines militaire. Ce qui a motivé les chercheurs à développer des architectures softwares et matérielles pour implanter en temps réel cet algorithme dans les deux modes cryptage et décryptage. Dans ce mémoire nous avons implanté sur circuit FPGA une architecture pipeline de l'algorithme AES. Nous avons utilisé la plate-forme logicielle ISE 14.2 pour réaliser cette implantation en langage VHDL. Nous avons utilisés le simulateur lsim de ISE pour effectuer les simulations des différentes phases de l'algorithme AES à savoir les fonctions SubBytes, ShiftRows, MixColumns et AddRoundKey et enfin l'algorithme globale de cryptage AES. Les résultats de simulation montrent bien la validité des données à chaque phase et le texte chiffré est bien correcte.

Nous proposons comme perspective d'étendre cette implantation pour inclure une optimisation de la puissance consommée et la surface occupée sur le circuit FPGA cible.

Bibliographie

[**1**]: Bruce Schneier “Cryptographie Appliquee“ 2emme edition ITP 1997

[**2**]: A.Menezes, P.Van Oorschot and S.Vanstone, “Hand Book of Applied Cryptography” crcpress 1996.

[3] C. Figlarek, l’utilisation de la cryptographie dans les échanges de données médico-sociale, mémoire de l’école national de la santé publique, Rennes, France, 2000.

[4] D.G.Mesquita, Architectures reconfigurables et cryptographie : une analyse

[**5**] David Kahn. Codebreakers : L'histoire de l'écriture secrète. 2008. (Cité en page 8.)

[6] Petite histoire de la cryptologie. Apprendre-en-ligne. [En ligne] [Citation : 27 Janvier 2015.] http://www.apprendre-en-ligne.net/crypto/histoire/.

[7] Le chiffre de César. La cryptographie expliquée. [En ligne] [Citation : 26 Janvier 2015.] http://www.bibmath.net/crypto/index.php?action=affiche&quoi=substi/cesar

[8] Le chiffre de Vigenère. La cryptographie expliquée. [En ligne] [Citation : 2 Février 2015.] http://www.bibmath.net/crypto/index.php?action=affiche&quoi=poly/vigenere.

[9] L'affaire du télégramme de Zimmermann - Cryptographie et diplomatie. La cryptographie expliquée. [En ligne] [Citation : 4 Février 2015.] http://www.bibmath.net/crypto/index.php?action=affiche&quoi=debvingt/zimmermann.

[**10**] Breaking Germany's Enigma Code. BBC. [En ligne] 17 Février 2011. [Citation : 09

[11] La machine Enigma. apprendre-en-ligne. [En ligne] [Citation : 16 Janvier 2015.] http://www.apprendre-en-ligne.net/crypto/Enigma. 2015.] http://www.bbc.co.uk/history/worldwars/wwtwo/enigma_01.

[**12**] Jean-Guillaume Dumas, Jean-Louis Roch, Eric Tannier, Sébastien Varrette. Théorie des codes, Compression, cryptage, correction. Paris : Dunod, 2013. 978-2-10-059911-0

[**13**] Whitfield Diffie, Martin Hellman. New Directions in Cryptography. 6 Novembre 1976

[14] W. Di_e et M. E. Hellman. New Directions in Cryptography. In IEEE Transactions on Information Theory, 1976. (Cité en page 9.)

[15] : Cryptographie - partie 4 : cryptographie à clé publique .

[16] P.A. Fouque, Cryptographie appliquée, Techniques de l’Ingénieur, traité Sécurité des systèmes d’information, 2003.

[18] Niels Ferguson, Bruce Schneier. Cryptographie en pratique. Paris : Vuibert, 2004. 2-7117-4820-0.

[19] La cryptographie quantique en bref. Genève, Université de http://www.unige.ch/sciences/lenya/pdf/info_crypographie_quantique.pdf.

[20] J. Daemen, V. Rijmen, The Design of Rijndael, AES : The Advanced Encryption Standard, ISBN 3540425802, 2001.https://autonome-antifa.org/IMG/pdf/Rijndael.pdf

[**21**] Joan Daemen and Vincent Rijmen, AES submission document on Rijndael, June 1998.

[22] FIPS PUB 197, Advanced Encryption Standard (AES), National Institute of Standards and Technology, U.S. Department of Commerce, November 2001. http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf.

[23] A Polynomial Description of the Rijndael Advanced Encryption Standard / Joachim Rosenthal /Department of Mathematics University of Notre Dame Notre Dame, Indiana 46556, USA/ February 25, 2003 www.math.uzh.ch/fileadmin/user/rosen/publikation/rijndael.pdf.

[24] The Design of Rijndael/J Daemen, V Rijmen – 2001 http://jda.noekeon.org/JDA_VRI_Rijndael_2002.pdf.

[**25]** O. Azzouzi, Système embarqué flexible pour un chiffrement hybride symétrique / asymétrique, Mémoire de Magistère en Informatique, Ecole Nationale Supérieure, Oued Smar, Algérie, 2014.

[26] Marcelo B. de Barcelos Design Case, "Optimized performance and area implementation of Advanced Encryption Standard in Altera Devices, by

[27]. http://dumas.ccsd.cnrs.fr/dumas-00574220/document.

[**28]**. http://perso.univ-st- etienne.fr

[29]. L'enseignement des systèmes numériques complexes _Patrice Kadionik, Patrice Nouel, Ahmed Ben Attitalah, Philippe Dondon _ ENSEIRB-IXL BP 99 33402 TALENCE Cedex kadionik@enseirb.fr.

[30]. http://www.ni.com/white-paper/8043/fr/.

[**31**]. http://www.xilinx.com/.

[**32**]. http://www.yannthoma.com/research/these/These_Thoma_chap2.pdf.

[33]. Cyclone II Device Handbook, Volume 1.pdf.

[34]. Introduction aux FPGA_Mickaël Dardaillon_M2RTS.pdf.

[35]. www.techniques-ingenieur.fr/.

[36]. Magazine electronique N 4

[37]. WWW.ALTERA.COM.

[38] B. M. Attar, A. Amini, Contribution à La Conception d'un Système d'authentification Cryptographique Embarqué sur FPGA, Mémoire de Master en INFOTRONIQUE, Université M'Hamed BOUGARA Boumerdes, Algérie, 2013.