**République Algérienne Démocratique et Populaire**

**Ministère de l'Enseignement Supérieur et de la Recherche Scientifique**

**Université Dr. Yahia Farès de Médéa**



***Faculté des Sciences et de la Technologie***

***Département du Génie Électrique et de l'Informatique***

***Projet de fin d'étude pour l'obtention***

***du diplôme de Master en Génie électrique***

***Option : Systèmes Intelligents et Communicants***

**Thème**

# Implémentation de l'Algorithme de Cryptage AES sur un circuit FPGA

**Réalisé par :**

KHIDER Ali et YAALA Mohamed

**Encadré par :**

Dr. OULD ZMIRLI Mohamed

*Année universitaire*

*2012-2013*

# *Remerciements*

Nous remercions le dieu pour nous avoir donné le courage, la patience et la force nécessaire pour venir au bout de ce travail.

Nous tenons à présenter nos vifs remerciements à notre promoteur Dr. OULD ZMIRLI Mohamed Maître de Conférences à l'université de MEDEA, d'avoir proposé et dirigé ce travail de si près, avec qualité remarquable. Nous avons grandement apprécié ses compétences et ses qualités scientifiques ainsi que son intelligence et son intuition, à travers un suivi régulier, une grande disponibilité accordés à l'avancement des travaux.

Sans oublier le chef de laboratoire de télécommunications Mr. GUDEMANI Mohamed et son assistant qui nous ont aidés pendant la réalisation de ce projet au niveau du laboratoire.

Nous saisons aussi cette occasion pour remercier l'ensemble des enseignants du département du génie électrique et de l'informatique ayant consacré leurs vies sur la voie noble de l'enseignement.

KHIDER Ali ET YAÄLA Mohamed

Juin 2013

# *Dédicace*

*Je dédie ce modeste travail à :*

*Toute ma famille,*

*Tous mes amis,*

*Et tous ceux qui m'aident de près ou de loin*

*YAALA Mohamed*

# *Dédicace*

*Je dédie ce modeste travail*

*A mes très chers parents, pour leur soutien, leur*

*Patience et leur amour*

*A ma grand-mère*

*A mes chères sœurs*

*A mon cher frère*

*A tous mes amis*

*KHIDER Ali*

**ملخص**

هذا المشروع يشتمل على تصميم وانشاء خوارزمية AES الخاصة بعمليات التشفير وفك التشفير للبيانات ذات 128 بت على الرقاقة الإلكترونية FPGA. هذا التصميم تم تحسينه لاستهلاك أقل موارد الرقاقة مع أكبر سرعة معالجة ممكنة. هذا التصميم تمت محاكاته، تركيبه على البطاقة: XtremeDSP Starter Kit ذات الرقاقة Spartan-3A DSP Edition 1800A لشركة XILINX.

**كلمات مفتاحية:** التشفير, علم التشفير, علم كسر التشفير, نظام التشفير AES, فك التشفير, Rijndael, FPGA, VHDL, Spartan-3A DSP.

## Résumé

Ce projet consiste à la conception et l'implémentation matérielle de l'algorithme de chiffrement et déchiffrement AES des données de 128 bits sur un circuit FPGA. L'architecture développée a été optimisée pour réduire au minimum la consommation des ressources matérielles avec une vitesse d'exécution plus rapide. Cette architecture a été simulée, synthétisée et mise en application dans une XtremeDSP Starter Kit équipé du circuit Spartan-3A DSP 1800A Edition de XILINX.

**Mots clés :** AES, chiffrement, cryptage, cryptanalyse, cryptographie, déchiffrement, décryptage, FPGA, Rijndael, Spartan-3A DSP, VHDL.

## Abstract

This project consists on the design and the implementation of the encryption and the decryption of AES algorithm with 128 bits of data in FPGA circuit. This architecture was optimized to reduce the consumption of the material resources with a faster speed of execution. This architecture is simulated, synthesized and implemented in a XtremeDSP Starter-Kit equipped with the circuit Spartan-3A DSP 1800A Edition of XILINX.

**Key words:** AES, cipher, cryptanalysis, cryptography, decryption, encryption, FPGA, inverse cipher, Rijndael, Spartan-3A DSP, VHDL.

# Sommaire

**Résumé** ..... i
**Sommaire** ..... ii
**Liste des figures** ..... v
**Liste des tableaux** ..... vii
**Introduction générale** ..... 1
**Chapitre I : Description de l'algorithme de chiffrement AES** ..... 3
I.1. Introduction à la cryptographie ..... 4
I.1.1. Généralités sur la Cryptologie ..... 4
I.1.2. Historique de la Cryptographie ..... 4
I.2. Principe de Chiffrement et Déchiffrement ..... 5
I.3. La Cryptanalyse ..... 6
I.4. Chiffrement par Blocs ..... 6
I.5. Deux Méthodes pour le Chiffrement ..... 8
I.5.1. Chiffrement Asymétrique ..... 8
I.5.2. Chiffrement Symétrique ..... 9
I.6. AES (Advanced Encryption Standard) ..... 10
I.6.1. Origine d'AES ..... 10
I.6.2. Domaines d'application de l'AES ..... 11
I.6.3. Les résultats d'un concours ..... 12
I.6.3.1. Avantages et inconvénients des participants aux concours ..... 14
I.6.3.2. Le choix : Rijndael ..... 15
I.6.4. Architecture de l'algorithme AES de Rijndael ..... 16
I.6.4.1. Propriété des transformations ..... 17
I.6.4.2. Description de l'architecture (AES-128, 192, 256) ..... 17
I.7. Conclusion ..... 25
**Chapitre II : Implémentation de l'algorithme AES sur un circuit FPGA** ..... 26
II.1. Introduction ..... 27
II.2. Les outils Software et Hardware de développement utilisés ..... 27

II.2.1. Présentation de l'environnement de développement "XILINX Integrated Software Environment (ISE) " ..... 27
II.2.2. La carte de développement Xtreme DSP Starter Kit-Spartan-3A DSP 1800A Edition ..... 29
II.3. Etapes pour l'implémentation d'une spécification HDL sur un FPGA ..... 31
II.3.1. Spécification HDL ..... 32
II.3.2. Vérification ..... 32
II.3.3. Implémentation ..... 32
II.3.4. Débogage du système ..... 33
II.4. Description de l'Architecture du circuit développé ..... 33
II.4.1. Circuit globale développé ..... 34
II.4.2. Architecture du module de cryptage ..... 35
II.4.2.1. Le sous-module Key_Gen ..... 37
II.4.2.2. Le sous-module AddRoundKey ..... 39
II.4.2.3 Le sous-module Shift_Row ..... 39
II.4.2.4. Le sous-module S_Box ..... 39
II.4.2.5. Le sous-module Mix_Columns ..... 40
II.4.2.6. Le sous-module Reg32_128 ..... 42
II.4.2.7. Le sous-module Enc_Controller ..... 42
II.4.3. Architecture du module de décryptage ..... 43
II.4.3.1. Le sous-module InvMix_Column ..... 45
II.4.3.2. Le sous-module InvS_Box ..... 47
II.4.3.3. Le sous-module InvShift_Row ..... 47
II.4.3.4. Le sous-module Dec_Controller ..... 48
II.4.4. Architecture du module de communication RS232 ..... 49
II.4.4.1. Le sous-module Rs232_rxd ..... 50
II.4.4.2. Le sous-module Bufer8_128 ..... 50
II.4.4.3. Le sous-module Rs232_128btxd ..... 51
II.5. Conclusion ..... 51
**Chapitre III : Résultats Pratiques** ..... 52
III.1. Introduction ..... 53
III.2. Simulation temporelle des différents modules de l'algorithme AES ..... 53
III.2.1. Module Key_Gen ..... 54

III.2.2. Module AddRoundKey ..... 54
III.2.3. Module S-Box ..... 55
III.2.4. Module InvS-Box ..... 55
III.2.5. Module ShiftRow ..... 56
III.2.6. Module InvShiftRow ..... 56
III.2.7. Module MixColumns ..... 57
III.2.8. Module InvMixColumns ..... 57
III.2.9. Module Data_Encryption ..... 58
III.2.10. Module Data_Decryption ..... 58
III.2.11. Simulation de l'architecture pipeline ..... 59
III.3. Résultats de synthèse et d'implémentation du circuit développé ..... 59
III.3.1. Rapport après la synthèse et l'implémentation ..... 59
III.3.2. Comparaison des résultats après l'implémentation du circuit développé avec d'autres circuits réalisés ..... 60
III.3.3. les schémas RTL générés après la synthèse ..... 60
III.4. Résultats de test pratique sur la carte FPGA ..... 63
III.5. Conclusion ..... 64
**Conclusion générale** ..... 65
**Bibliographie** ..... 66
**Annexe A** ..... 68
**Annexe B** ..... 73

# Liste des figures

Figure I.1 : Substitution et permutation ..... 7
Figure I.2 : Succession de rondes ..... 8
Figure I.3 : Chiffrement asymétrique ..... 8
Figure I.4 : Chiffrement symétrique ..... 9
Figure I.5 : Historique de chiffrements symétriques ..... 11
Figure I.6 : Différents participants au concours AES ..... 12
Figure I.7 : Distribution des jurys et Sécurité ..... 13
Figure I.8 : Efficacité matérielle ..... 13
Figure I.9 : Efficacité logicielle des algorithmes participants ..... 13
Figure I.10 : Résultats des votes ..... 14
Figure I.11: L'organigramme général des différentes étapes ..... 16
Figure I.12 : Particularité des transformations ..... 17
Figure I.13 : Schéma des étapes d'un seul tour ..... 18
Figure I.14 : Transformation d'un bloc à une table ..... 18
Figure I.15 : la transformation SubBytes ..... 19
Figure I.16 : la transformation ShiftRows ..... 20
Figure I.17 : la transformation MixColumns ..... 20
Figure I.18 : l'étape AddRoundKey ..... 21
Figure I.19 : Le schéma général d'extension d'une clé ..... 22
Figure II.1 : Le système de test ..... 27
Figure II.2 : Le navigateur des projets de ISE Design Suite 14.1 ..... 28
Figure II.3 : Vue externe de la plateforme XtremeDSP Starter Kit - Spartan-3A DSP 1800A Edition ..... 30
Figure II.4 : Les étapes d'implémentation d'un circuit sur un circuit logique programmable (XILINX) ..... 31
Figure II.5 : Schéma bloc du circuit développé ..... 35
Figure II.6 : Schéma bloc du module de cryptage ..... 36
Figure II.7 : Schéma bloc du module Key_Gen ..... 37
Figure II.8 : Schéma de génération d'une clé de ronde ..... 38
Figure II.9 : Schéma bloc du module S-Box ..... 40
Figure II.10 : Schéma bloc du module MixColumns ..... 40

Figure II.11 : Schéma bloc de la multiplication dans GF($2^8$) par 2 ..... 41
Figure II.12 : Schéma bloc de la multiplication par 1, 2 et 3 ..... 41
Figure II.13 : Schéma bloc du module de décryptage ..... 44
Figure II.14 : Schéma bloc du module InvMixColumns ..... 45
Figure II.15 : Schéma bloc de la multiplication dans GF($2^8$) par 9, 11, 13, 14 ..... 46
Figure II.16 : La liaison série RS-232 ..... 49
Figure III.1 : Simulation temporelle du module Key_Gen ..... 54
Figure III.2 : Simulation temporelle du module AddRoundKey ..... 54
Figure III.3 : Simulation temporelle du module S-Box ..... 55
Figure III.4 : Simulation temporelle du module InvS-Box ..... 55
Figure III.5 : Simulation temporelle du module ShiftRow ..... 56
Figure III.6 : Simulation temporelle du module InvShiftRow ..... 56
Figure III.7 : Simulation temporelle du module MixColumns ..... 57
Figure III.8 : Simulation temporelle du module InvMixColumns ..... 57
Figure III.9 : Simulation temporelle du module Data_Encryption ..... 58
Figure III.10 : Simulation temporelle du module Data_Décncryption ..... 58
Figure III.11 : Simulation de l'architecture pipelines. ..... 59
Figure III.12 : Les schémas RTL des modules Data_Encryption et Data_Decryption ..... 61
Figure III.13 : Le schéma RTL du circuit développé global ..... 62
Figure III.14 : Résultat de test pratique sur la carte ..... 64

# Liste des tableaux

Tableau I.1 : Combinaisons Clé-Bloc-Ronde ..... 16
Tableau I.2 : La table de substitution utilisé dans le SubBytes ..... 19
Tableau I.3 : La table Rcon utilisée dans l'extension des clés ..... 22
Tableau I.4 : La table de substitution utilisé dans le InvSubBytes ..... 23
Tableau II.1 : description des signaux Entrées/Sorties du module Data_Encryption ..... 36
Tableau II.2 : Table Rcon utilisée dans l'extension des clés ..... 37
Tableau II.3 : description des signaux Entrées/Sorties du module Key_Gen ..... 38
Tableau II.4 : description des signaux Entrées/Sorties du module AddRoundKey ..... 39
Tableau II.5 : description des signaux Entrées/Sorties du module ShiftRow ..... 39
Tableau II.6 : description des signaux Entrées/Sorties du module S_Box ..... 40
Tableau II.7 : description des signaux Entrées/Sorties du module MixColumns ..... 42
Tableau II.8 : description des signaux Entrées/Sorties du module Reg32_128 ..... 42
Tableau II.9 : La table d'ordonnancement des unités du module de cryptage ..... 43
Tableau II.10 : description des signaux Entrées/Sorties du module Enc_Controller ..... 43
Tableau II.11 : description des signaux Entrées/Sorties du module Data_Decryption ..... 44
Tableau II.12 : description des signaux Entrées/Sorties du module InvMix_Columns ..... 46
Tableau II.13 : description des signaux Entrées/Sorties du module InvS_Box ..... 47
Tableau II.14 : description des signaux Entrées/Sorties du module InvShift_Row ..... 47
Tableau II.15 : La table d'ordonnancement des unités du module de décryptage ..... 48
Tableau II.16 : description des signaux Entrées/Sorties du module Dec_Controller ..... 48
Tableau II.17 : description des signaux Entrées/Sorties du module Rs232_rxd ..... 50
Tableau II.18 : description des signaux Entrées/Sorties du module Bufer8_128 ..... 50
Tableau II.19 : description des signaux Entrées/Sorties du module Rs232_128btxd ..... 51
Tableau III.1: Les résultats de l'implémentation des différents modules ..... 60
Tableau III.2 : Comparaison du circuit développé avec d'autres circuits réalisés ..... 60
Tableau III.3 : La configuration des pins de la carte FPGA ..... 63
Tableau A.1 : Glossaire des termes et des acronymes ..... 69
Tableau A.2 : Paramètres, symboles et fonctions. ..... 71
Tableau B.1 : Exemple de génération des clés ..... 74
Tableau B.2 : Exemple de chiffrement ..... 76

# Introduction générale

En monde d'aujourd'hui plus de la communication est fait en utilisant des supports électroniques. La protection des données joue un rôle essentiel dans une telle communication. Par conséquent, il y a un besoin de protéger des données contre des attaques malveillantes.

La cryptographie est la science des codes secrets, permettant la confidentialité de communication par un canal peu sûr. Elle se protège contre les parties non autorisées en empêchant le changement non autorisé d'utilisation. D'une façon générale, elle emploie un système cryptographique pour transformer un texte clair en texte chiffré, en utilisant une clé.

La norme avancée de chiffrement AES (Advanced Encryption Standard), également connue sous le nom de Rijndael (J.Daeme et V. Rijmen), est une norme de chiffrement utilisée pour sécuriser l'information. AES a été édité par NIST (National Institute of Standards and Technology). L'AES est un algorithme de chiffrement par bloc qui a été analysé intensivement et est maintenant répandu largement. AES est un chiffrement par bloc symétrique qui est prévu pour remplacer le DES comme norme approuvée pour des larges applications **[1]**.

Le chiffrement et déchiffrement par bloc Rijnddael a été conçu par Dr. Joan Daemen et Dr. Vincent Rijmen et le nom de l'algorithme est une combinaison des noms de ses deux créateurs. Il prend un bloc d'entrée d'une certaine taille, habituellement 128 bits, et produit un bloc de sortie correspondant des mêmes tailles. La transformation exige une deuxième entrée, qui est la clé secrète. Il est important de savoir cette clé secrète **[1]**.

Le présent projet consiste à l'implémentation de l'algorithme de chiffrement et de déchiffrement symétrique AES sur un circuit FPGA. Dans ce travail, le chiffrement et le déchiffrement seront effectués avec une longueur principale de 128 bits, le bloc d'entrée et la clé secrète seront donnés pour le chiffrement, le bloc chiffré et la même clé secrète seront fournis au déchiffrement pour obtenir le bloc clair comme sortie. Toutes les transformations de chiffrement et de déchiffrement seront développées en utilisant le langage VHDL et vérifiées à l'aide des résultats de simulations.

Et finalement, le code en VHDL développé est synthétisé sur le circuit FPGA: Spartan-3A DSP 1800A Edition en utilisant l'outil de Xilinx ISE Design Suite 14.1.

Une approche méthodologique en trois parties a été appliquée :

La première partie fournit un bref aperçu sur la cryptographie, les différentes méthodes de chiffrement/déchiffrement, une description théorique des différents éléments de l'algorithme AES élaboré par l'Institut National des Standards et Technologies NIST.

La deuxième partie présente l'environnement de développement, la description de l'architecture matérielle de notre propre conception du circuit développé sur un circuit FPGA. La dernière partie est dotée aux résultats expérimentaux de simulation, synthèse, implémentation et tests pratiques du circuit développé sur la plate-forme FPGA.

# Chapitre I

# Description de l'algorithme de chiffrement AES

Sommaire

I.1. Introduction à la cryptographie ..... 4
I.1.1. Généralités sur la Cryptologie ..... 4
I.1.2. Petit Historique de la Cryptographie ..... 4
I.2. Principe de Chiffrement et Déchiffrement ..... 5
I.3. La Cryptanalyse ..... 6
I.4. Chiffrement par Blocs ..... 6
I.5. Deux Méthodes pour le Chiffrement ..... 8
I.5.1. Chiffrement Asymétrique ..... 8
I.5.2. Chiffrement Symétrique ..... 9
I.6. AES (Advanced Encryption Standard) ..... 10
I.6.1. Origine d'AES ..... 10
I.6.2. Domaines d'application de l'AES ..... 11
I.6.3. Les résultats d'un concours ..... 12
I.6.3.1. Avantages et inconvénients des participants aux concours ..... 14
I.6.3.2. Le choix : Rijndael ..... 15
I.6.4. Architecture de l'algorithme AES de Rijndael ..... 16
I.6.4.1. Propriété des transformations ..... 17
I.6.4.2. Description de l'architecture (AES-128, 192, 256) ..... 17
I.7. Conclusion ..... 25

# I.1. Introduction à la cryptographie

## I.1.1. Généralités sur la Cryptologie [2]

La cryptologie, "la science du secret", est composée de deux disciplines complémentaires : la cryptographie et la cryptanalyse. La cryptographie est l'art de cacher l'information. Elle répond en particulier aux besoins suivants :

- ***Confidentialité*** : garantir que les données échangées ne sont aperçues qu'aux personnes voulues.
- ***Intégrité*** : garantir que les données échangées ne sont pas modifiées.
- ***Authenticité*** : prouver l'identité d'une personne ou l'origine d'un document.
- ***Signature*** : permettre à une personne de "signer" informatiquement un document sans qu'elle puisse le renier par la suite.

Lorsqu'un système cryptographique est utilisé, certains vont tenter de l'attaquer pour mettre en défaut sa sécurité. L'ensemble des attaques sur les crypto-systèmes est la cryptanalyse.

## I.1.2. Historique de la Cryptographie

L'histoire de la cryptographie est indissociable de celle de l'écriture. Les premières traces de son utilisation datent de 2000 ans avant Jésus-Christ en Égypte. Durant des siècles, la cryptographie se fonda sur deux grands principes :

- La permutation : on modifie dans le texte clair l'ordre des lettres.
- La substitution : dans un alphabet, on remplace une lettre par une autre (substitution

Mono-alphabétique ou poly-alphabétique).

L'alphabet de César reste le plus fameux exemple de ce type de chiffrement. Il consiste en un décalage de 3 positions des lettres de l'alphabet.

Ce n'est qu'en 1949 que Shannon, dans le cadre de la théorie de l'information, formalise les deux concepts utiles à la construction de crypto-systèmes qu'il nomma respectivement diffusion (chaque bit du texte clair doit influencer le plus grand nombre possible de bits du chiffré) et confusion (la structure liant le chiffré et le clair doit être très complexe). L'apparition et le développement des ordinateurs et des réseaux de télécommunications au

cours des dernières décennies ont sorti la cryptographie de son cadre strictement diplomatique et militaire.

La diffusion d'information à l'échelle de la planète a rendu nécessaire la création de systèmes cryptographiques communs réputés très sûrs, car la puissance de calcul des ordinateurs personnels permet aujourd'hui à chacun de s'essayer à la cryptanalyse. C'est dans ce contexte qu'est né en 1975 l'un des premiers grands algorithmes de la cryptographie à clé secrète : le DES "Data Encryption Standard", très influencé par la théorie de l'information et choisi comme standard de chiffrement par le gouvernement américain en 1977. Depuis, la recherche en cryptographie n'a cessé d'avancer tant en clé publique qu'en clé secrète. Citons ici le choix, en octobre 2000, du nouveau standard de chiffrement américain pour le XXIème siècle : l'AES "Advanced Encryption Standard", en remplacement du DES **[2]**.

## I.2. Principe de Chiffrement et Déchiffrement [2]

Le premier objectif de la cryptographie est d'assurer la confidentialité des données. Supposons que deux individus veuillent communiquer entre eux sur un canal non sûr sans que personne d'autre ne puisse lire le contenu du message ; ces deux individus, que nous appellerons Alice et Bob comme le veut la tradition cryptographique, doivent pour se protéger des éventuelles attaques du dénommé Oscar utiliser un système cryptographique. Si Alice veut envoyer un message à Bob, elle chiffre le texte du message grâce à une fonction de chiffrement $\boldsymbol{E}$ utilisant une clé $\boldsymbol{k}$. Elle envoie alors le message chiffré que seul Bob sera capable de déchiffrer à l'aide d'une fonction de déchiffrement $\boldsymbol{D}$ et d'une donnée supplémentaire (la clé par exemple). On peut alors formaliser la notion de fonction de chiffrement et de déchiffrement de la façon suivante :

**Définition**

Soit $\boldsymbol{M}$ l'ensemble des textes clairs, $\boldsymbol{C}$ l'ensemble des textes chiffrés, $\boldsymbol{K}$ l'ensemble des clés. Une fonction de chiffrement $\boldsymbol{E}$ associe à tout couple $(\boldsymbol{k}, \boldsymbol{m})$ de $\boldsymbol{K} \times \boldsymbol{M}$ un élément $\boldsymbol{c} = \boldsymbol{E}(\boldsymbol{k}, \boldsymbol{m})$ de $\boldsymbol{C}$. La fonction de déchiffrement correspondante, paramétrée par la clé de déchiffrement $\boldsymbol{d}$, associe alors au couple $(\boldsymbol{d}, \boldsymbol{c})$ du $\boldsymbol{K} \times \boldsymbol{C}$ le message $\boldsymbol{m}$ de $\boldsymbol{M}$, elle vérifie :

$$\boldsymbol{D}(\boldsymbol{d}, \boldsymbol{E}(\boldsymbol{k}, \boldsymbol{m})) = \boldsymbol{m} \quad \text{(I.1)}$$

## I.3. La Cryptanalyse

Jusqu'à une époque assez récente, la sécurité d'un chiffrement reposait tout autant sur le non divulgation de l'algorithme utilisé que sur la clé. Ce n'est plus le cas aujourd'hui et on attend d'un algorithme, conformément au principe de Kerckhoffs, qu'il reste solide lorsque l'attaquant en connaît la spécification et ignore seulement la clé utilisée. La cryptanalyse d'un système peut être alors soit partielle (l'attaquant découvre le texte clair correspondant à un ou plusieurs messages chiffrés interceptés), soit totale (l'attaquant peut déchiffrer tous les messages, par exemple en trouvant la clé). Il existe plusieurs types d'attaques selon les moyens dont dispose l'attaquant :

- Attaques à chiffré connu : l'attaquant a seulement accès à des messages chiffrés.
- Attaques à clair connu : l'attaquant dispose d'un ou plusieurs messages clairs et des chiffrés correspondants.
- Attaques à clair choisi : l'attaquant choisit des clairs et peut obtenir les chiffrés correspondants.
- Attaques à chiffré choisi : l'attaquant peut déchiffrer les messages de son choix.

L'attaque la plus simple et la plus brutale est la recherche exhaustive. L'attaquant teste l'ensemble des clés possibles sur un cryptogramme donné dont il est supposé connaître au moins partiellement le clair ; il a découvert la bonne clé lorsque le déchiffrement redonne le clair attendu. La complexité d'une recherche exhaustive sur un algorithme de chiffrement par blocs dont la taille des clés est n bits est de l'ordre de $2^n$ chiffrements. Il est donc nécessaire lorsque l'on souhaite créer un algorithme de chiffrement de prendre des clés suffisamment longues pour se prémunir contre ce type d'attaque. Malheureusement, ce n'est pas la seule condition à vérifier. C'est pourquoi, la construction de tels algorithmes doit s'appuyer sur des problèmes mathématiques difficiles **[2]**.

## I.4. Chiffrement par Blocs [3]

On oppose souvent chiffrement à flot et chiffrement par blocs et pour cause : le premier utilise une suite chiffrant, le deuxième une fonction. Le chiffrement par blocs consiste à découper en blocs de longueur fixe $n$ le message $m$ à chiffrer, puis à traiter chaque bloc séparément. Il faut que $n$ soit suffisamment grand pour éviter les attaques par dictionnaire.

L'idée générale du chiffrement par blocs est la suivante :

1. Remplacer les caractères par un code binaire
2. Découper cette chaîne en blocs de longueur donnée
3. Chiffrer un bloc en "l'additionnant" bit par bit à une clef.
4. Déplacer certains bits du bloc.
5. Recommencer éventuellement un certain nombre de fois l'opération 3.
6. Passer au bloc suivant et retourner au point 3 jusqu'à ce que tout le message soit chiffré.

On distingue trois catégories de chiffrement par bloc :

- Chiffrement par substitution : Les substitutions consistent à remplacer des symboles ou des groupes de symboles par d'autres symboles ou groupes de symboles dans le but de créer de la confusion Figure I.1.
- Chiffrement par transposition : Les transpositions consistent à mélanger les symboles ou les groupes de symboles d'un message clair suivant des règles prédéfinies pour créer de la diffusion. Ces règles sont déterminées par la clé de chiffrement. Une suite de transpositions forme une permutation Figure I.1.
- Chiffrement par produit : C'est la combinaison des deux. Le chiffrement par substitution ou par transposition ne fournit pas un haut niveau de sécurité, mais en combinant ces deux transformations, on peut obtenir un chiffrement plus robuste. La plupart des algorithmes à clés symétriques utilisent le chiffrement par produit. On dit qu'un "round" est complété lorsque les deux transformations ont été faites une fois (substitution et transposition).

Ces successions des rondes portent également le nom de réseaux S-P de Shannon **[3]**.

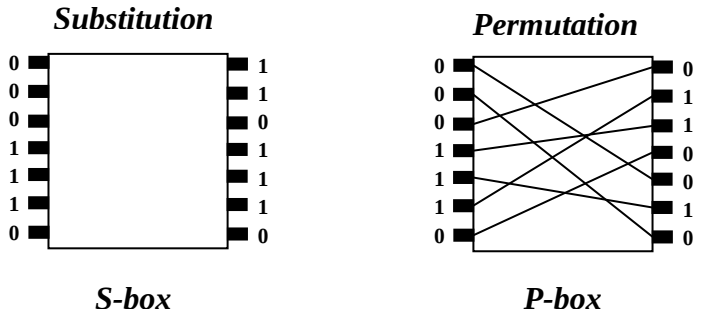



***Figure I.1 :*** *Substitution et permutation*

Un chiffrement par blocs composés de couches (layers) ou "rondes" caractérisés par des opérations de substitution et permutation de chaque ronde **[3]**.

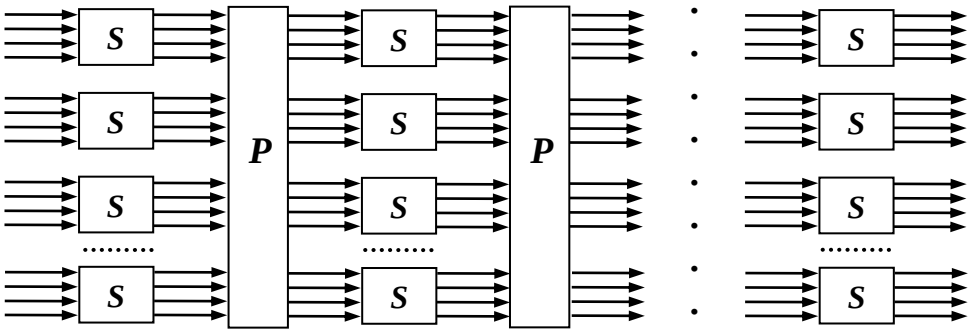



***Figure I.2 :*** *Succession de rondes*

## I.5. Deux Méthodes pour le Chiffrement

Jusqu'en 1976, la seule méthode connue pour échanger de l'information était la cryptographie à clé secrète ou symétrique ; pour que deux personnes puissent échanger un message chiffré, elles devaient partager un même secret connu d'elles seules. En 1976, est apparue une nouvelle forme de chiffrement dite à clé publique (ou asymétrique), où les protagonistes ne sont plus tenus de partager une même clé.

### I.5.1. Chiffrement Asymétrique

**Diffie et Hellmen** ont dégagé vers 1976 la notion de code à clef publique ou code asymétrique fondé sur la notion de fonction à sens unique.

Si Alice veut envoyer un message confidentiel à Bob, elle se procure (par exemple dans un annuaire public) la clé publique $K_p$ de Bob et chiffre le message $m$ à l'aide d'un algorithme de chiffrement asymétrique et de la clé $K_p$. Elle envoie ensuite sur le canal non sûr le message chiffré $c$ que seul Bob peut déchiffrer grâce à la clé secrète correspondante $K_s$ de Bob, (connue de lui seul) Figure I.3. Cette méthode de chiffrement repose sur la difficulté du problème de la factorisation de grands nombres.

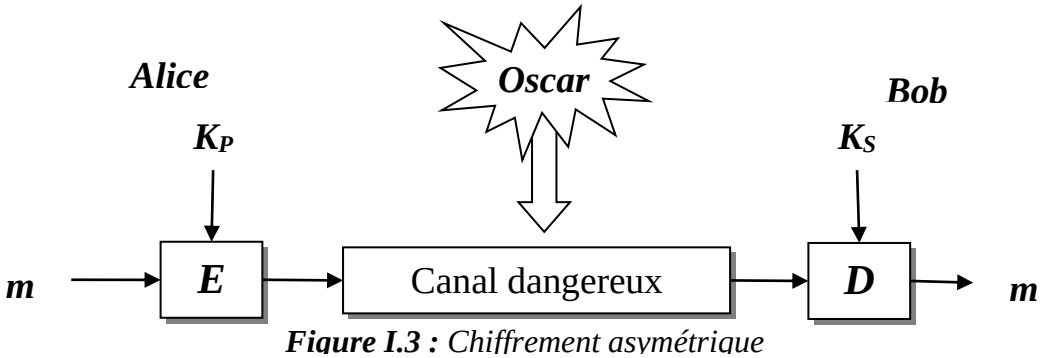



***Figure I.3 :*** *Chiffrement asymétrique*

Les méthodes asymétriques les plus utilisés :

- ***RSA*** basé sur la difficulté calculatoire de la factorisation des grands entiers.

- ***El Gamal*** basé sur la difficulté calculatoire de calculer le logarithme discret dans un corps fini.

- ***Menezes-Vanstonne*** basé sur le logarithme associé au groupe des points d'une courbe elliptique sur un corps fini. C'est une modification d'autres crypto-systèmes, comme El Gamal. Sa sureté peut être mieux analysée. La longueur de sa clé est bien inférieure à celle des systèmes RSA et El Gamal pour une sureté équivalente. Ils offrent des fonctionnalités supplémentaires comme la possibilité de monter un crypto-système basé sur l'identité.

### I.5.2. Chiffrement Symétrique [2]

Si Alice et Bob décident d'utiliser pour communiquer un principe de chiffrement symétrique, ils doivent partager la même clé ***k*** connue d'eux seuls qui est identique pour le chiffrement et le déchiffrement Figure I.4:

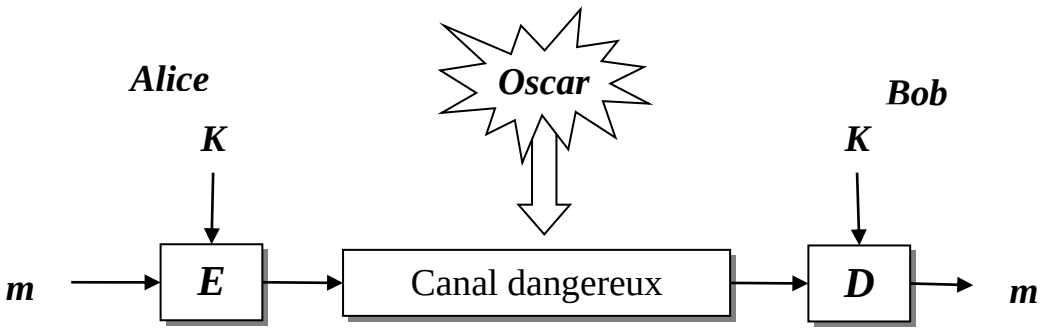



***Figure I.4 :*** *Chiffrement symétrique*

La cryptographie symétrique utilise la même clé pour les processus de chiffrement et de déchiffrement ; cette clé est le plus souvent appelée "secrète" (en opposition à "privée") car toute la sécurité de l'ensemble est directement liée au fait que cette clé n'est connue que par l'expéditeur et le destinataire. La cryptographie symétrique est très utilisée et se caractérise par une grande rapidité (opérations simples, chiffrement à la volée) et par des implémentations aussi bien software que hardware ce qui accélère nettement les débits et autorise son utilisation massive **[3]**.

Les méthodes asymétriques les plus utilisés **[3]**.

- **Le système DES (Data Encryption Standard) :** le D.E.S. est un crypto-système agissant par blocs. Cela signifie que D.E.S. ne chiffre pas les données à la volée quand les caractères arrivent, mais il découpe virtuellement le texte clair en blocs de 64 bits qu'il

chiffre séparément, puis il les concatène. Un bloc de 64 bits du texte clair entre par un coté de l'algorithme et un bloc de 64 bits de texte chiffré sort de l'autre coté.

- **2DES :** suite aux failles du DES, quelques modifications ont été apportées, mais pas toujours avec succès. Ce fut notamment le cas avec le 2DES.

- **3DES :** Il est très robuste contre toutes les attaques faisables connues. Cependant, il est beaucoup plus lent que le DES car on triple les opérations.

- **AES (Advanced Encrytion Standard) [4] :** de J.Daeme et V.Rijmen, basé sur le même principe avec une clé plus longue (128, 192 ou 256)bits, plus structuré et avec des fonctionnalités plus étendues. AES a été retenu en 2000 après un appel d'offre international**.**

## I.6. AES (Advanced Encryption Standard)

### I.6.1. Origine d'AES [5]

Le 2 janvier 1997, l'Institut National des Standards et Technologies du Département du Commerce des Etats-Unis d'Amérique annonce le développement d'une norme nationale pour le traitement de l'information nommée Advanced Encryption Standard, ou Norme de Chiffrement Avancé**.**

Apparu fin 1976, le DES (Data Encryption Standard) a alors près de vingt ans d'âge. Le NIST, lors de la dernière révision de sécurité de cet algorithme, en 1993, insère la note suivante dans le dossier.

A la prochaine révision (1998), l'algorithme spécifié par ce standard aura plus de vingt ans. Le NIST se doit de considérer une alternative apportant un niveau supérieur de sécurité. Cette alternative devra être procédée en remplacement de ce standard lors de sa prochaine révision.

Le NIST, conscient qu'un tel changement ne pourra s'opérer en une fois, annonce alors la fin du DES en decrescendo, au fur et à mesure que son remplaçant, l'AES, prendra place au sein de la communauté cryptographique et dans l'industrie.

L'organisme sollicite alors l'opinion publique, les manufacturiers, les organismes de standardisation, ainsi que les utilisateurs gouvernementaux afin d'obtenir leur point de vue dans le développement d'AES. Celui-ci devra s'orienter autour de quelques axes définis préalablement. L'algorithme choisi devra **[5]**:

- être public, utilisant un chiffrement symétrique par bloc ;
- Avoir une clef dont la longueur pourra être incrémentée autant que nécessaire ;
- Garder la capacité d'être implémenté de manière matérielle ainsi que logicielle ;
- Nécessairement être disponible librement, selon les termes inscrits dans les licences de l'American National Standards Institute (ANSI).

Tout algorithme présenté répondant à ces exigences sera alors jugé sur les critères suivant :

- La sécurité, comme par exemple l'effort effectue sur la cryptanalyse,
- l'efficacité informatique,
- l'utilisation de la mémoire,
- l'adéquation matérielle et logicielle, la simplicité, la flexibilité et le respect des licences.

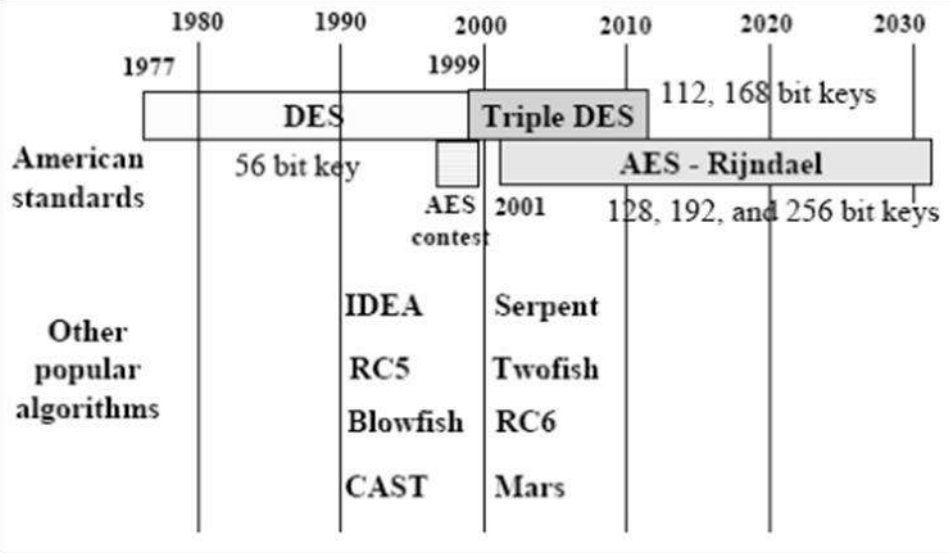



***Figure I.5:*** *Historique de chiffrements symétriques*

### I.6.2. Domaines d'application de l'AES

- L'AES peut être employé par des départements fédéraux et des agences quand une agence détermine que les informations (non secrètes) sensibles exigent la protection cryptographique.
- Commutateurs (switches) à grande vitesse d'ATM/Ethernet/Fiber-Channel.
- Fixer le système de téléconférences visuel.
- Routeurs et serveurs d'accès à distance.

- En outre, L'AES peut être adopté et employé par des organismes non-Fédéraux de gouvernement. Une telle utilisation est encouragée quand elle fournit la sécurité désirée pour des organismes commerciaux et privés.

### I.6.3. Les résultats d'un concours [3]

La progression de la puissance des ordinateurs a causé la mort du DES. Ce dernier est en voie de disparition, remplacé par AES (Advanced Encryption Standard). Le Triple DES demeure une norme acceptée pour les documents gouvernementaux aux U.S.A. Pour l'instant, il n'y a pas de projet ou d'obligation de ré-chiffrer les documents existants.

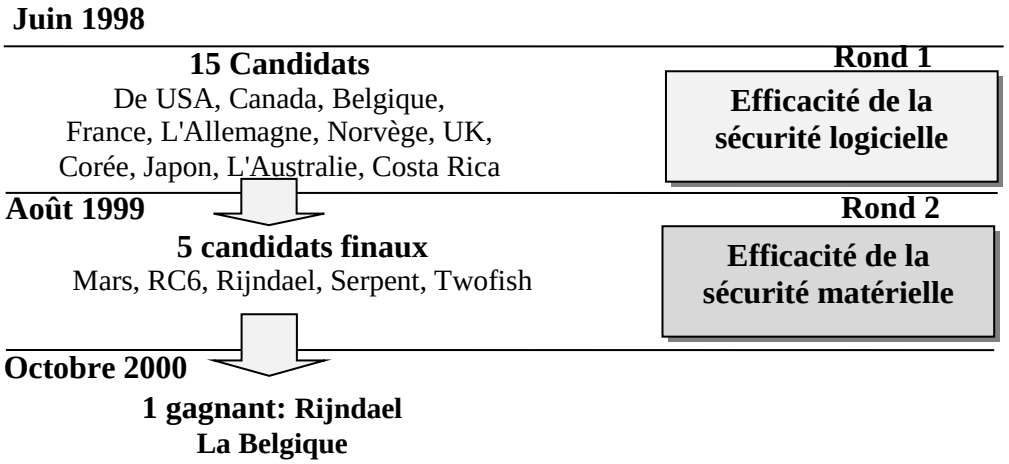



***Figure I.6:*** *Différents participants au concours AES*

**Cahier des charges [3]**

Au second tour du concours, les jurys devaient juger différents critères :

- ✓ La sécurité générale,
- ✓ Le coût en termes de calculs (rapidité),
- ✓ La simplicité de l'algorithme et ses facilités d'implémentation,
- ✓ Une lecture facile de l'algorithme, puisqu'il est destiné à être rendu public,
- ✓ La résistance aux attaques connues,
- ✓ Flexibilité - Portabilité : l'algorithme devant remplacer le DES, il est destiné à servir aussi bien dans les cartes à puces, aux processeurs 8 bits peu puissants, que dans des processeurs spécialisés pour chiffrer des milliers de télécommunications à la volée.
- ✓ Techniquement, le chiffrement doit se faire par blocs de 128 bits, les clés comportant 128, 192 ou 256 bits.

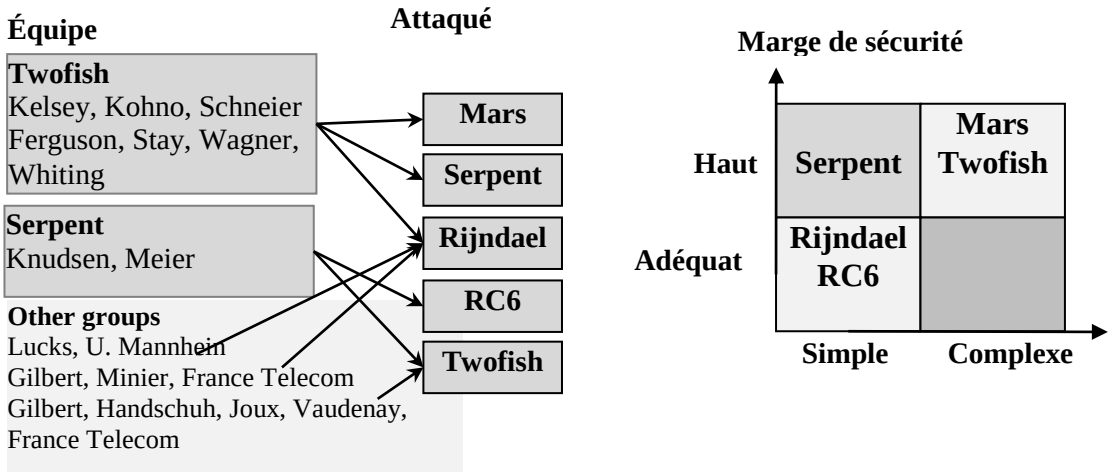



***Figure I.7 :*** *Distribution des jurys et Sécurité*

Au niveau du chiffrement/déchiffrement, les résultats varient assez fortement. Cependant, Serpent reste le moins bon pour la majorité des plateformes. Rijndael et RC6 étant les meilleurs. Au niveau de la gestion des clés, Twofish et Serpent sont les pires **[3]**.

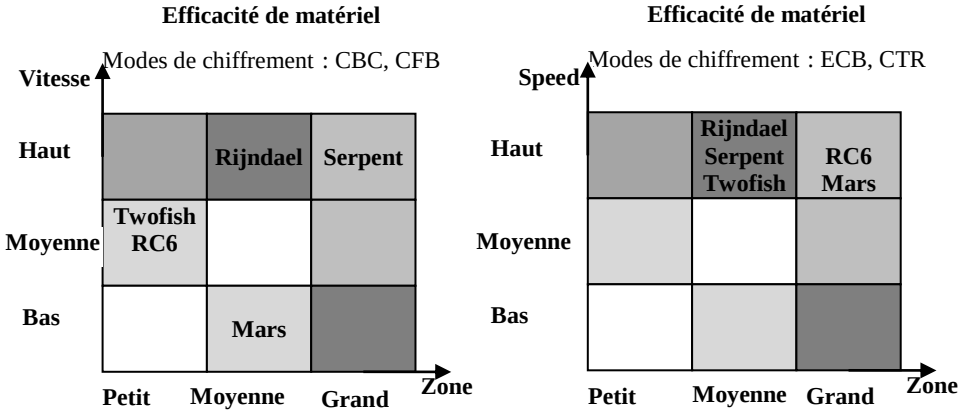



***Figure I.8 :*** *Efficacité matérielle*

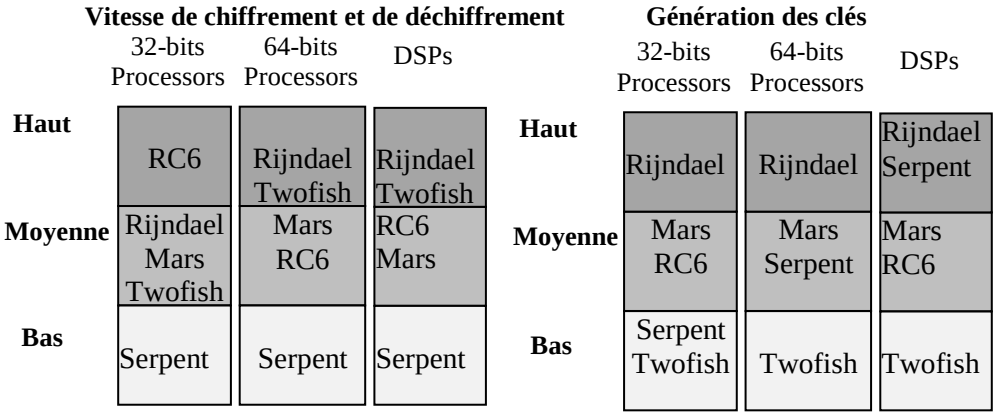



***Figure I.9 :*** *Efficacité logicielle des algorithmes participants*

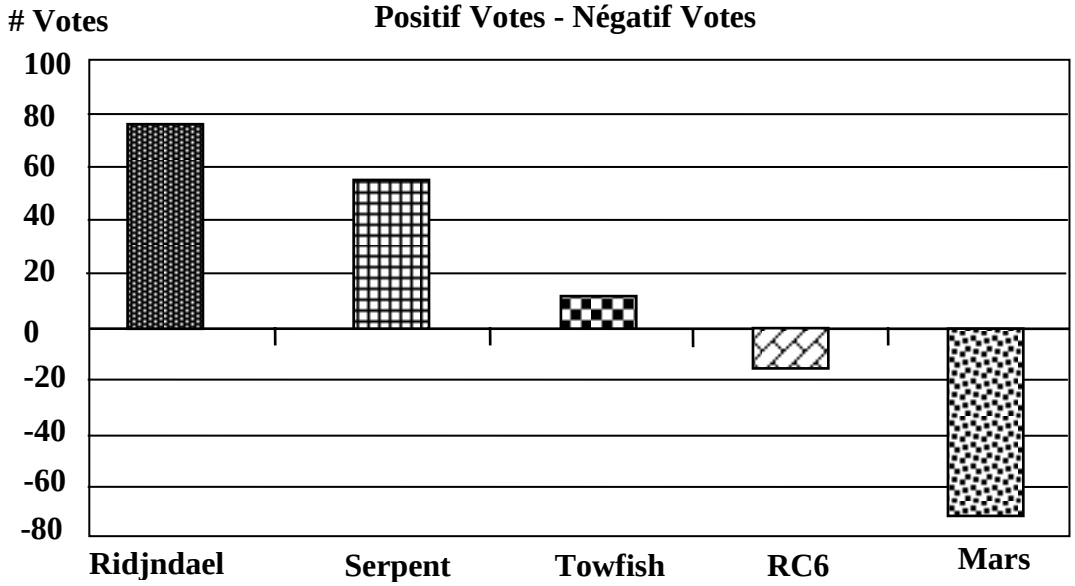



***Figure I.10 :*** *Résultats des votes*

### I.6.3.1. Avantages et inconvénients des participants aux concours [3]

**Rijndael – Rijmen, Daemen**

**Avantage :**

- ✓ Le plus rapide en implantation matérielle
- ✓ Proche des plus rapides en implantation logicielle
- ✓ Très grande flexibilité

**Inconvénients :**

- ☞ Marge de sécurité

**Serpent - Anderson, Biham, Knudsen**

**Avantage :**

- ✓ Large marge de sécurité
- ✓ Construction conservatrice
- ✓ Très rapide en implantation matérielle
- ✓ Réputation crypte-analytique de ses auteurs (Anderson, Biham, Knudsen)

**Inconvénients :**

- ☞ Lent en implantation logicielle
- ☞ Flexibilité modérée

**Twofish - Schneier, Wagner**

**Avantage :**

- ✓ Bonne marge de sécurité
- ✓ Rapide pour le chiffrement/déchiffrement en implantation logicielle
- ✓ Américain
- ✓ Beaucoup de publicité

**Inconvénients :**

- ☞ Modérément rapide en implantation matérielle
- ☞ Configuration lente de la clé en implantation logicielle
- ☞ Flexibilité modérée

### I.6.3.2. Le choix : Rijndael

A la suite de nombreux tests, c'est finalement Rijndael qui a remporté la médaille, et est ainsi devenu le remplaçant officiel du DES **[3]**.

Il possède les propriétés suivantes **[3]**:

- ✓ Plusieurs longueurs de clef et de bloc sont possibles : 128, 192, ou 256 bits ;
- ✓ Le nombre de cycles (ou "rondes") varie en fonction de la longueur des blocs et des clés (de 10 12 ou 14) ;
- ✓ La structure générale ne comprend qu'une série de transformations/ permutations/ sélections ;
- ✓ Il est beaucoup plus performant que le DES ;
- ✓ Il est facilement adaptable à des processeurs de 8 ou de 64 bits ;
- ✓ Le parallélisme peut être implémenté

### I.6.4. Architecture de l'algorithme AES de Rijndael

L'algorithme peut être employé avec les trois longueurs principales différentes indiquées ci-dessus, et peuvent désigné sous le nom de "AES-128", de "AES-192", et de "AES-256".pour chacun la taille de bloc (données) d'entrée et de sortie est toujours de 128 bits.

Pour les trois cas de l'algorithme, la longueur de clé et le nombre de rondes (tours) sont définit dans le tableau I.1 :

***Tableau I.1:*** *Combinaisons Clé-Bloc-Ronde*

| | Longueur de Clé (NK : Mot 32bits) | La taille de bloc (Nb : Mot 32bits) | Nombre de rondes (tour) (Nr) |
|---|---|---|---|
| AES-128 | 4 | 4 | 10 |
| AES-192 | 6 | 4 | 12 |
| AES-256 | 8 | 4 | 14 |

L'ordonnancement des étapes est illustré dans la figure I.11 **[3]**:

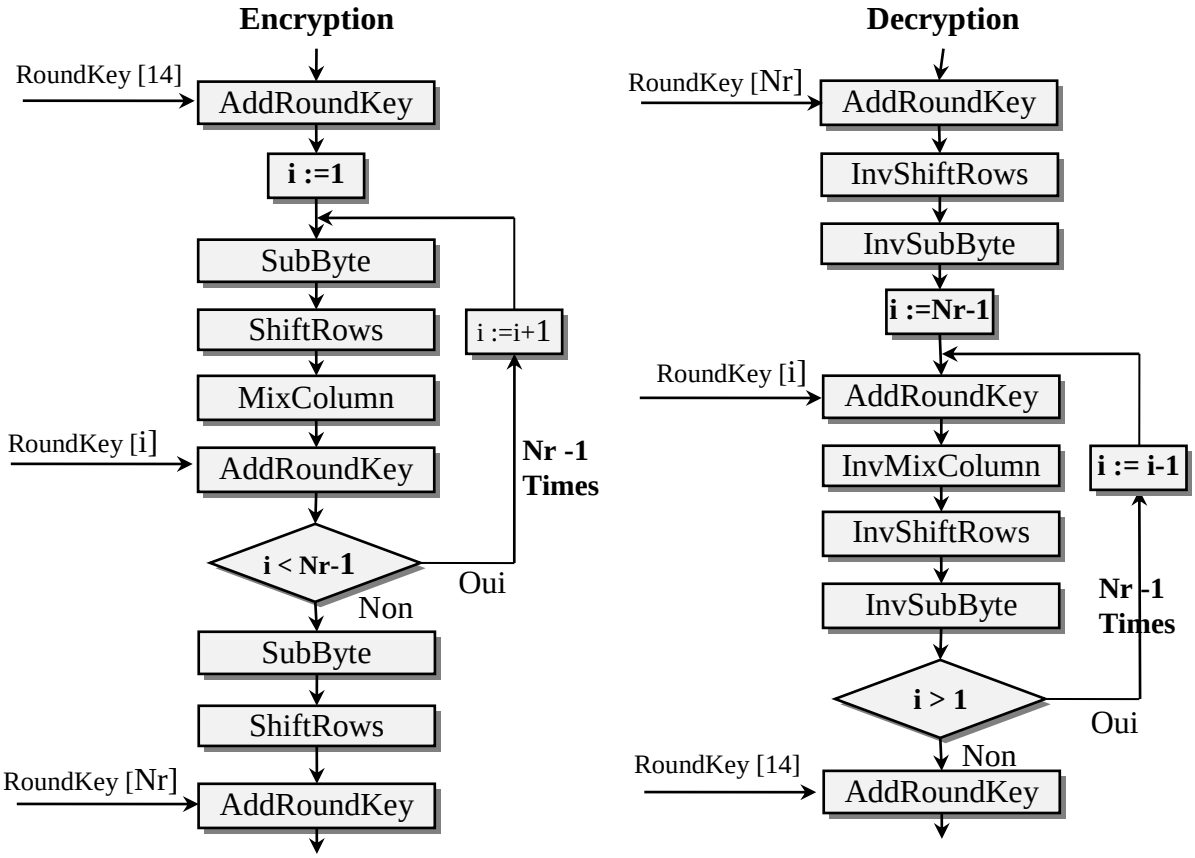



***Figure I.11:*** *L'organigramme général des différentes étapes*

### I.6.4.1. Propriété des transformations

Le Rijndael a été conçu de manière à ce que les étapes *SubBytes* et *ShiftRows* soient interchangeables. La même remarque reste vraie dans le cas du déchiffrement **[3]**.

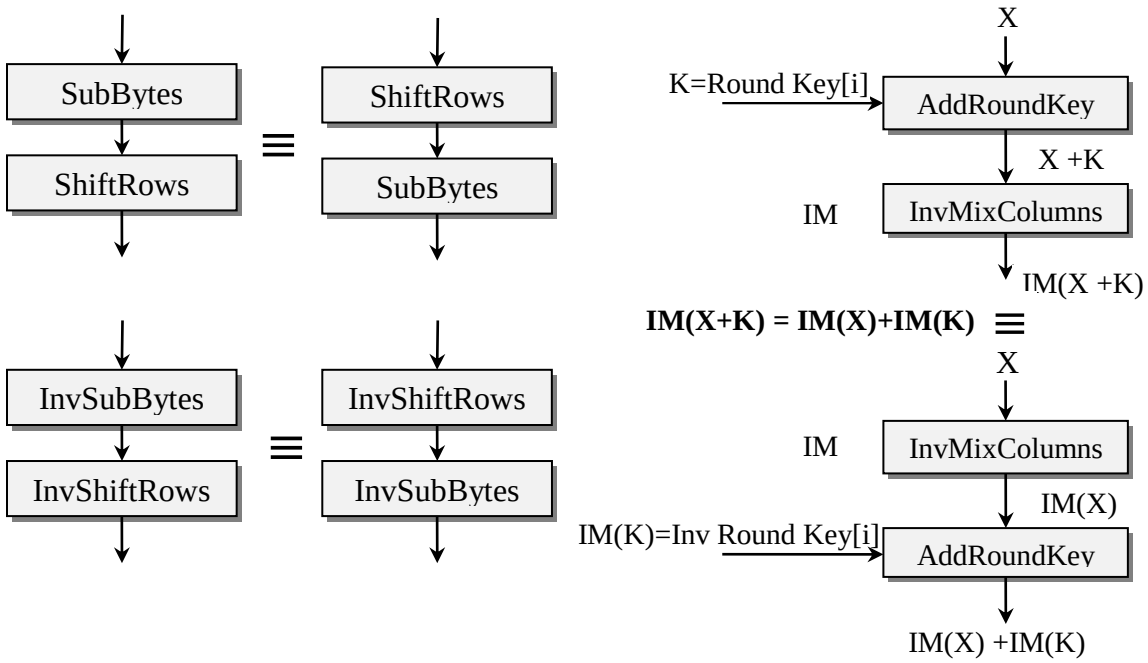



***Figure I.12 :*** *Particularité des transformations*

### I.6.4.2. Description de l'architecture (AES-128, 192, 256)

Le chiffrement et le déchiffrement de Rijndael se fait par **Nr** tours (10,12 ou 14) pour un bloc (message) de taille 128 bits et une clé de tour à une longueur (128,192 ou 256) bits, à chaque ronde, quatre transformations sont appliquées **[3]**:

1. Substitution d'octets dans le tableau de message.
2. Décalage des rangées dans le tableau de message.
3. Mélange des colonnes dans le tableau de message (sauf à la dernière ronde)
4. Addition d'une "clef de ronde" qui varie à chaque ronde

Le schéma d'un seul tour est illustré dans la figure I.13 :

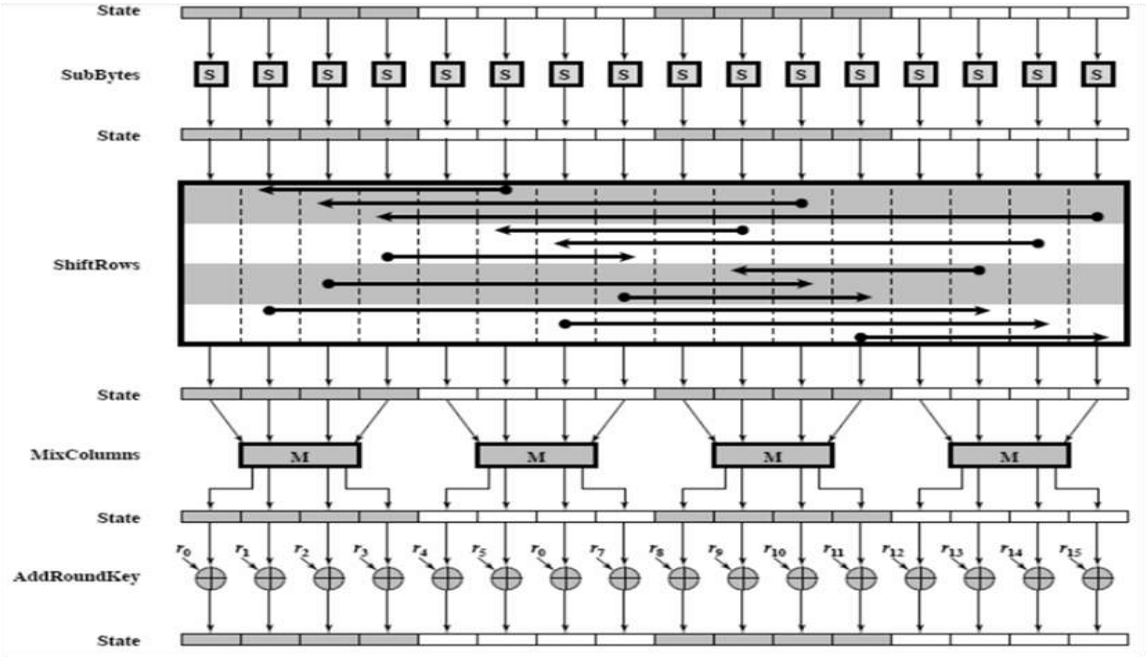



***Figure I.13:*** *Schéma des étapes d'un seul tour*

Pour simplifier les transformations et augmenter la vitesse de chiffrement/déchiffrement Le message et la clé sont conservés sous forme de tables comme la figure I.14 (exemple de 128 bits). Le nombre de colonnes dépend de taille de bloc et de la clé (pour clé de 192, 256).

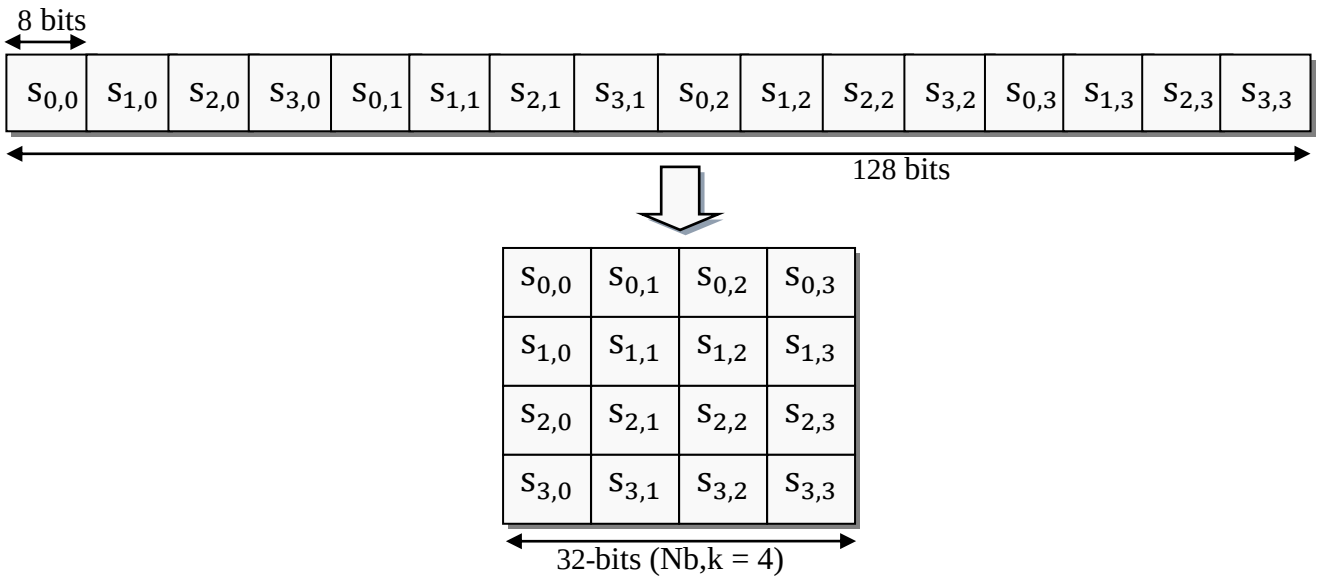



***Figure I.14:*** *Transformation d'un bloc à une table*

### I.6.4.2.1. Chiffrement [6]

#### A. La substitution (S-Box /SubBytes)

La transformation de *SubBytes* (Figure I.15) est une substitution non linéaire d'un bloc de 8-bits (byte) qui fonctionne indépendamment sur chaque byte de bloc en utilisant une table de substitution (boîte de substitution). Cette boîte de substitution (*Tableau I.2*), qui est inversible, une seule boîte est suffisante pour toute la phase de chiffrement.

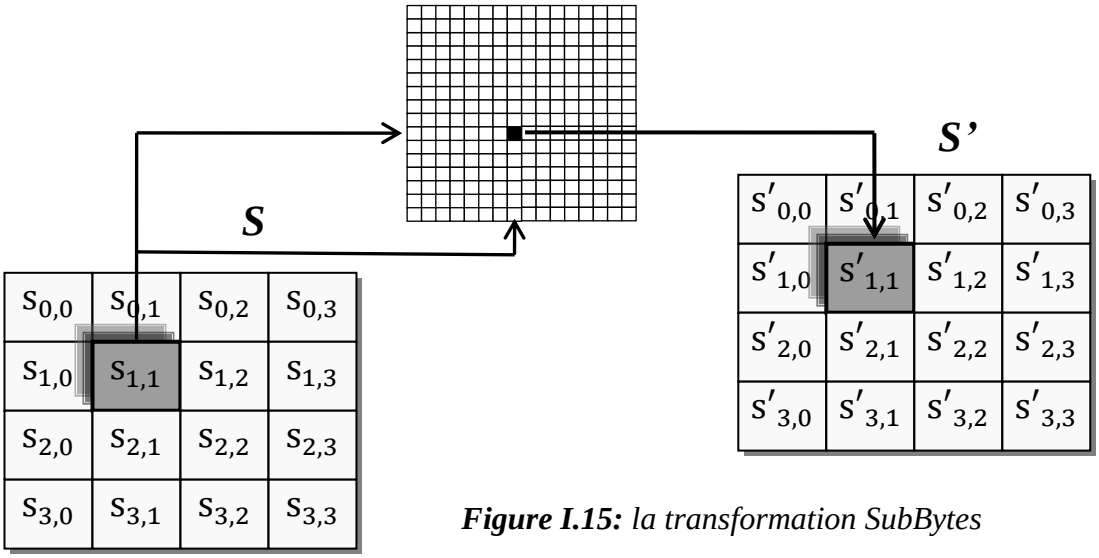



***Figure I.15:*** *la transformation SubBytes*

La table de substitution utilisé dans le *SubBytes* est représenté en Hexadécimal, voir le tableau I.2. Par exemple pour $s_{1,1} = \{53\}$, donc leur substitution est déterminée par l'intersection de la ligne d'indice '5' et la colonne d'indice '3', le résultat de la substitution obtenu est $s'_{1,1} = \{ed\}$

***Tableau I.2:*** *La table de substitution utilisé dans le SubBytes*

| x \ y | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | a | b | c | d | e | f |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 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 |

### B. Le décalage de rangées (*ShiftRows*)

Dans la transformation de *ShiftRows* (Figure I.16), les bytes dans les trois dernières lignes de la table (message) sont cycliquement décalés à gauche. La première ligne, r=0, n'est pas décalée.

Spécifiquement, la transformation de *ShiftRows* procède comme suit :

$$\boldsymbol{s'_{r,c}} = \boldsymbol{s_{r,(c-r)mod\ Nb}}, \qquad avec\ (\mathbf{0} \le r \le 3 \ \ et\ \ \mathbf{0} \le \boldsymbol{c} < Nb)$$

Où ***Nb*** = 4 pour AES-128

La figure I.16 illustre la transformation *ShiftRows* :

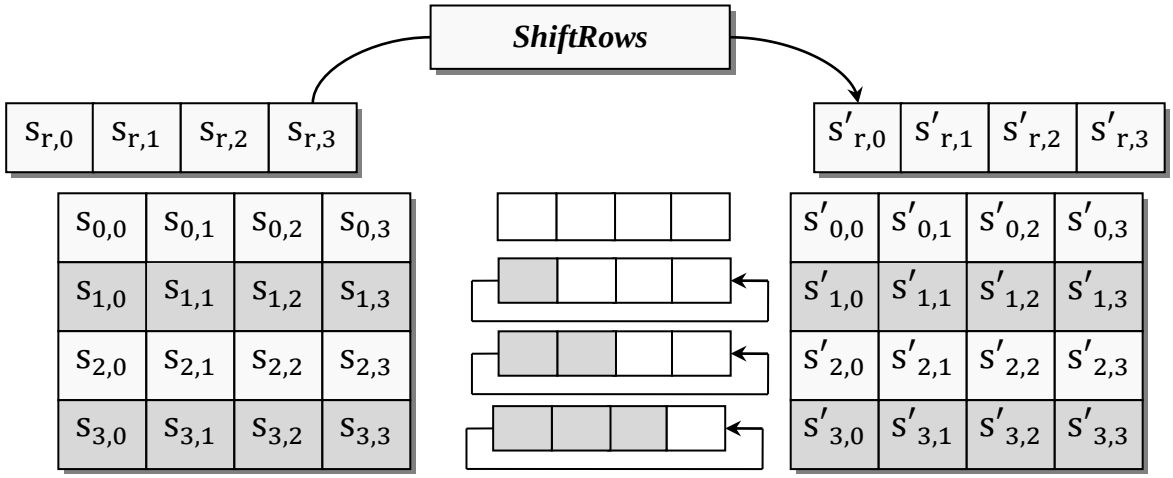



***Figure I.16:*** *la transformation ShiftRows*

### C. Mélange des colonnes (*MixColumns*)

Une différence sur 1 byte d'entrée se propage sur les 4 bytes de sortie. On a donc encore une étape de diffusion. La matrice utilisée est définie par Rijndael. Elle contiendra toujours ces valeurs Figure I.17.

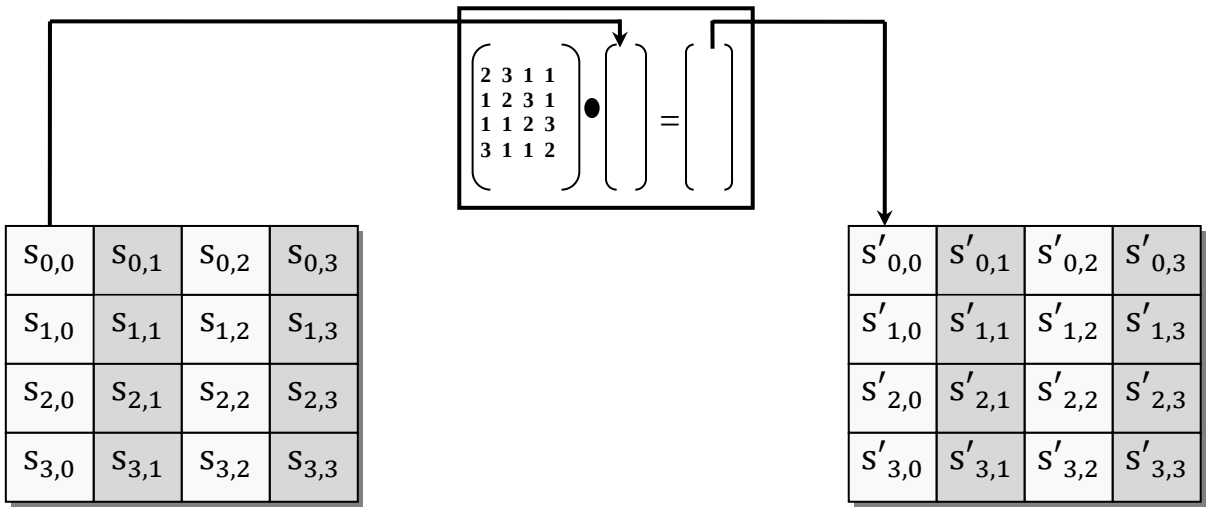



***Figure I.17:*** *la transformation MixColumns*

En raison de cette multiplication, les quatre bytes dans une colonne sont remplacés par ce qui suit :

$$\begin{pmatrix} S'_{0,c} \\ S'_{1,c} \\ S'_{2,c} \\ S'_{3,c} \end{pmatrix} = \begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix} \bullet \begin{pmatrix} S_{0,c} \\ S_{1,c} \\ S_{2,c} \\ S_{3,c} \end{pmatrix} \quad \text{Pour } 0 \le c \le Nb \text{.....(I.2)}$$

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

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

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

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

**Remarque :** la multiplication polynomiale (●) est définie sur le champ fini GF($2^8$), qui est spécifiée dans l'annexe (A.3).

### D. Addition d'une clé de ronde (*AddRoundKey*)

Dans la transformation *AddRoundKey*, une clé de ronde (pour chaque ronde il y'a une clé différente) est ajoutée au message par une opération Ou-Exclusif(XOR) au niveau du bit Figure I.18:

$$S'_{r,c} = S_{r,c} \oplus k_{r,c} \; avec \; 0 \le r \le 3 \; et \; 0 \le c < Nb.$$

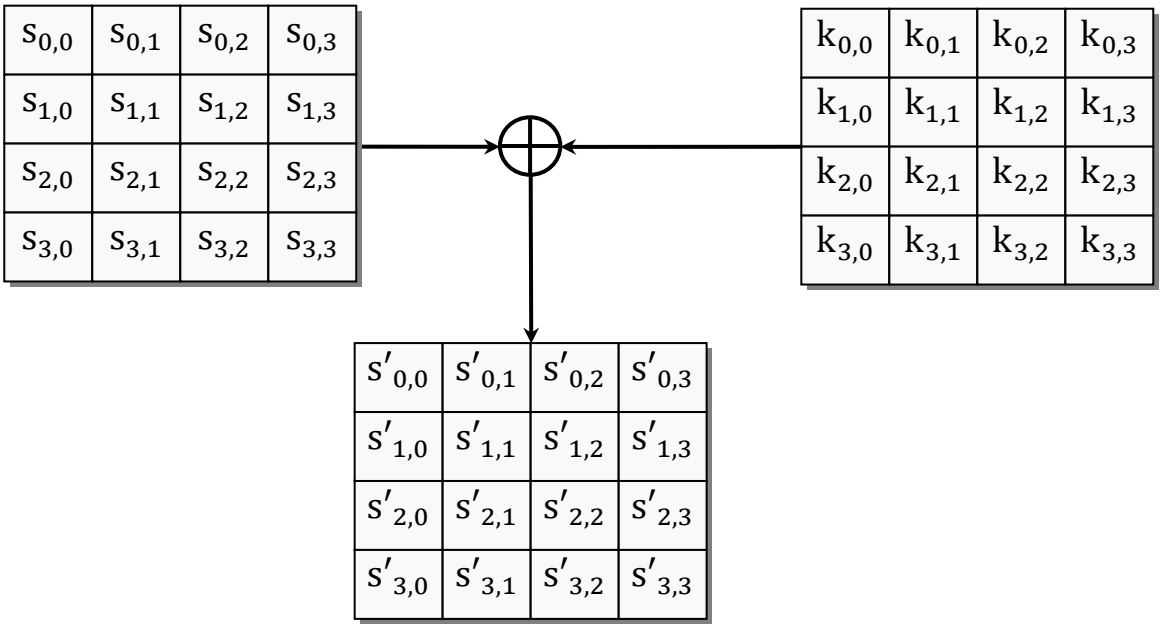



***Figure I.18:*** *l'étape AddRoundKey*

### E. Génération (extension) des clés

La clé étendue est un tableau à une dimension de mots de 32 bits noté ***W[Nb*(Nr+l)]*** dont les ***Nk*** premiers mots sont la clé principale Key0. Les autres mots sont obtenus récursivement à partir des mots précédents. La fonction d'expansion de la clé dépend de la valeur de ***Nk***. Il y a en fait deux fonctions d'expansion : une lorsque ***Nk*** $<$ 6 et une autre lorsque ***Nk*** $\geq$ 6.

Chaque ronde consiste une clé qui est composé de 4 mots de 32 bits cette clé est générée comme montre la figure I.19 :

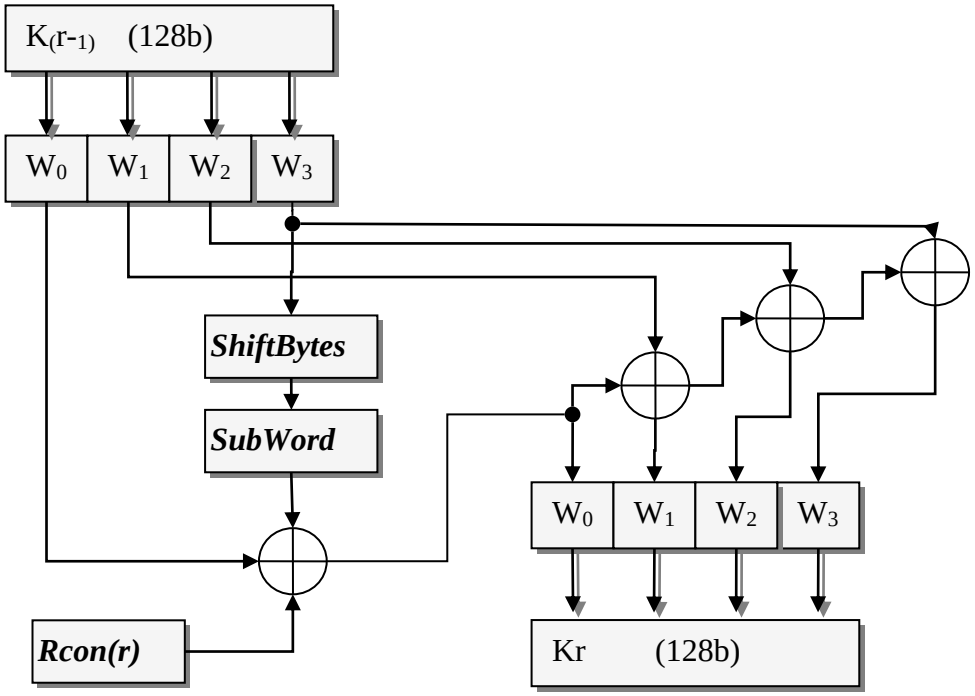



***Figure I.19:*** *Le schéma général d'extension d'une clé*

***SubWord :*** c'est une substitution de 32 bits des données qui est composée de quatre *SubBytes* en parallèle.

***ShiftBytes :*** c'est une transformation de décalage à gauche des bytes (8-bits) des données, et son fonctionnement est comme suite :

$$ShiftBytes(a,b,c,d) = (b,c,d,a),$$ avec **a**, **d**, **c** et **d** de 8-bits……..(I.3)

***Rcon(r) : c'***est un vecteur de ***Nr*** cases chaque case est coder en hexadécimale la valeur de la case ***r*** est ajoute à la sortie de *SubBytes* avec l'opération *xor***.**

La table utilisée de *Rcon[]* est donnée aux tableau I.3 :

***Tableau I.3:*** *La table Rcon utilisée dans l'extension des clés*

| ***R*** | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| ***Rcon[r]*** | 0x01 | 0x02 | 0x04 | 0x08 | 0x10 | 0x20 | 0x40 | 0x80 | 0x1b | 0x36 |

La règle de construction de cette table est :

***Rcon(1) = 1 et Rcon(r) = 2* Rcon(r-1)*** **;** **(**pour $1 < \mathbf{r} \leq 10$)……..(I.4)

***Remarque :***

- Un exemple détaillé d'expansion des clés est illustré en *Annexe* (B.1).
- Un exemple détaillé de chiffrement est illustré en *Annexe* (B.2).

**I.6.4.2.2. Déchiffrement [6]**

Rijndael a été conçu pour fonctionner de la même manière au chiffrement qu'au déchiffrement en remplaçant chaque opération élémentaire par son inverse, l'ordre du déchiffrement est l'inverse du chiffrement, les clés additionnées sont les mêmes du chiffrement, mais son ordre est décroissant **[7]**.

Les différentes transformations utilisées dans le déchiffrement sont *InvShiftRows, InvSubBytes, InvMixColumns,* et *AddRoundKey.*

**A. La substitution inverse (*InvSubBytes*)**

C'est la même opération que *SubBytes* (Figure I.15), seulement la table de substitution qui change.

La table de substitution utilisé dans le *InvSubBytes* est représenté en Hexadécimal, voir le tableau I.4. Par exemple pour $a_{1,1} = \{ed\}$, donc leur substitution est déterminée par l'intersection de la ligne d'indice '5' et la colonne d'indice '3', le résultat de la substitution obtenu est $a'_{1,1} = \{53\}$ .

***Tableau I.4:*** *La table de substitution utilisé dans le InvSubBytes*

| | | y | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | a | b | c | d | e | f |
| x | 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 |

### B. La décalage inverse de rangées (*InvShiftRows*)

Dans la transformation de *InvShiftRows* (Figure I.20), les bytes dans les trois dernières lignes de tableau de donnée sont cycliquement décalés à droite. La première ligne, r = 0, n'est pas décalée. Spécifiquement, la transformation de *InvShiftRows* procède comme suit :

$$\boldsymbol{S'_{r,c}} = \boldsymbol{S_{r,(c-r+Nb)mod\ Nb}}\ , \qquad avec\ (\mathbf{0} \le r \le 3\ \ et\ \ \mathbf{0} \le \boldsymbol{c} < Nb) \text{……..(I.5)}$$

Où $\boldsymbol{Nb}$ = 4 pour AES-128

La figure I.20 illustre la transformation *InvShiftRows* **:**

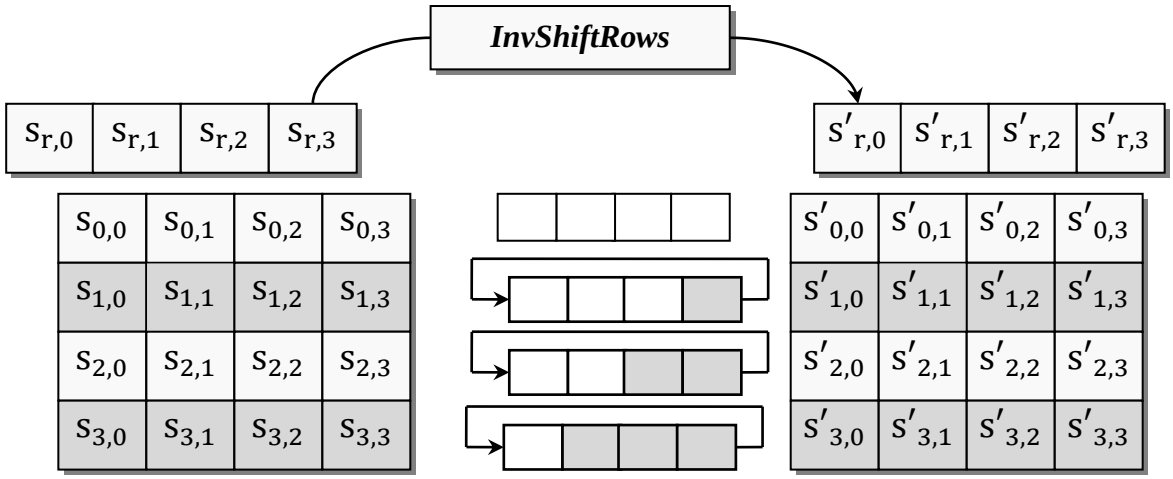



***Figure I.20:*** *la* transformation *InvShiftRows*

### C. Mélange inverse de colonnes (*InvMixColumns*)

Cette opération est différente de *MixColumns* (Figure I.17) dans la matrice de multiplication, cette matrice est illustrée dans les équations suivantes.

$$\begin{pmatrix} \boldsymbol{S'_{0,c}} \\ \boldsymbol{S'_{1,c}} \\ \boldsymbol{S'_{2,c}} \\ \boldsymbol{S'_{3,c}} \end{pmatrix} = \begin{pmatrix} \mathbf{0e} & \mathbf{0b} & \mathbf{0d} & \mathbf{09} \\ \mathbf{09} & \mathbf{0e} & \mathbf{0b} & \mathbf{0d} \\ \mathbf{0d} & \mathbf{09} & \mathbf{0e} & \mathbf{0b} \\ \mathbf{0b} & \mathbf{0d} & \mathbf{09} & \mathbf{0e} \end{pmatrix} \bullet \begin{pmatrix} \boldsymbol{S_{0,c}} \\ \boldsymbol{S_{1,c}} \\ \boldsymbol{S_{2,c}} \\ \boldsymbol{S_{3,c}} \end{pmatrix} \text{ Pour } \mathbf{0} \le \boldsymbol{c} \le \boldsymbol{Nb} \text{……..…..(I.6)}$$

$$\mathbf{S'_{0,c}} = (\{\mathbf{0e}\} \bullet \mathbf{S_{0,c}}) \oplus (\{\mathbf{0b}\} \bullet \mathbf{S_{1,c}}) \oplus (\{\mathbf{0d}\} \bullet \mathbf{S_{2,c}}) \oplus (\{\mathbf{09}\} \bullet \mathbf{S_{3,c}})$$
$$\mathbf{S'_{1,c}} = (\{\mathbf{09}\} \bullet \mathbf{S_{0,c}}) \oplus (\{\mathbf{0e}\} \bullet \mathbf{S_{1,c}}) \oplus (\{\mathbf{0b}\} \bullet \mathbf{S_{2,c}}) \oplus (\{\mathbf{0d}\} \bullet \mathbf{S_{3,c}}) \text{….(I.7)}$$
$$\mathbf{S'_{2,c}} = (\{\mathbf{0d}\} \bullet \mathbf{S_{0,c}}) \oplus (\{\mathbf{09}\} \bullet \mathbf{S_{1,c}}) \oplus (\{\mathbf{0e}\} \bullet \mathbf{S_{2,c}}) \oplus (\{\mathbf{0b}\} \bullet \mathbf{S_{3,c}})$$
$$\mathbf{S'_{3,c}} = (\{\mathbf{0b}\} \bullet \mathbf{S_{0,c}}) \oplus (\{\mathbf{0d}\} \bullet \mathbf{S_{1,c}}) \oplus (\{\mathbf{09}\} \bullet \mathbf{S_{2,c}}) \oplus (\{\mathbf{0e}\} \bullet \mathbf{S_{3,c}})$$

**Remarque**

- L'opération *AddRoundKey*, avec son extension de ses clés sont identiques que dans chiffrement.
- Dans le déchiffrement les clés sont ajoutées de manière décroissante (de clé 10 vers la clé 0), c'est le contraire du chiffrement.

## I.7. Conclusion

A travers ce chapitre, nous avons pu parcourir la cryptographie de sa naissance aux temps modernes. Tout au long du processus de sélection de l'algorithme Advanced Encryption Standard (AES), nous avons pu prendre conscience de l'implication de la communauté cryptographique dans ce procédé. L'étude des finalistes a permis de se plonger de manière concrète dans le mode de fonctionnement d'un algorithme de chiffrement symétrique et d'en comprendre les rouages élémentaires.

# Chapitre II

# Implémentation de l'algorithme AES sur un circuit FPGA

Sommaire

II.1. Introduction ..... 27
II.2. Les outils Software et Hardware de développement utilisés ..... 27
II.2.1. Présentation de l'environnement de développement "XILINX Integrated Software Environment (ISE) " ..... 27
II.2.2. La carte de développement Xtreme DSP Starter Kit-Spartan-3A DSP 1800A Edition ..... 29
II.3. Etapes pour l'implémentation d'une spécification HDL sur un FPGA ..... 31
II.3.1. Spécification HDL ..... 32
II.3.2. Vérification ..... 32
II.3.3. Implémentation ..... 32
II.3.4. Débogage du système ..... 33
II.4. Description de l'Architecture du circuit développé ..... 33
II.4.1. Circuit globale développé ..... 34
II.4.2. Architecture du module de cryptage ..... 35
II.4.2.1. Le sous-module Key_Gen ..... 37
II.4.2.2. Le sous-module AddRoundKey ..... 39
II.4.2.3 Le sous-module Shift_Row ..... 39
II.4.2.4. Le sous-module S_Box ..... 39
II.4.2.5. Le sous-module Mix_Columns ..... 40
II.4.2.6. Le sous-module Reg32_128 ..... 42
II.4.2.7. Le sous-module Enc_Controller ..... 42
II.4.3. Architecture du module de décryptage ..... 43
II.4.3.1. Le sous-module InvMix_Column ..... 45
II.4.3.2. Le sous-module InvS_Box ..... 47
II.4.3.3. Le sous-module InvShift_Row ..... 47
II.4.3.4. Le sous-module Dec_Controller ..... 48
II.4.4. Architecture du module de communication RS232 ..... 49
II.4.4.1. Le sous-module Rs232_rxd ..... 50
II.4.4.2. Le sous-module Bufer8_128 ..... 50
II.4.4.3. Le sous-module Rs232_128btxd ..... 51
II.5. Conclusion ..... 51

## II.1. Introduction

Le projet traite l'algorithme de chiffrement/déchiffrement AES et ses opérations.

Pour implémenter un circuit sur un FPGA, il y a trois éléments indispensables pour y parvenir :

1. Disposer la description logique du circuit (schématique, diagrammes d'états ou code VHDL).
2. L'environnement de développent est choisi en fonction du composant sur lequel le circuit sera implémenté et spécialement le logiciel du fabricant du composant (ISE : XILINX Integrated Software Environment).
3. Une carte de développement pour implémenter le circuit et vérifier son comportement.

Dans ce chapitre, il sera présenté l'architecture et la description des modules de cryptage/décryptage AES, l'implémentation du code VHDL, et la présentation de la plateforme de développement SPARTAN 3A DSP XC3SD1800A pour l'implémentation physique de l'algorithme AES.

Pour tester que le circuit AES fonctionne correctement, nous avons réalisés une simple interface de communication série (RS232) avec un ordinateur Figure II.1.

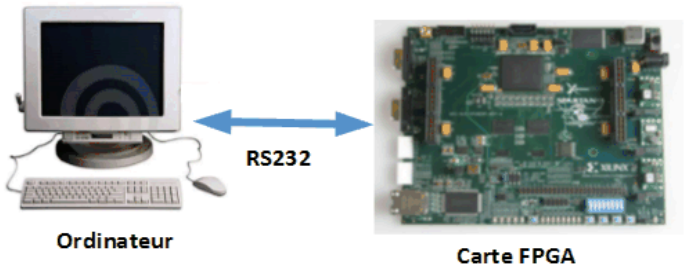



***Figure II.1 :*** *Le système de test*

## II.2. Les outils Software et Hardware de développement utilisés

Pour ce qui est de la synthèse et du placement/routage, nous utiliserons l'outil de développement intégré au FPGA ciblé. Il s'agit d'un SPARTAN 3A DSP XC3SD1800A chez Xilinx, donc cette étape se fera à l'aide de ISE Design Suite 14.1.

### II.2.1. Présentation de l'environnement de développement "XILINX Integrated Software Environment (ISE) "

C'est le logiciel de programmation des produits Xilinx (CPLD, FPGA Spartan et Virtex…) téléchargeable gratuitement sur le site internet de Xilinx dans sa version Web

Pack. Il intègre différents outils permettant de passer à travers le flot de conception d'un système numérique. Il dispose de :

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

C'est un outil de développent complet pour toutes les gammes des produits XILINX.

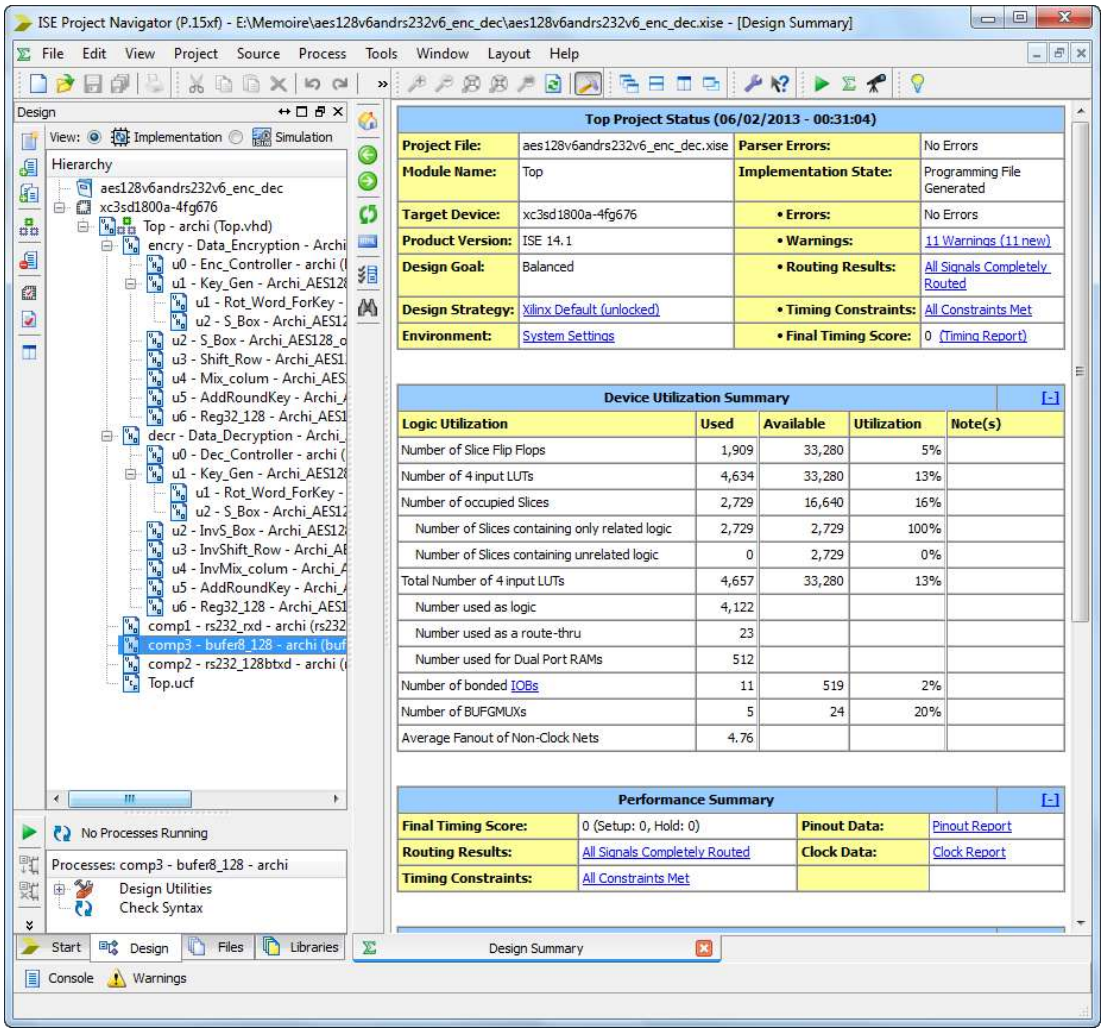



***Figure II.2 :*** *Le navigateur des projets de ISE Design Suite 14.1*

### II.2.2. La carte de développement Xtreme DSP Starter Kit-Spartan-3A DSP 1800A Edition

Pour implémenter l'algorithme AES sur FPGA, nous avons utilisé la carte "Xtreme DSP Starter Kit-Spartan-3A DSP 1800A Edition" (Figure II.3), elle est présentée dans la figure II.3, Cette carte possède les caractéristiques suivantes [12] **:**

- XC3SD1800A-4FGG676C Spartan-3A DSP FPGA
- Horloges 125 MHz LVTTL SMT Oscillateur
  - LVTTL Plot d'oscillateur
  - 25.175z LVTTL SMT Oscillateur (Horloge vidéo)
  - 25 MHz Horloge d'Ethernet (accessible à FPGA)
- Mémoire
  - 128 MB (32M x 32) DDR2 SDRAM
  - 16Mx8 Parallèle / BPI Configuration Flash
  - 64 Mb SPI Configuration / Mémoire Flash (avec 4 SPI supplémentaire choisit)
- Interfaces
  - 10/100/1000 PHY
  - JTAG Programmation/ Port de Configuration
  - Port RS232
  - Low-cost VGA
  - 4 SPI sélecteurs des lignes
- Boutons et Commutateurs
  - 8 LEDs
  - 8 Commutateurs
  - 4 Boutons poussoirs
  - Boutons poussoir de remise à zéro
- E/S
  - En-tête de 6 broches de Digilent (2)
  - Connecteur d'expansion EXP (2)
  - 30 broches GPIO Connecteur peut être utilisé pour System ACE™ Compact Flash carte de fille (non inclus)

- Configuration et Débogage
  - JTAG
  - Connecteur de module de l'ACE de système
  - Eridon debug connector (SATA)

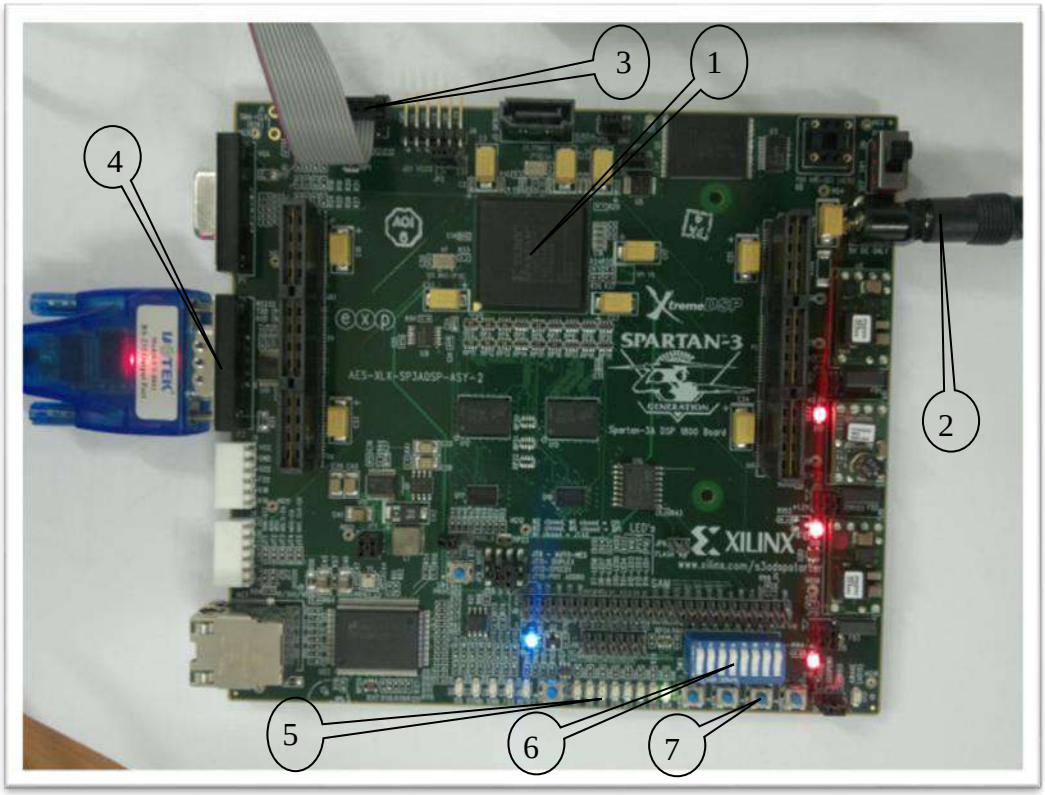



***Figure II.3 :*** *Vue externe de la plateforme XtremeDSP Starter Kit - Spartan-3A DSP 1800A Edition*

1) Le Circuit FPGA XC3SD1800A-4FGG676C
2) Connecteur d'alimentation de la carte
3) Port de programmation/configuration
4) Port série RS-232
5) 8 LEDS d'utilisateur
6) 8 interrupteurs d'utilisateur
7) 4 boutons poussoir d'utilisateur

## II.3. Etapes pour l'implémentation d'une spécification HDL sur un FPGA [11]

Pour implémenter une spécification HDL sur un FPGA, sur la plateforme ISE Design Suite 14.1 quatre étapes sont à suivre tel illustré par la figure suivante (II.4).

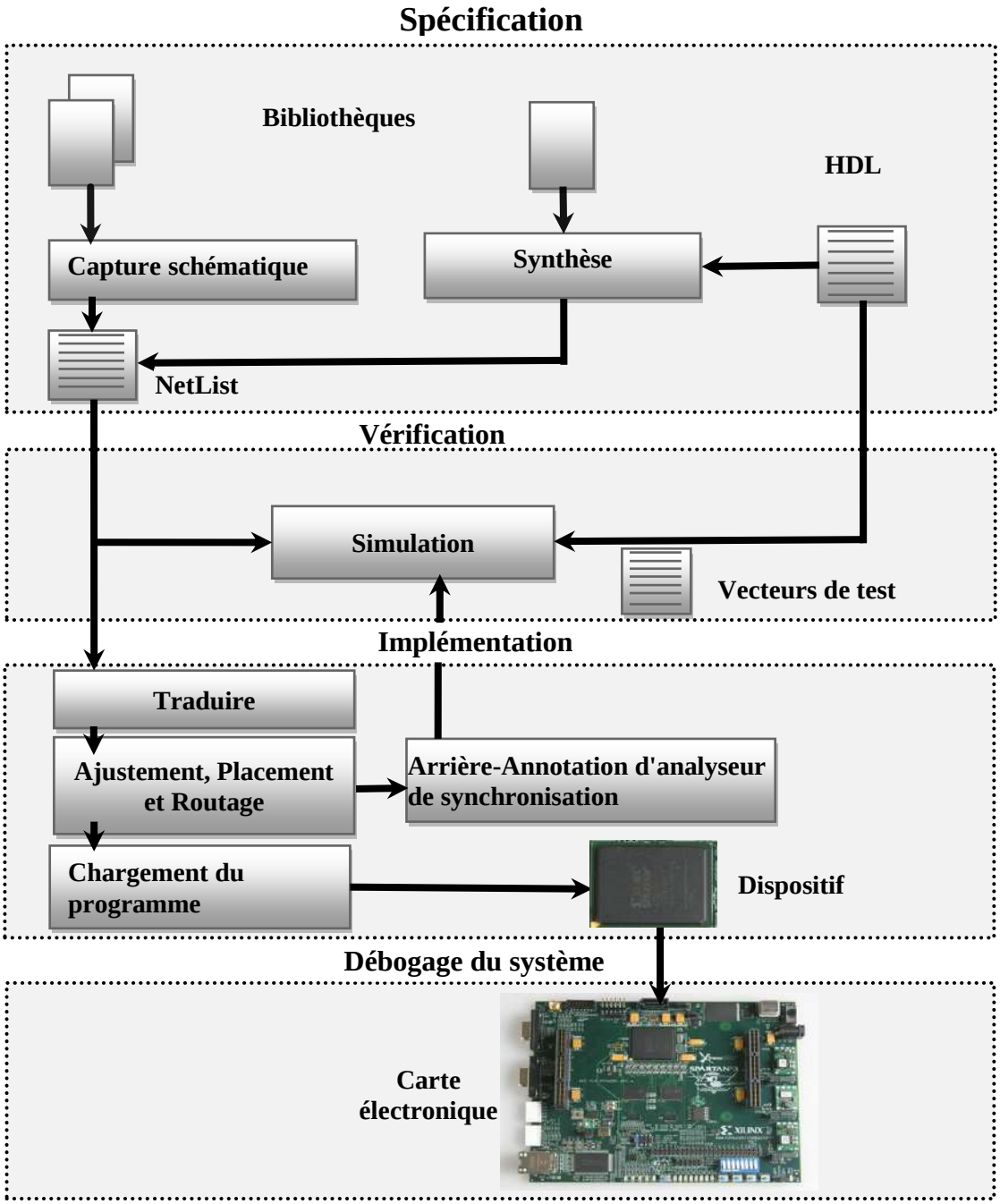



***Figure II.4 :*** *Les étapes d'implémentation d'un circuit sur un circuit logique programmable (XILINX)*

### II.3.1. Spécification HDL [11]

Le terme spécification est un terme qui regroupe les trois modes (schématique, diagrammes d'états ou HDL) de saisie d'un circuit électronique. La spécification HDL est synthétisé pour générer un fichier appelé NETLIST qui décrit les interconnexions entre les registres.

### II.3.2. Vérification [11]

La vérification du design est une étape parallèle où le concepteur observe le comportement du code et observe s'il se comporte tel qu'il est supposé. Un simulateur simule le circuit à condition de lui fournir les vecteurs de test. Les vecteurs de test peuvent se présenter sous plusieurs formes, la plus courante est les TESTBENCHS rédigés dans un langage de description matériel ou sous forme de lignes de code TCL pour entrer les instructions au simulateur. En appliquant les vecteurs de test sur le code, le simulateur fournit les sorties du circuit.

### II.3.3. Implémentation [11]

Une fois le Netlist (les interconnexions entre les portes logiques) décrit la conception en utilisant les portes logiques en tenant compte du fabriquant et d'une famille de composants bien définie. Une fois la vérification est terminée, le circuit est implémenté sur le composant en spécifiant les références exactes de celui-ci à savoir : le boitier, la fréquence de travail et les autres options spécifiques à chaque composant. Cette étape est conclu par un rapport de tous les sous-programmes exécutés, les avertissements, les erreurs, les I/O utilisés, des données qui permettent de savoir si le composant choisit est le mieux adapté pour l'application ciblé.

Cette étape se décompose à son tour en sous-étapes comme suit :

- ***Fitting :*** c'est le dimensionnement de la conception en fonction des ressources internes du composant cible.

- ***Place and route :*** les sous-programmes de placement et routage sont exécutés après compilation du code.

  **Place :** c'est le processus de sélection des modules ou blocks logiques où les portes logiques seront placées.

  **Route :** le routage est l'interconnexion physique entre les différents blocks logiques.

- ***Downloading ou programming :*** après que le fichier de programmation est généré vient l'étape où il faut le charger sur le FPGA ciblé par l'application. Les FPGA de type

SRAM nécessite une mémoire de type PROM pour avoir le programme même après extinction de la carte électronique comportant le FPGA, au démarrage il y a chargement ou downloading des interconnexions sur FPGA. Dans le cas des FPGA à anti-fusibles le programme peut être chargé ou programmé par l'intermédiaire d'un programmeur ou par la technique de programmation sur circuit JTAG (Joint Test Action Group est le nom de la norme IEEE 1149.1 intitulé "Standard Test Access Port and Boundary-Scan Architecture").

### II.3.4. Débogage du système [11]

Après chargement des interconnexions sur le FPGA, des tests peuvent être effectués sur le circuit implémenté à fin de voir le comportement du circuit et détecter les anomalies à fin de les corriger. En cas de rien d'anomalie, la spécification est validée et le prototypage est considéré opérationnel.

## II.4. Description de l'Architecture du circuit développé

Dans cette partie, il sera présenté la description de l'architecture des différents modules du circuit de cryptage/décryptage AES ainsi que les différents modèles VHDL de tout le projet.

Dans ce circuit développé, pour tous les modules de l'architecture, il est utilisé le signal d'horloge pour activer les différentes opérations. L'utilisation de l'horloge a pour but de synchroniser les modules et les bits de donnée en sortie de chaque module.

Pour réaliser le circuit de l'AES, on peut distinguer plusieurs architectures. Les architectures les plus connus et les plus utilisés par des développeurs dans le domaine sont :

1. Architecture qui fait le traitement de toute la donnée de 128 bits. Cette architecture fait les opérations (*S-Box, MixColumns, AddRoundKey et ShiftRows*) sur toute la donnée de 128 bits. Cette architecture a comme avantage la vitesse de traitement est très grande, mais l'inconvénient est qu'elle demande beaucoup de surface dans l'FPGA.

2. Architecture qui fait le traitement des 4 mots de 32 bits de la donnée de 128 bits (l'un après l'autre). Cette architecture fait les opérations (*S-Box, MixColumns*) sur chaque 32 bits de la donnée de 128 bits. Cette architecture à une vitesse de traitement moins que la 1$^{ière}$ architecture, mais elle ne demande pas beaucoup de surface dans l'FPGA.

3. Architecture qui fait le traitement des 16 mots de 8 bits de la donnée de 128 bits (l'un après l'autre). Cette architecture fait les opérations (*S-Box*) sur chaque 8 bits de la

donnée de 128 bits. Cette architecture à petite vitesse de traitement, c'est la plus lente que la 1ière et la 2ième, mais en termes d'occupation dans l'FPGA est la plus économique.

Dans ce projet, l'architecture choisie est la 2ième, puisque cette architecture fait un compromis des deux critères de performances (la vitesse et l'occupation du circuit).

Dans notre architecture, nous avons séparés complètement les deux modules (*Data_Encryption* et *Data_Decryption*), pour l'utilisation du cryptage et du décryptage simultanément dans le domaine de la communication.

### II.4.1. Circuit globale développé

Dans ce circuit, et avec un sélecteur, l'utilisateur peut choisir le test d'une fonction parmi les suivantes : chiffré, déchiffré, chiffré et déchiffré.

Le principe général de fonctionnement du circuit est comme suit :

Le message clair ou chiffré est envoyé du PC à travers le port RS-232 en série. Ce message passera par l'interface RS-232 de la carte FPGA où le message sera composé en trames de 8bits, ces trames sont enregistrés dans un registre (*Bufer8_128)* de 128 bits, après que le registre soit rempli de 128 bits complètement, ces bits seront prêtes à être chiffrer/déchiffrer.

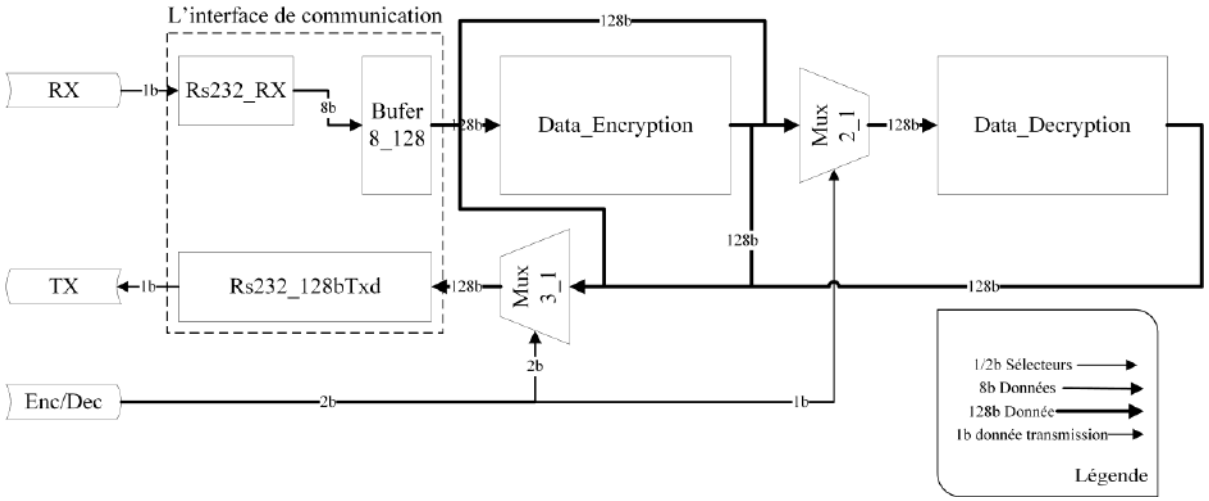
Pour le chiffrement, le module *Data_Encryption* reçoit les 128 bits, puis il applique toutes les opérations nécessaires concernant le chiffrement par l'algorithme AES, en sortie du module, on obtient une séquence de 128 bits du message chiffré.

Pour le déchiffrement le module *Data_Decryption* reçoit les 128 bits du message chiffré et applique toutes les opérations nécessaires concernant le déchiffrement par l'algorithme AES, en sortie du module on obtient une séquence de 128 bits du message déchiffré.

Après le chiffrement et/ou déchiffrement du message, il sera envoyé à nouveau vers le PC à travers l'interface RS232.

Chaqu'un des modules *Data_Encryption et Data_Decryption* possède son propre générateur des clés des rondes. Il est possible d'utiliser un seul générateur des clés commun pour les deux modules, mais il prend plus du temps puisqu'il est possible que les deux modules accèdent en même temps au générateur des clés.

Les différents modules du circuit global développé sont illustrés dans la figure II.5 :




Le circuit global inclus les principaux modules suivants :

1. Le module de cryptage *Data_encryption.*
2. Le module de décryptage *Data_Decryption.*
3. Les modules de la liaison série RS232 : *Bufer8_128, Rs232_RX* et *Rs232_128bTxd.*

### II.4.2. Architecture du module de cryptage

Le principe de fonctionnement du circuit est comme suit :

Pour chaque ronde, il faut une clé qui est générée par le sous-module *Key_Gen* pour faire l'opération *AddRoundKey* entre la donnée et cette clé.

Le résultat de l'opération *AddRoundKey* sur 128 bits va entrer dans le sous-module *ShiftRow* qui fait une opération de décalage.

Le résultat du sous-module *ShiftRow* se divise en quatre mots de 32 bits, chaque mot va s'affecter au sous-module *SubBytes* puis le résultat obtenu passe au sous-module *MixColumns.*

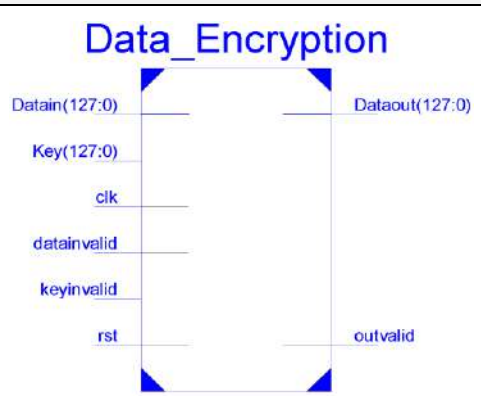
Les quatre mots de 32 bits du message vont être mémorisés dans un registre de 128 bits (*Reg32_128bits*) pour faire la ronde suivante.

Pour la 10$^{ème}$ ronde l'opération *MixColumns* n'est pas faite, Après la 10$^{ème}$ ronde, le résultat final (*Dataout*) est obtenu avec une ligne de validation *Outvalid* sur la sortie de sous-module *AddRoundKey.*

Le sous-module *Enc_Controller* fait la vérification de la présence des données dans l'entrée (message clair), et synchronise les différents sous-modules de cryptage, c'est lui qui

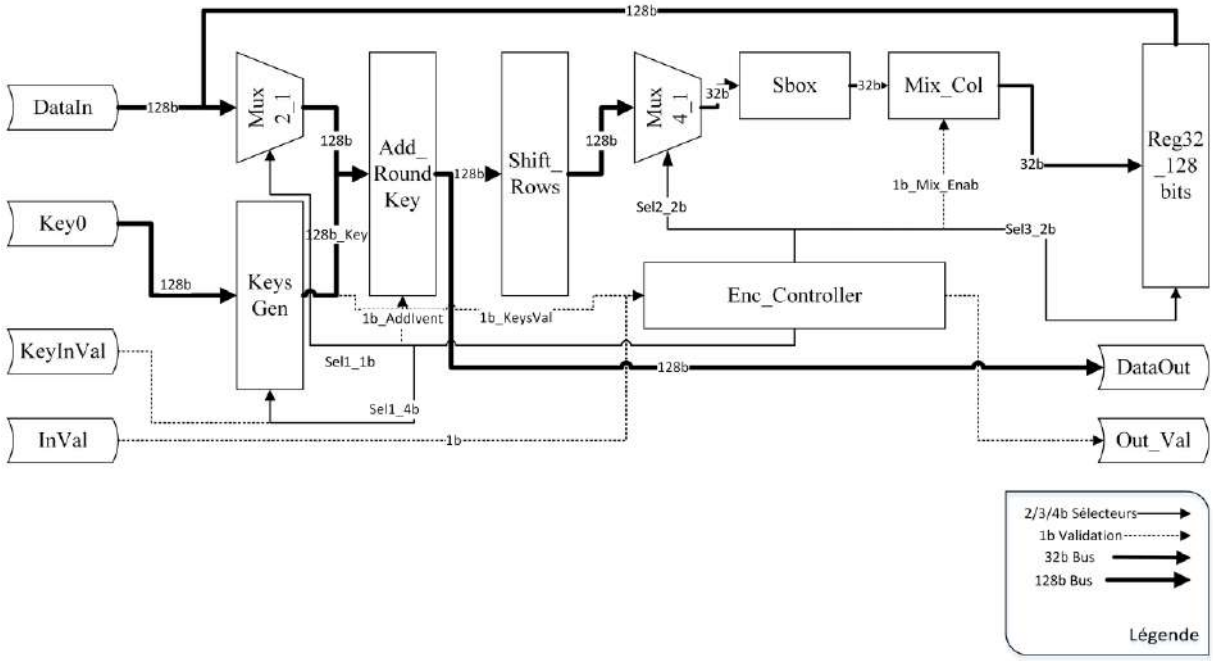
active les opérations : *AddRoundKey, SubBytes, ShiftRows et MixColumns* ainsi que la validation de la sortie finale (message crypté).

***Tableau II.1 :*** *description des signaux Entrées/Sorties du module Data_Encryption*

| Data_Encryption.vhd | | |
|---|---|---|
| Port | Description | Schéma bloc du module |
| Datain | les données claires à crypter |  |
| Key | la clé de cryptage | |
| Clk | signal d'horloge (freq = 125 MHZ) | |
| Datainvalid | les données d'entrée sont prêtes | |
| Keyinvalid | la clé est valide | |
| Rst | remise à zéro du circuit | |
| Dataout | les données en sortie cryptées | |
| Outvalid | fin de cryptage | |



Les différents modules du circuit de cryptage sont illustrés dans la figure II.6 :




Le module de cryptage inclus les différents sous-modules suivants :

1. Le sous-module *Key_Gen.*
2. Le sous-module *AddRoundKey.*
3. Le sous-module *Shift_Row.*
4. Le sous-module *S_Box.*
5. Le sous-module *Mix_Columns.*
6. Le sous-module *Reg32_128.*
7. Le sous-module *Enc_Controller.*

### II.4.2.1. Le sous-module Key_Gen

Ce module fait l'expansion des clés de chaque ronde. Il reçoit en entrée la clé initiale *key0*, puis il fait la génération des 10 clés des rondes (§ *I.6.4.2.1.E).* Ces clés seront mémorisées dans une RAM de 160 mots de 8 bits. Le bus d'adresse *Round* fait la sélection (lecture/écriture) d'une clé dans la RAM.

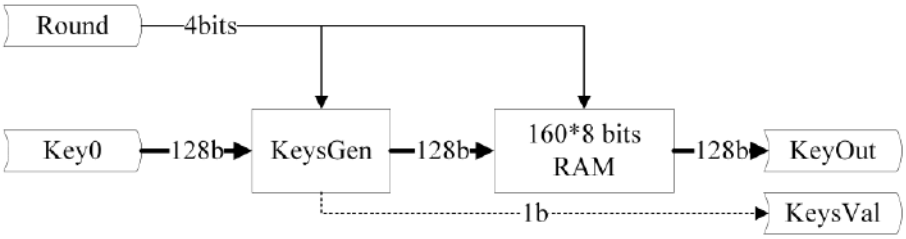



***Figure II.7 :*** *Schéma bloc du module Key_Gen*

Chaque clé générée est obtenu en passant par les étapes suivantes :

a) Diviser la clé de 128 bits en quatre (04) mots de 32 bits $k_0$, $k_1$, $k_2$ et $k_3$.

b) Le premier mot de la clé à générer $k'_0$ est obtenu en passant par les opérations suivantes :

- Un décalage à droite des octets du mot de 32 bits $k_3$.
- Le résultat passe par une table S-Box pour faire la substitution des quatre octets en parallèle.
- Le premier octet à gauche de la sortie de substitution fait un XOR avec la valeur de la case correspondante du vecteur *Rcon*. La table du vecteur de *Rcon* est la suivante (Tableau II.2) :

***Tableau II.2:*** *Table Rcon utilisée dans l'extension des clés*

| ***R*** | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| ***Rcon[r]*** | 0x01 | 0x02 | 0x04 | 0x08 | 0x10 | 0x20 | 0x40 | 0x80 | 0x1b | 0x36 |

c) Les trois derniers mots de la clé à générer ($k'_1$, $k'_2$ et $k'_3$) sont obtenu à partir d'un XOR des deux mots : le mot de la clé précédente $k_i$ avec le précédent mot de la clé $k'_{i-1}$ ($i = 1, 2\ ou\ 3$ ).

Le détail de la génération d'une clé est comme le montre la figure II.8.

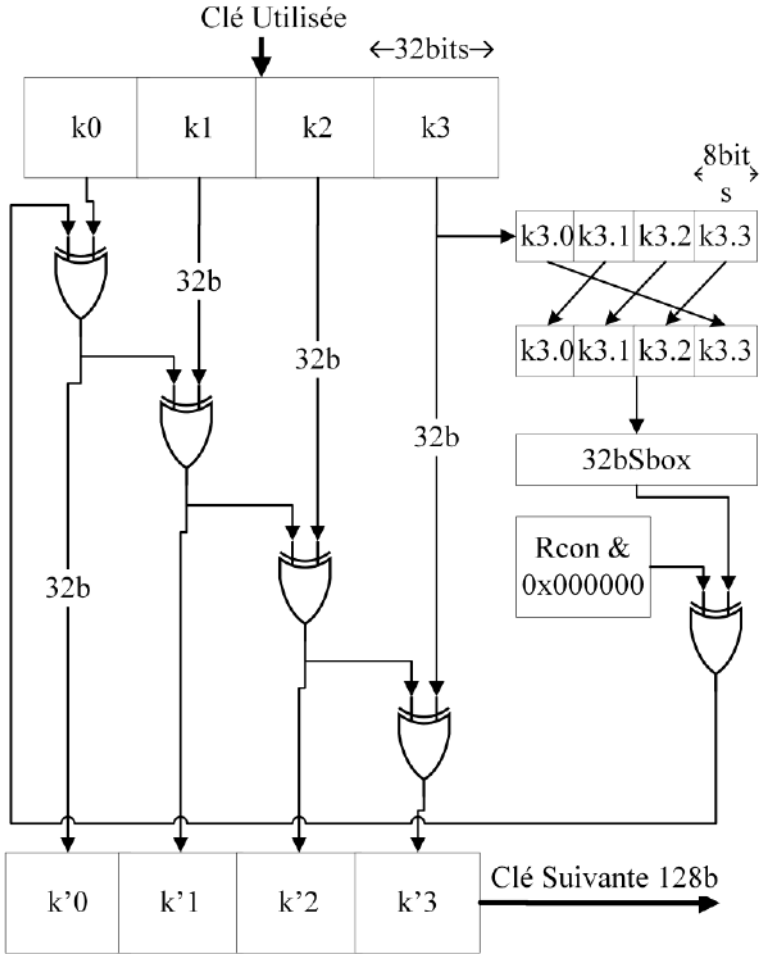
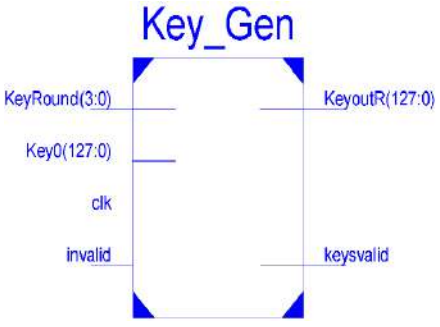



***Figure II.8 :*** *Schéma de génération d'une clé de ronde*

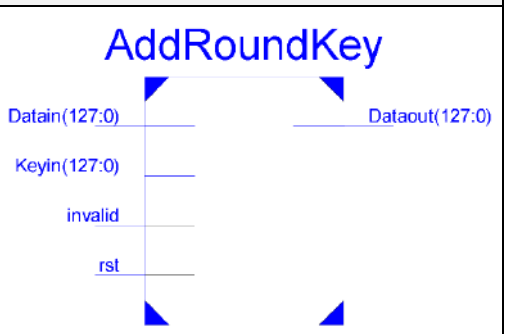
***Tableau II.3 :*** *description des signaux Entrées/Sorties du module Key_Gen*

| Key_Gen.vhd | | |
|---|---|---|
| Port | Description | Schéma bloc du module |
| KeyRound | sélecteur de ronde |  |
| Key0 | clé initiale pour générer les clés des rondes | |
| Clk | signal d'horloge (freq = 125 MHZ) | |
| InValid | début de génération | |
| KeyoutR | clé de sotie d'une ronde | |
| KeysValid | toutes les clés sont générées | |

### II.4.2.2. Le sous-module AddRoundKey

Ce module fait un *XOR* entre deux entrées de 128 bits, et donne le résultat de l'opération sur 128 bits. Ce module est entièrement combinatoire. Il possède deux entrées de 128 bits et une sortie de 128 bits (§ *I.6.4.2.1.D).*

***Tableau II.4 :*** *description des signaux Entrées/Sorties du module AddRoundKey*

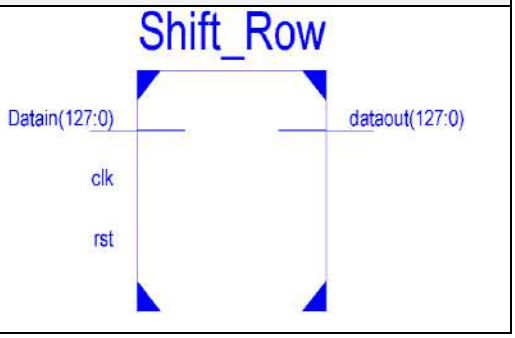
| AddRoundKey.vhd | | |
|---|---|---|
| Port | Description | Schéma bloc du module |
| Datain | donnée à additionner avec la clé |  |
| Keyin | clé à additionner avec la donnée | |
| Invalid | activation du module | |
| Rst | remise à zéro du circuit | |
| Dataout | résultat de l'addition clé et donnée | |



### II.4.2.3 Le sous-module Shift_Row

Ce module dispose d'une entrée de 128 bits, il fait des rotations sur les lignes de la matrice d'entrée, le résultat de sortie est 128 bits (§ *I.6.4.2.1.B).*

***Tableau II.5 :*** *description des signaux Entrées/Sorties du module ShiftRow*

| Shift_Row.vhd | | |
|---|---|---|
| Port | Description | Schéma bloc du module |
| Datain | donnée avant le décalage |  |
| Clk | signal d'horloge (freq = 125 MHZ) | |
| Rst | remise à zéro du circuit | |
| Dataout | donnée après le décalage | |



### II.4.2.4. Le sous-module S_Box

Ce module réalise une substitution d'une donnée de 32 bits d'entrée, la donnée est subdivisée en 4 mots de 8 bits. Comme montre la figure II.9, ce module utilise 4mémoires ROM de 256 mots de 8 bits pour la substitution des différents mots. Les 8 bits de chaque entrée servent à sélectionner une case dans la table *Sbox* (§ *I.6.4.2.1.A).*

Les 4 substitutions seront concaténer en sortie pour constituer la donnée de 32 bits.

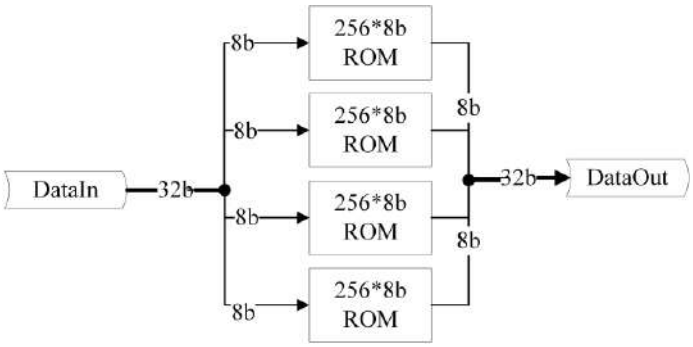
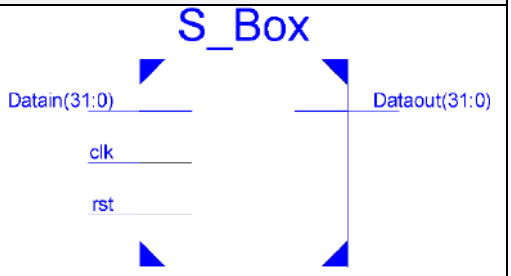



***Figure II.9 :*** *Schéma bloc du module S-Box*

***Tableau II.6 :*** *description des signaux Entrées/Sorties du module S_Box*

| S_Box.vhd | | |
|---|---|---|
| Port | Description | Schéma bloc du module |
| Datain | donnée avant la substitution |  |
| Clk | signal d'horloge (freq = 125 MHZ) | |
| Rst | remise à zéro du circuit | |
| Dataout | donnée après la substitution | |



### II.4.2.5. Le sous-module Mix_Columns

Ce module travaille sur un mot de 32 bits d'entrée qui multiplie ses octets dans le champ fini GF($2^8$) par une matrice prédéfinie (§ *I.6.4.2.1.C)*. La sortie obtenue est de 32 bits Figure II.10.

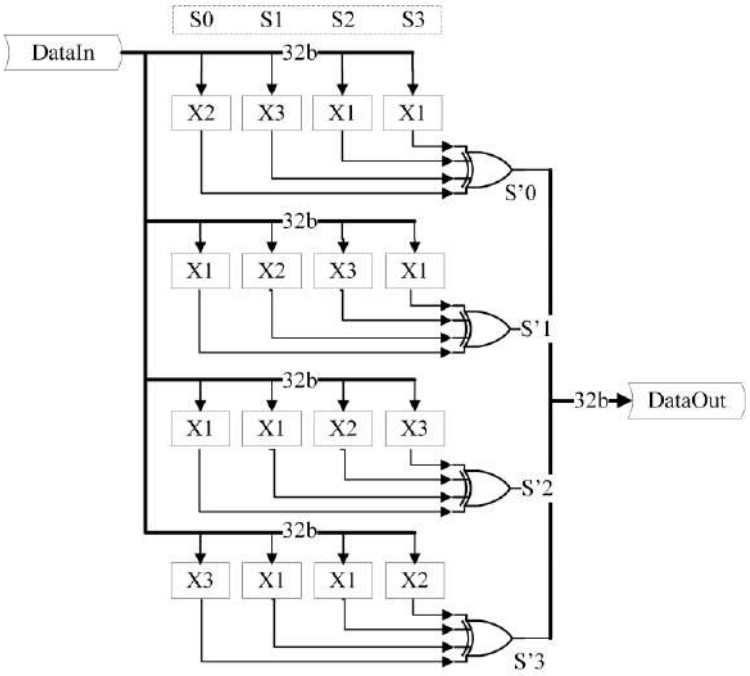



***Figure II.10 :*** *Schéma bloc du module MixColumns*

La multiplication dans GF($2^8$) par *2* se réalise comme suit :

Si le bit de poids fort de la donnée 8 bits est à 1, alors on fait un XOR entre la donnée décalée d'un bit à gauche et la séquence *0x1b*. Si non la sortie est seulement la donnée décalée d'un bit à gauche comme montre la figure II.11.

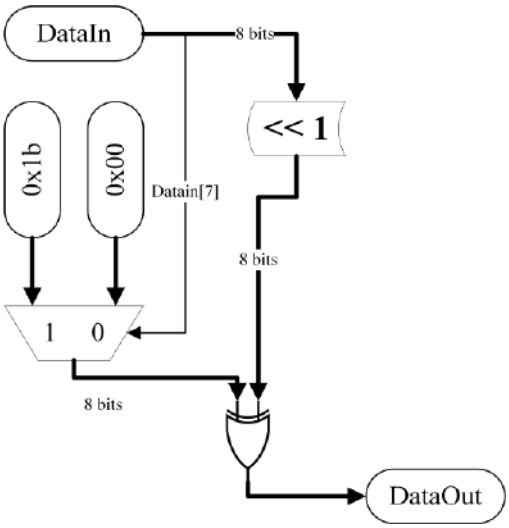



***Figure II.11 :*** *Schéma bloc de la multiplication dans GF($2^8$) par 2*

Comme il est montré précédemment, la multiplication dans GF($2^8$) d'un octet par *1*, *2*, ou *3* se fait comme suit (Figure II.12) :

- La multiplication par *2* est montrée dans la figure II.11.
- La multiplication par *3* est la multiplication par *2 XOR* cet octet.
- La multiplication par *1* est l'octet lui-même.

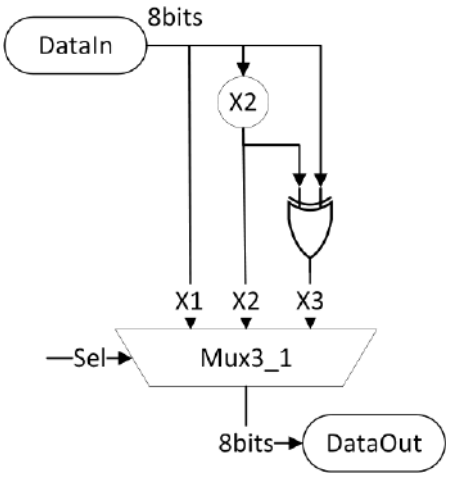
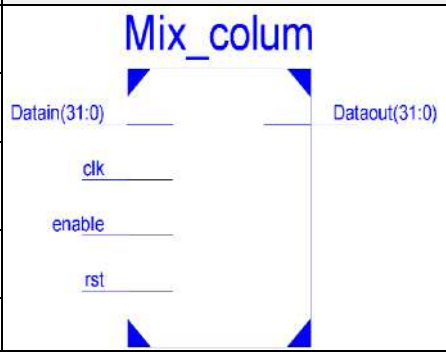



***Figure II.12 :*** *Schéma bloc de la multiplication par 1, 2 et 3*

Dans ce projet, la multiplication dans GF(28) par 1, 2, 3 est réalisé comme une fonction en VHDL qui porte le nom *Mix_prod.*

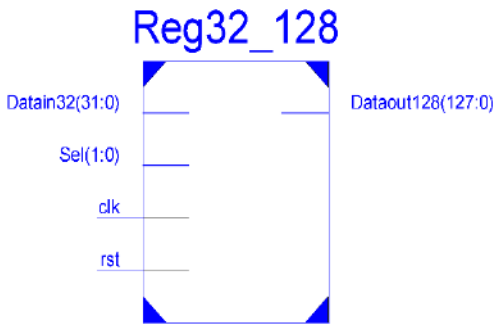
***Tableau II.7 :*** *description des signaux Entrées/Sorties du module MixColumns*

| Mix_Columns.vhd | | |
|---|---|---|
| Port | Description | Schéma bloc du module |
| Datain | donnée avant le mélange des colonnes |   |
| Clk | signal d'horloge (freq = 125 MHZ) | |
| Enable | active/désactive l'opération MixColumns (si elle est désactive alors la sortie = l'entrée) | |
| Rst | remise à zéro du circuit | |
| Dataout | donnée après le mélange des colonnes | |

### II.4.2.6. Le sous-module Reg32_128

La donnée obtenue après l'opération de MixColumns est composée de 4 mots de 32 bits. Ces mots sont mémorisés dans un registre de 128 bits en utilisant un démultiplexeur 1-4 avec mémorisation pour effectuer l'opération selon la valeur du sélecteur *sel*.

***Tableau II.8 :*** *description des signaux Entrées/Sorties du module Reg32_128*

| Reg32_128.vhd | | |
|---|---|---|
| Port | Description | Schéma bloc du module |
| Datain | donnée à mémoriser dans Reg32_128 |   |
| Sel | Sélecteur de démultiplexage | |
| Clk | signal d'horloge (freq = 125 MHZ) | |
| Rst | remise à zéro du circuit | |
| Dataout | résultat d'assemblage des 4 mots de 32bits | |

### II.4.2.7. Le sous-module Enc_Controller

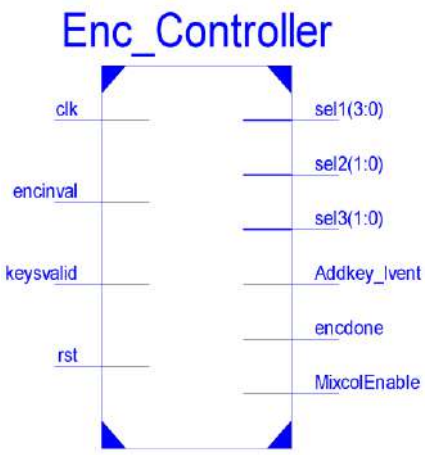
Ce contrôleur fait la synchronisation des différents modules de cryptage : *AddRoundKey*, S-*Box*, et *MixColumns*. Il permet de lancer le cryptage pour une nouvelle donnée lorsque les clés sont prêtes : sélectionner la ronde à exécuter, le mot à substituer, le mot à enregistrer pendant la ronde, l'indication de démarrage de *AddRoundKey*, l'activation de *MixColumns*, ainsi que la validation de la sortie.

La table d'ordonnancement (en mode pipeline) des différentes unités de ce contrôleur (séquenceur) est montrée dans le tableau II.9.

***Tableau II.9 :*** *La table d'ordonnancement des unités du module de cryptage*

| **Cycles** | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |
|---|---|---|---|---|---|---|---|---|---|---|
| **Unités** | ADD | SHIFT | Sel2=0<br>SHIFT | Sel2=1<br>SBOX | Sel2=2<br>SBOX<br>MIXC<br><br>Sel3=0 | Sel2=3<br>SBOX<br>MIXC<br>REG<br>Sel3=1<br>Sel1++ | SBOX<br>MIXC<br>REG<br>Sel3=2 | MIXC<br>REG<br>Sel3=3 | REG | AddIv |

***Tableau II.10 :*** *description des signaux Entrées/Sorties du module Enc_Controller*

| Enc_Controller.vhd | | |
|---|---|---|
| Port | Description | Schéma bloc du module |
| Clk | signal d'horloge (freq = 125 MHZ) |  |
| Encinval | le message est prêt à crypter | |
| Keysvalid | les clés sont prêtes | |
| Rst | remise à zéro du circuit | |
| Sel1 | sélecteur de ronde | |
| Sel2 | sélecteur de *mux* avant S-Box | |
| Sel3 | sélecteur de *demux* après *Mix_Columns* | |
| AddKey_Ivent | effectuer *AddRoundKey* | |
| Encdone | valide en sortie la donnée cryptée | |
| MixcolEnable | active/ désactive *Mix_Columns* | |



### II.4.3. Architecture du module de décryptage

Le principe de fonctionnement du circuit est comme suit :

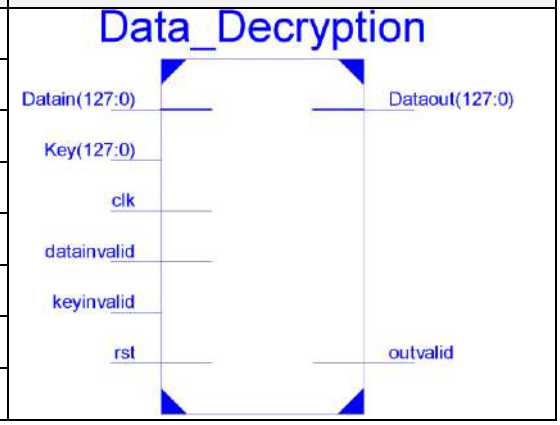
Pour chaque ronde, il faut une clé de ronde qui est générée par le sous-module *Key_Gen* pour faire l'opération *AddRoundKey* entre la donnée et cette clé.

Le résultat du sous-module *AddRoundKey* se divise en quatre mots de 32 bits, chaque mot va s'affecter au sous-module *InvMixColumns* puis le résultat obtenu passe au sous-module *InvSubBytes*. Ces derniers sont mémorisés dans un registre de 128 bits (*Reg32_128bits*) pour faire l'opération *InvShiftRow*. Le résultat de cette opération est envoyé dans la ronde suivante.

Pour la 1ère ronde l'opération *InvMixColumns* n'est pas faite. Et en fin, le résultat final (*Dataout*) est obtenu avec une validation *Outvalid* sur la sortie du sous-module *AddRoundKey.*

Le sous-module Dec*_Controller* fait la vérification de la présence des données dans l'entrée (message crypté), et synchronise les différents sous-modules de décryptage, c'est lui qui active les opérations : *AddRoundKey*, *InvSubBytes*, *InvMixColumns*, *InvShiftRows*, ainsi que la validation de la sortie finale (message décrypté).

***Tableau II.11 :*** *description des signaux Entrées/Sorties du module Data_Decryption*

| Data_Decryption.vhd | | |
|---|---|---|
| Port | Description | Schéma bloc du module |
| Datain | donnée à décryptée |  |
| Key | clé de décryptage | |
| Clk | signal d'horloge (freq = 125 MHZ) | |
| Datainvalid | les données d'entrée sont prêtes | |
| Keyinvalid | la clé est valide | |
| Rst | remise à zéro du circuit | |
| Dataout | donnée en sortie décryptée | |
| Outvalid | fin de décryptage | |



Les différents modules du circuit de décryptage sont illustrés dans la figure II.13 :

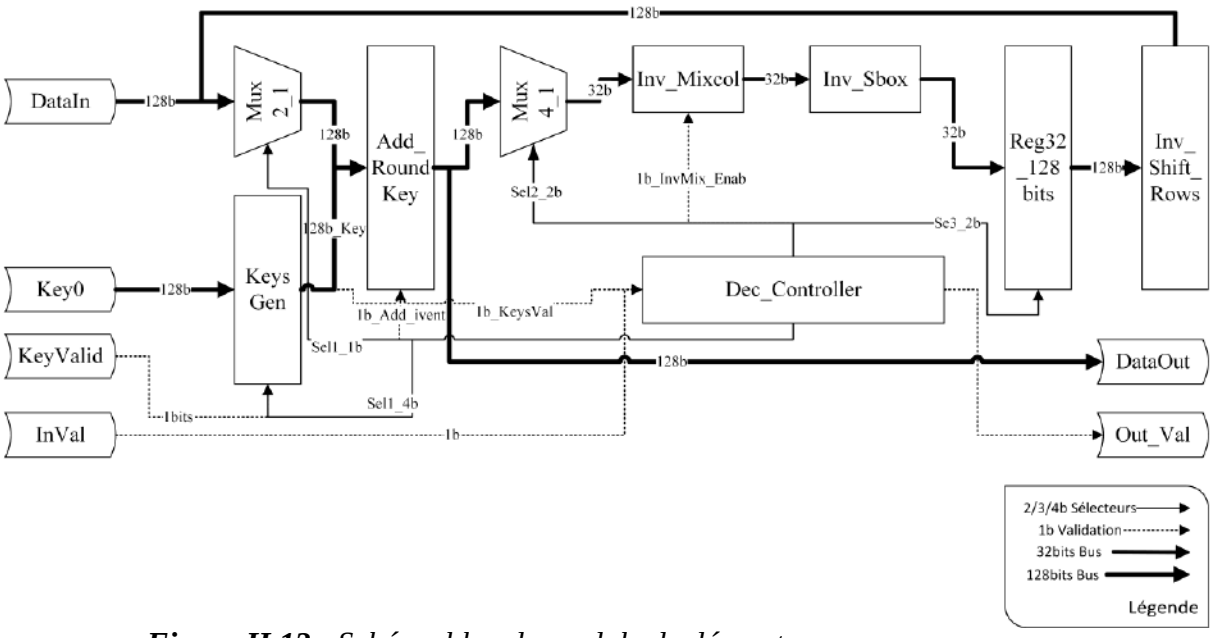



***Figure II.13 :*** *Schéma bloc du module de décryptage*

Le module de décryptage inclus les différents sous-modules suivants :

1. Le sous-module *Key_Gen*
2. Le sous-module *AddRoundKey*
3. Le sous-module *InvMix_Columns*
4. Le sous-module *InvS_Box*
5. Le sous-module *Reg32_128*
6. Le sous-module *InvShift_Row*
7. Le sous-module *Dec_Controller*

### II.4.3.1. Le sous-module InvMix_Column

Ce module travaille sur un mot de 32 bits d'entrée qui multiplie ses octets dans le champ fini GF($2^8$) par une matrice prédéfinie (§ *I.6.4.2.2.C)*. La sortie obtenue est de 32 bits (figure II.14).

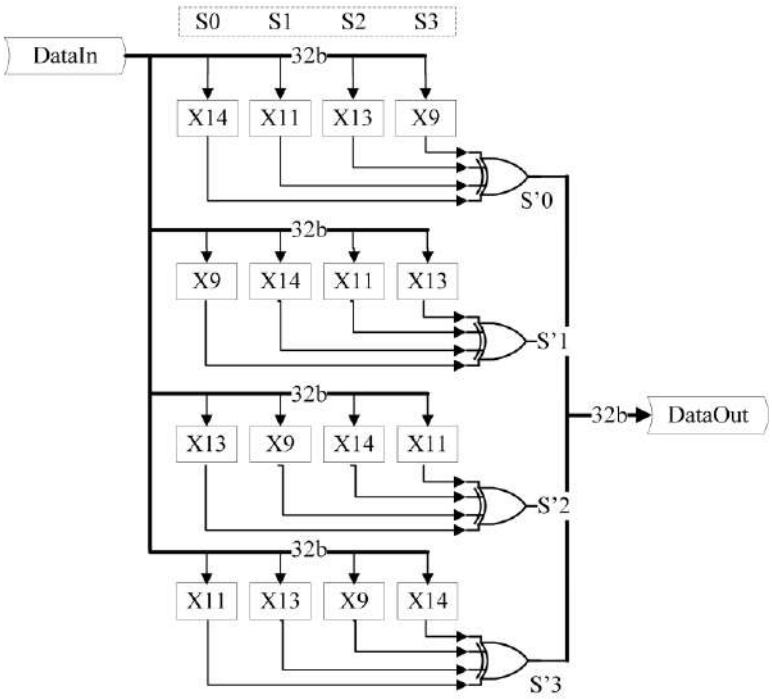



***Figure II.14 :*** *Schéma bloc du module InvMixColumns*

La multiplication dans GF($2^8$) d'un octet par *9*, *11*, *13* ou *14* se fait comme suit (figure II.15) :

- La multiplication par *9* est la multiplication par 8 *XOR* cet octet.
- La multiplication par *11* est la multiplication par 8 *XOR* (la multiplication par 2 *XOR* cet octet).

- La multiplication par *13* est la multiplication par 8 *XOR* (la multiplication par 4 *XOR* cet octet).
- La multiplication par *14* est la multiplication par 8 *XOR* (la multiplication par 4 *XOR* la multiplication par 2).

**Remarque**

- La multiplication dans GF($2^8$) par 2 est détaillée dans le module *MixColumns* (figure II.11).
- La multiplication dans GF($2^8$) par 8 est de trois multiplications dans GF($2^8$) par 2.
- Cette multiplication est réalisée comme une fonction *InvMix_prod.*

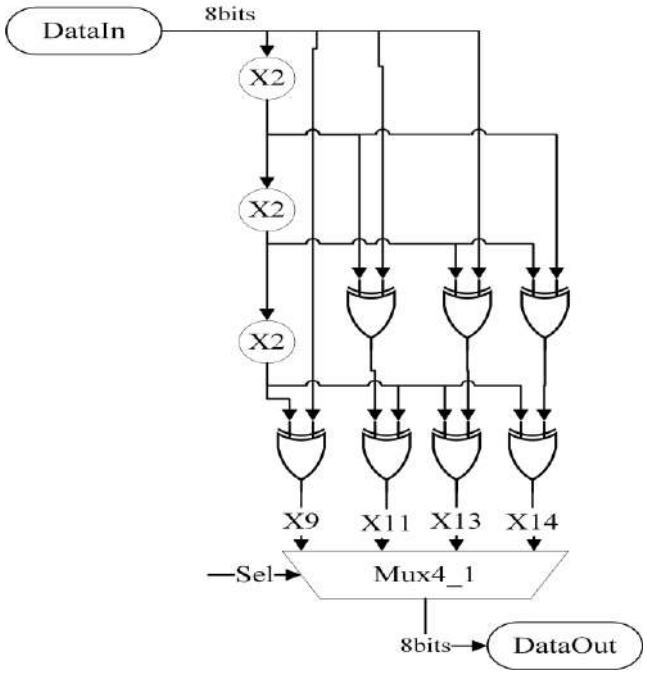
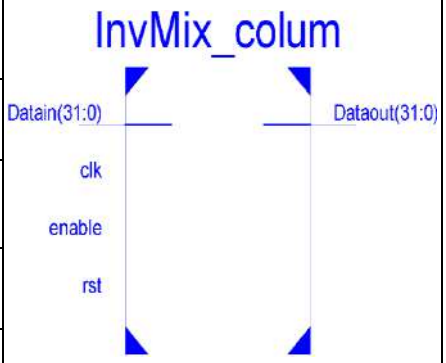



***Figure II.15 :*** *Schéma bloc de la multiplication dans GF($2^8$) par 9, 11, 13, 14*

***Tableau II.12 :*** *description des signaux Entrées/Sorties du module InvMix_Columns*

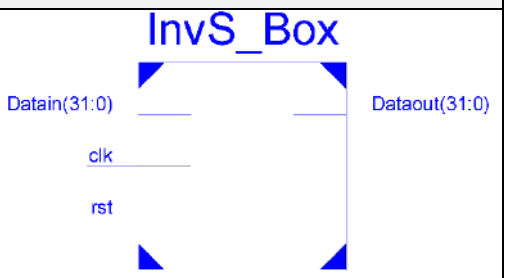
| InvMix_Columns.vhd | | |
|---|---|---|
| Port | Description | Schéma bloc du module |
| Datain | donnée avant l'inverse de mélange des colonnes |  |
| Clk | signal d'horloge (freq = 125 MHZ) | |
| Enable | active/désactive l'opération de Inv*Mix_Column* (si elle est désactive alors la sortie = l'entrée) | |
| Rst | remise à zéro du circuit | |
| Dataout | donnée après l'inverse de mélange des colonnes | |

### II.4.3.2. Le sous-module InvS_Box

Ce module réalise une substitution d'une donnée de 32 bits d'entrée, la donnée est subdivisée en 4 mots de 8 bits. Ce module utilise 4 mémoires ROM de 256 mots de 8 bits pour la substitution des différents mots. Les 8 bits des de chaque entrée servent à sélectionner une case dans la table *InvSbox* (§*I.6.4.2.2.A).*

Les 4 substitutions seront concaténer en sortie pour obtenir la donnée de 32 bits.

***Tableau II.13 :*** *description des signaux Entrées/Sorties du module InvS_Box*

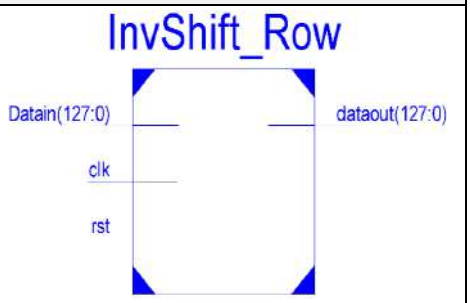
| InvS_Box.vhd | | |
|---|---|---|
| Port | Description | Schéma bloc du module |
| Datain | donnée avant l'inverse de la substitution |  |
| Clk | signal d'horloge (freq = 125 MHZ) | |
| Rst | remise à zéro du circuit | |
| Dataout | donnée après l'inverse de la substitution | |



### II.4.3.3. Le sous-module InvShift_Row

Ce module dispose d'une entrée de 128 bits, il fait des rotations sur les lignes de la matrice d'entrée, le résultat de sortie est de 128 bits (§ *I.6.4.2.2.B).*

***Tableau II.14 :*** *description des signaux Entrées/Sorties du module InvShift_Row*

| InvShift_Row.vhd | | |
|---|---|---|
| Port | Description | Schéma bloc du module |
| Datain | donnée avant l'inverse de décalage |  |
| Clk | signal d'horloge (freq = 125 MHZ) | |
| Rst | remise à zéro du circuit | |
| Dataout | donnée après l'inverse de décalage | |

### II.4.3.4. Le sous-module Dec_Controller

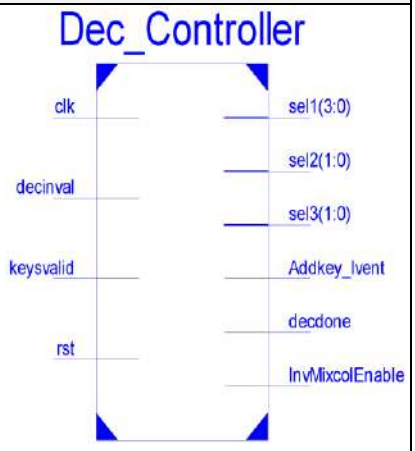
Ce contrôleur fait la synchronisation des différents modules de décryptage : *AddRoundKey*, *Inv*S-*Box*, et *InvMixColumns*. Il permet de lancer le décryptage pour une nouvelle donnée lorsque les clés sont prêtes, sélectionner la ronde à exécuter, le mot à substituer, le mot à enregistrer pendant la ronde, l'indication de démarrage de *AddRoundKey*, l'activation de *InvMixColumns*, ainsi que la validation de la sortie.

La table d'ordonnancement (en mode pipeline) des différentes unités de ce contrôleur (séquenceur) est montée dans le tableau II.15.

***Tableau II.15 :*** *La table d'ordonnancement des unités du module de décryptage*

| **Cycles** | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |
|---|---|---|---|---|---|---|---|---|---|---|
| **Unités** | Sel2=0<br>ADD | Sel2=1<br>IMIXC | Sel2=2<br>IMIXC<br>ISBOX<br><br>Sel3=0 | Sel2=3<br>IMIXC<br>ISBOX<br>REG<br>Sel3=1<br>Sel1++ | IMIXC<br>ISBOX<br>REG<br>Sel3=2 | ISBOX<br>REG<br>Sel3=3 | REG | ISHIFT | ISHIFT | AddIv |

***Tableau II.16 :*** *description des signaux Entrées/Sorties du module Dec_Controller*

| Dec_Controller.vhd | | |
|---|---|---|
| Port | Description | Schéma bloc du module |
| Clk | signal d'horloge (freq = 125 MHZ) |  |
| Decinval | le message est prêt à être décrypter | |
| Keysvalid | clés des rondes sont prêtes | |
| Rst | remise à zéro du circuit | |
| Sel1 | sélecteur de ronde | |
| Sel2 | sélecteur de *mux* avant *InvMixColumns* | |
| Sel3 | sélecteur de *demux* après *InvS-Box* | |
| AddKey_Ivent | effectuer *AddRoundKey* | |
| Decdone | donnée décryptée est valide | |
| InvMixcolEnable | active *InvMixColumn* | |



**Remarque :** les autres modules de décryptage (*Reg32_128*, *Key_Gen* et *AddRoundKey*) sont identiques à celle du circuit de cryptage.

### II.4.4. Architecture du module de communication RS232

Pour vérifier le bon fonctionnement de notre implémentation du système de cryptage/décryptage sur le circuit FPGA, nous avons développé une simple interface de communication série permettant au circuit de recevoir et de transmettre les données (message en clair ou chiffré) entre le PC et la carte FPGA via le port série (RS232).

La liaison RS232 est de type série asynchrone, c'est à dire qu'elle ne nécessite pas de signal d'horloge pour la synchronisation des deux systèmes en communication. Il est donc nécessaire que ces derniers soient configurés de la même manière, seulement trois pins du port série sont utilisés :

- Le pin 2 : c'est le pin par laquelle la carte transmet des données.
- Le pin 3 : c'est le pin par laquelle la carte reçoit des données.
- Le pin 5 : masse commune.

Pour assurer une meilleure synchronisation entre les deux systèmes, leur configuration doit être identique. Elle est basée sur : le débit du transfert, la taille de la donnée et quelques bits de contrôle. Les débits de transmission les plus courants sont : 300, 600, 1200, 4800, 9600,19200, 38400, 57600 et 115200 bps (bits par second). Le transfert de l'extérieur vers la carte ou vice-versa se fait par l'envoi d'un paquet de 7 ou 8 bits en série et auxquels sont rajoutés quelques bits de contrôle. La totalité de ces bits constituent une trame de donnée **[9].**

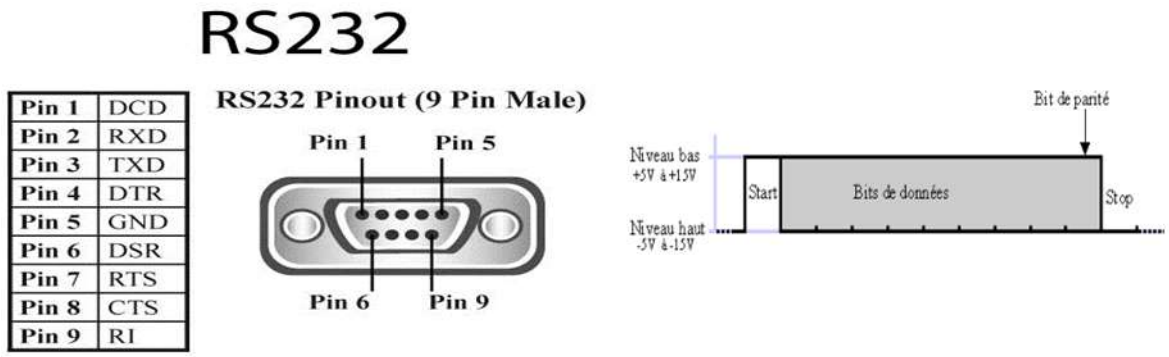



***Figure II.16 :*** *La liaison série RS-232*

**Remarque**

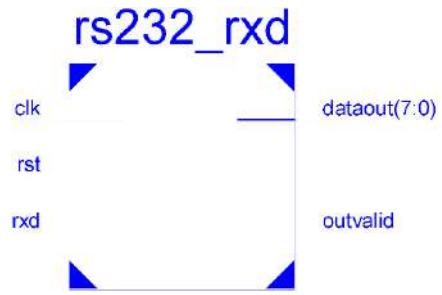
Pour notre module nous avons utilisé la configuration suivante :

Le débit de transfert = 115200 bps ; la taille de la donnée = 8 bits ; pas de bits de parité ; 1 bit stop.

### II.4.4.1. Le sous-module Rs232_rxd

A travers le port série, les données sont reçues en série. Après le bit start, chaque bit reçu via l'entrée *rxd* est mémorisé dans un registre de 8 bits. Dès qu'il est reçu 8 bits, un indicateur de sortie est valide.

***Tableau II.17 :*** *description des signaux Entrées/Sorties du module Rs232_rxd*

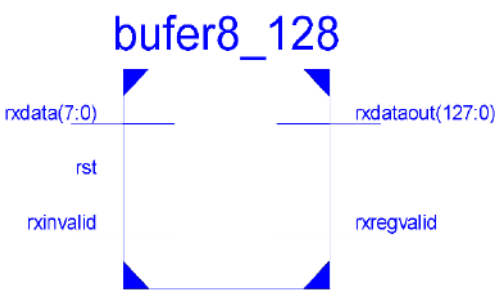
| Rs232_rxd.vhd | | |
|---|---|---|
| Port | Description | Schéma bloc du module |
| Clk | signal d'horloge (freq = 125 MHZ) |  |
| Rst | remise à zéro du circuit | |
| Rxd | signal reçu (vers le pin rxd) | |
| Dataout | 8 bits reçues par RS232 | |
| Outvalid | données reçues prêtes | |



### II.4.4.2. Le sous-module Bufer8_128

Ce module permet la mémorisation des trames de 8 bits reçu par le module *Rs232_rxd* dans un registre de 128 bits. Cette donnée de 128 bits est utilisée par les traitements de cryptage/décryptage. Ce module joue le rôle d'un démultiplexeur 1-16 avec mémorisation.

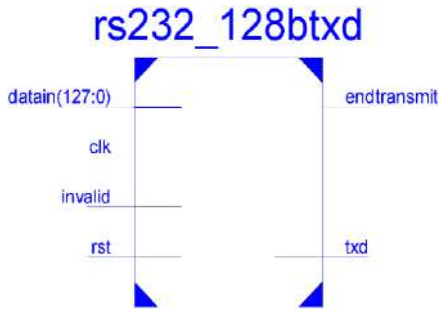
***Tableau II.18 :*** *description des signaux Entrées/Sorties du module Bufer8_128*

| Bufer8_128.vhd | | |
|---|---|---|
| Port | Description | Schéma bloc du module |
| Rxdata | 8 bits de données reçues par RS232 |  |
| Rst | remise à zéro du circuit | |
| RxInvalid | les données d'entrée sont prêtes | |
| RxDataout | 128 bits pour le cryptage/décryptage | |
| RxRegvalid | les données en sortie sont prêtes | |

### II.4.4.3. Le sous-module Rs232_128btxd

La donnée de 128 bits obtenue après le traitement de cryptage et/ou décryptage est transmise à travers l'interface RS232 vers le PC en utilisant la sortie RS232 *Txd*.

***Tableau II.19 :*** *description des signaux Entrées/Sorties du module Rs232_128btxd*

| Rs232_128btxd.vhd | | |
|---|---|---|
| Port | Description | Schéma bloc du module |
| Datain | 128 bits de données à transmettre |  |
| Clk | signal d'horloge (freq = 125 MHZ) | |
| Invalid | les données d'entrée sont prêtes | |
| Rst | remise à zéro du circuit | |
| Endtrans | fin de transmission | |
| Txd | signal de transmission (vers le pin txd) | |



## II.5. Conclusion

Dans ce chapitre, nous avons donné une description détaillée de l'architecture du circuit de Cryptage et de Décryptage AES développé pendant ce travail. Pour cela, nous avons données une description du circuit globale, du module de cryptage, du module de décryptage, ainsi que leurs sous-modules. En fin une description de l'interface de communication RS232.

# Chapitre III

# Résultats Pratiques

Sommaire

III.1. Introduction ..... 53
III.2. Simulation temporelle des différents modules de l'algorithme AES ..... 53
III.2.1. Module Key_Gen ..... 54
III.2.2. Module AddRoundKey ..... 54
III.2.3. Module S-Box ..... 55
III.2.4. Module InvS-Box ..... 55
III.2.5. Module ShiftRow ..... 56
III.2.6. Module InvShiftRow ..... 56
III.2.7. Module MixColumns ..... 57
III.2.8. Module InvMixColumns ..... 57
III.2.9. Module Data_Encryption ..... 58
III.2.10. Module Data_Decryption ..... 58
III.2.11. Simulation de l'architecture pipeline ..... 59
III.3. Résultats de synthèse et d'implémentation du circuit développé ..... 59
III.3.1. Rapport après la synthèse et l'implémentation ..... 59
III.3.2. Comparaison des résultats après l'implémentation du circuit développé avec d'autres circuits réalisés ..... 60
III.3.3. les schémas RTL générés après la synthèse ..... 60
III.4. Résultats de test pratique sur la carte FPGA ..... 63
III.5. Conclusion ..... 64

## III.1. Introduction

Après la description de l'architecture du circuit de cryptage/décryptage AES développé, maintenant ce chapitre traite les résultats de simulation temporelle et de synthèse de ce circuit, ainsi que les tests pratiques. Il sera présenté dans ce chapitre :

- Les différents résultats de simulation temporelle des codes VHDL pour tous les modules du circuit réalisé par l'outil de simulation ISim ainsi que les schémas RTL des principaux modules (*Data_Encryption*, *Data_Decryption* et le module global).
- Une fois que la vérification fonctionnelle est faite, la conception sera prise par l'outil ISE Design Suite 14.1de Xilinx pour le processus de synthèse. Les résultats de synthèse seront présentés par des rapports détaillés sur la consommation du circuit AES sur FPGA.
- Les résultats du test pratique du circuit AES implémenté sur la carte FPGA.

## III.2. Simulation temporelle des différents modules de l'algorithme AES

La simulation temporelle des différents modules est faite à l'aide du simulateur ISim qui est intégré dans l'outil XILINX ISE Design Suite 14.1. La simulation temporelle est exécutée pour vérifier le fonctionnement de la conception. C'est une simulation *Post Route*, c'est-à-dire, elle est été réalisée après les étapes de la synthèse et de l'implémentation (après les interconnexions des différents blocks logiques) qui implique que cette simulation prend en considération les contraintes temporelles.

Les simulations des différents modules : *AddRoundKey, S-Box, InvS-Box, ShiftRow, Inv ShiftRow, MixColumns et InvMixColumns* représente la 1$^{ière}$ ronde et validées avec le document officiel Fips-197**[6]** (Annexe B).

Afin de valider les quatre unités *AddRoundKey, S-Box, ShiftRow, et MixColumns,* les données utilisées sont prises d'une manière à injecter la donnée de sortie d'une unité dans l'entrée de l'unité suivante. En plus, nous avons pris la donnée de sortie d'un module comme donnée d'entrée du module inverse pour valider les unités *InvS-Box, InvShiftRow, et InvMixColumns.*

La simulation a été faite en prenant le circuit SPARTAN 3A DSP (xc3sd1800a ; package fg676 ; vitesse -4) et avec une horloge de 125 MHZ.

### III.2.1. Module Key_Gen

Puisque les clés sont liées entre eux, alors il est suffisant d'illustré dans cette simulation la clé initiale *key0* et la dernière clé de la 10ème ronde (si la 10ème clé est correcte alors les autres sont aussi correctes).

Avant de générer les dix clés, il faut que le signal *invalid* passe à 1. Après la génération de toutes les clés le signal *keysvalid* envoi le bit 1.

Alors la clé initiale utilisée dans cette simulation est :

*key0* = 0x2b7e151628aed2a6abf7158809cf4f3c

La clé du 10ème rondes obtenue est *keyoutr* = 0xd014f9a8c9ee2589e13f0cc8b6630ca6

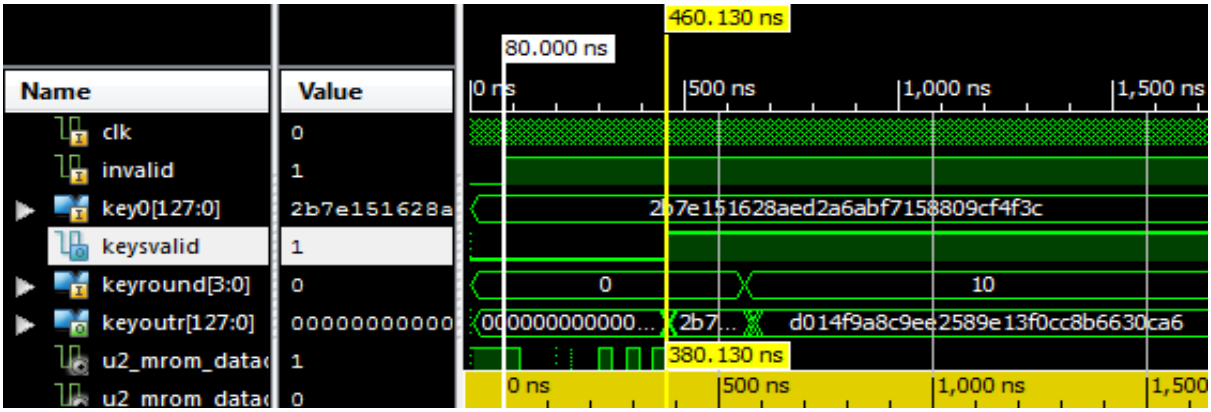



***Figure III.1 :*** *Simulation temporelle du module Key_Gen*

### III.2.2. Module AddRoundKey

La donnée d'entrée : *datain* = 0x3243f6a8885a308d313198a2e0370734

Et la clé : *key* = 0x2b7e151628aed2a6abf7158809cf4f3c

L'opération est effectuée lorsque le *rst* prend 0 et *invalid* passe de 0 à 1. Le résultat (figure III.2) de cette opération est :

*dataout* = 0x193de3bea0f4e22b9ac68d2ae9f84808 (même valeur donnée en Aneexe B).

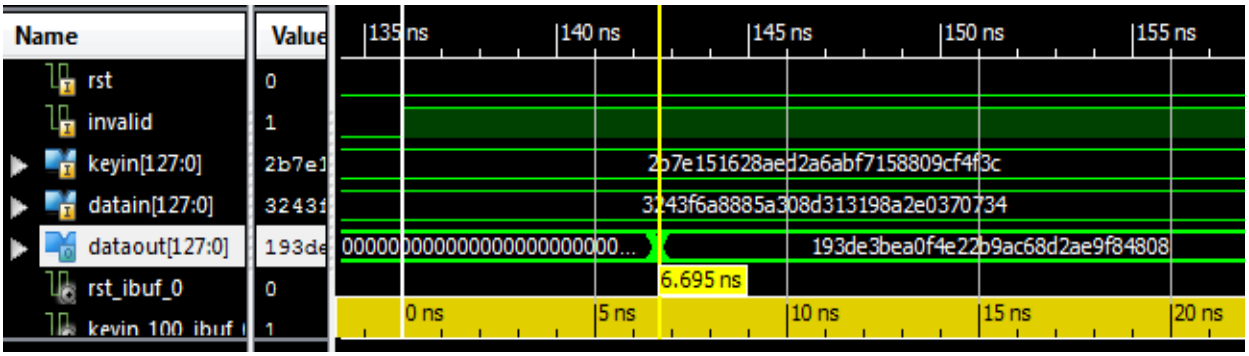



***Figure III.2 :*** *Simulation temporelle du module AddRoundKey*

### III.2.3. Module S-Box

La donnée à substituer est : 0x193de3bea0f4e22b9ac68d2ae9f84808. Cette donnée d'entrée est divisée sur des mots de 32 bits :

*datain*= 0x193de3be (1ier cycle), *datain*= 0xa0f4e22b (2ième cycle),

*datain*= 0x9ac68d2a (3ième cycle), *datain*= 0xe9f84808 (4ième cycle).

L'opération est effectuée lorsque *rst* est mise à 0. Le résultat de cette opération est :

*dataout*= 0xd42711ae (1ier cycle), *dataout*= 0xe0bf98f1 (2ième cycle),

*dataout*= 0xb8b45de5 (3ième cycle), *dataout*= 0x1e415230 (4ième cycle).

La sortie après les quatre *S-Box* est : 0xd42711aee0bf98f1b8b45de51e415230 (même valeur donnée en Aneexe B).

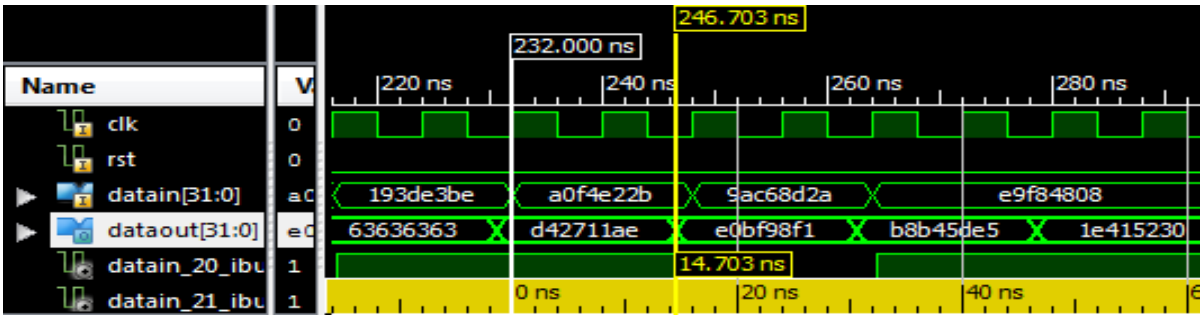



***Figure III.3 :*** *Simulation temporelle du module S-Box*

### III.2.4. Module InvS-Box

La donnée d'entrée : 0xd42711aee0bf98f1b8b45de51e415230

Cette donnée d'entrée est divisée sur des mots de 32 bits :

*datain*= 0xd42711ae (1ier cycle), *datain*= 0xe0bf98f1 (2ième cycle),

*datain*= 0xb8b45de5 (3ième cycle), *datain*= 0x1e415230 (4ième cycle).

L'opération est effectuée lorsque *rst* est mise à 0. Le résultat de cette opération est :

*dataout* = 0x193de3be (1ier cycle), *dataout* = 0xa0f4e22b (2ième cycle),

*dataout* = 0x9ac68d2a (3ième cycle), *dataout* = 0xe9f84808 (4ième cycle).

La sortie après quatre *InvS-Box* est : 0x193de3bea0f4e22b9ac68d2ae9f84808 (même valeur d'entrée de *S_Box*)

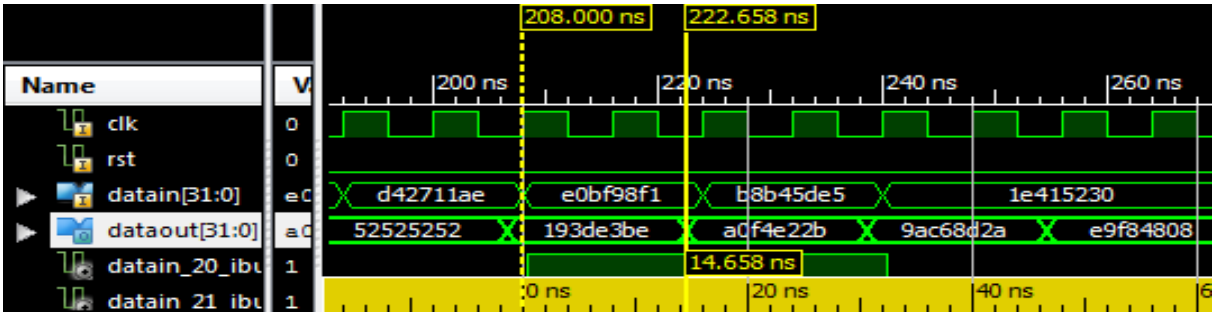



***Figure III.4 :*** *Simulation temporelle du module InvS-Box*

On remarque que la valeur de la donnée obtenue par ce module est la même que la valeur de la donnée d'entrée du module *S-Box*, ceci est justifié parce que ce deuxième module est l'inverse du premier.

### III.2.5. Module ShiftRow

La donnée d'entrée : *datain*= 0xd42711aee0bf98f1b8b45de51e415230.

L'opération est effectuée lorsque *rst* prend 0. La donnée de sortie est la suivante :

*dataout* = 0xd4bf5d30e0b452aeb84111f11e2798e5 (même valeur donnée en Aneexe B).

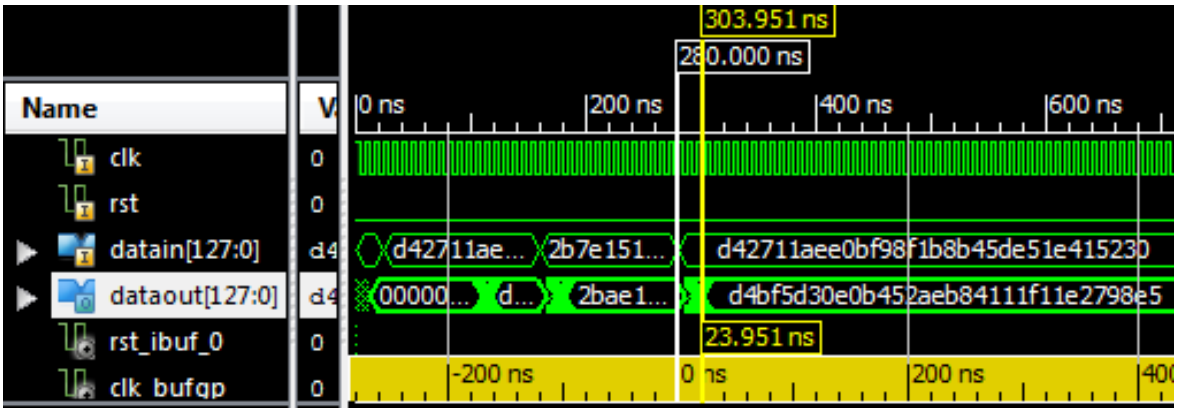



***Figure III.5 :*** *Simulation temporelle du module ShiftRow*

### III.2.6. Module InvShiftRow

La donnée d'entrée : *datain*= 0xd4bf5d30e0b452aeb84111f11e2798e5.

L'opération est effectuée lorsque *rst* prend 0. La donnée de sortie est la suivante :

*dataout* = 0xd42711aee0bf98f1b8b45de51e415230 (même valeur d'entrée de *shiftRows*)

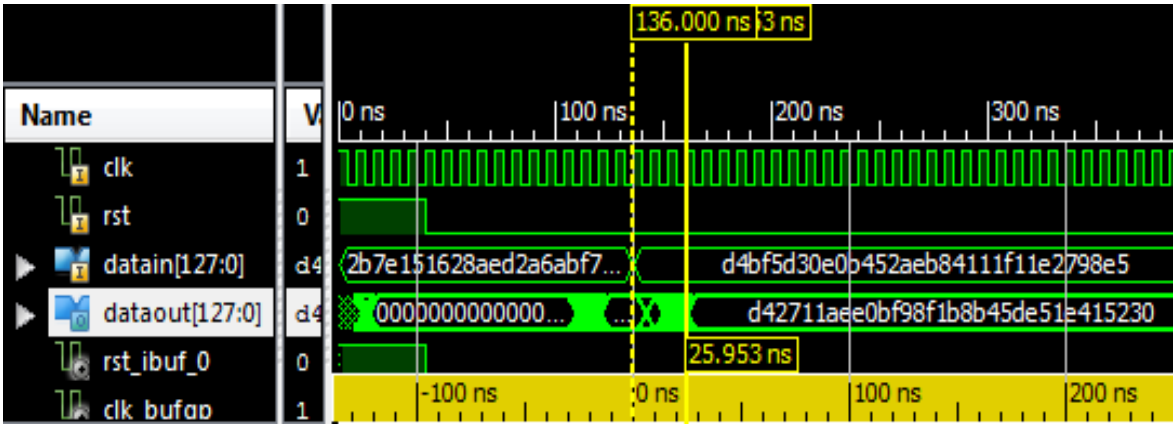



***Figure III.6 :*** *Simulation temporelle du module InvShiftRow*

### III.2.7. Module MixColumns

L'opération du *MixColumns* s'opère sur chaque 32 bits de la donnée complet de 128 bits. Les quatre mots d'entrées sont :

*datain* = 0xd4bf5d30, *datain* = 0xe0b452ae, *datain* = 0xb84111f1, *datain* = 0x1e2798e5.

Le signal *enable* est mis à 1 et le *rst* à 0. Le résultat de cette opération est :

*dataout*= 0x046681e5, *dataout*= 0xe0cb199a, *dataout* = 0x48f8d37a, *dataout*=0x2806264c.

La donnée de sortie est : 0x046681e5e0cb199a48f8d37a2806264c (même valeur donnée en Aneexe B).




***Figure III.7 :*** *Simulation temporelle du module MixColumns*

### III.2.8. Module InvMixColumns

L'opération du *InvMixColumns* s'opère sur chaque 32 bits de la donnée complet de 128 bits. Les quatre mots d'entrées sont les suivants :

*Datain* = 0x046681e5, *datain* = 0xe0cb199a, *datain* = 0x48f8d37a, *datain* = 0x2806264c.

Le signal *enable* est mis à 1 et le *rst* à 0. Le résultat de cette opération est :

*dataout*= 0xd4bf5d30, *dataout*= 0xe0b452ae, *dataout*= 0xb84111f1, *dataout*= 0x1e2798e5.

La donnée de sortie est : 0xd4bf5d30e0b452b84111f11e2798e5 (même valeur d'entrée de *Mixcolumns*).

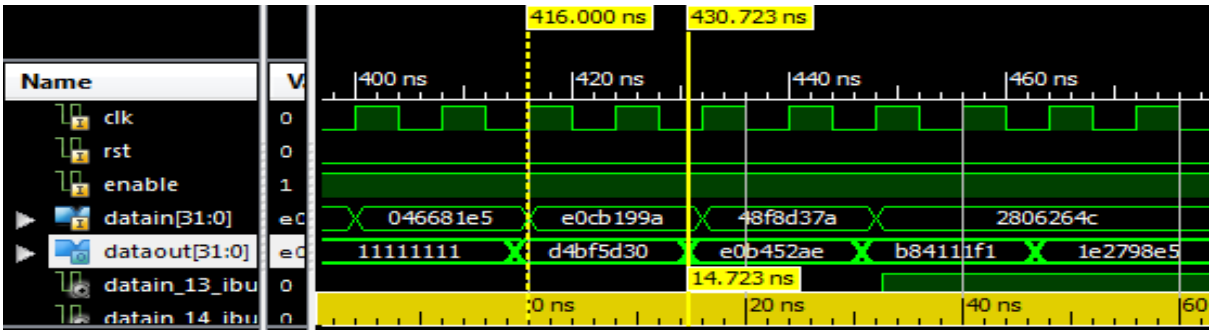



***Figure III.8 :*** *Simulation temporelle du module InvMixColumns*

### III.2.9. Module Data_Encryption

Les entrées sont comme suit :

Message clair : *datain* = 0x3243f6a8885a308d313198a2e0370734 (même valeur donnée en Aneexe B).

La clé : *key* = 0x2b7e151628aed2a6abf7158809cf4f3c

Le signal *keyinvalid* prend la valeur 1, rst la valeur 0 et *datainvalid* passe de 0 à 1. Dès que le signal *outvalid* passera de 0 à 1, le chiffrage est fait.

La donnée de sortie (message chiffré) obtenue est comme suit :

*dataout* = 0x3925841d02dc09fbdc118597196a0b32 (même valeur donnée en Aneexe B).

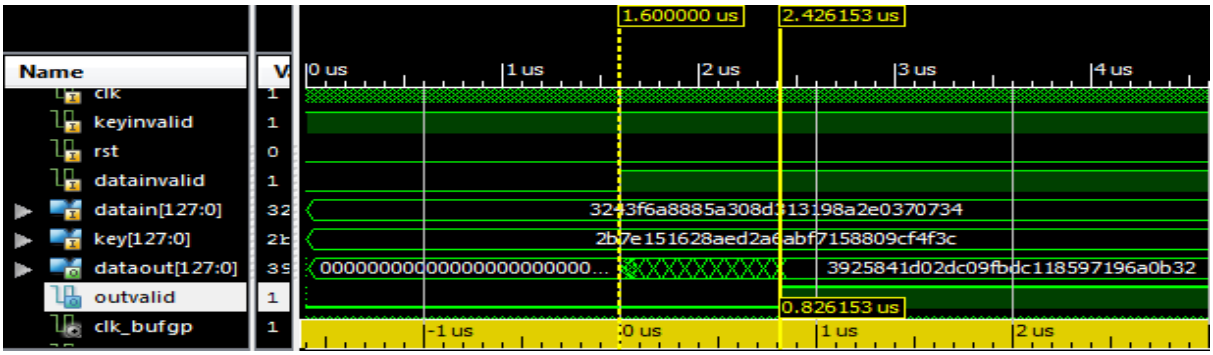



***Figure III.9 :*** *Simulation temporelle du module Data_Encryption*

### III.2.10. Module Data_Decryption

Les entrées sont comme suit :

*datain*= 0x3925841d02dc09fbdc118597196a0b32 (résultat de la simulation du module de cryptage).

*key* = 0x2b7e151628aed2a6abf7158809cf4f3c (même clé de cryptage).

Le signal *keyinvalid* prend la valeur 1, rst la valeur 0 et *datainvalid* passe de 0 à 1. Dès que le signal *outvalid* passera de 0 à 1, le déchiffrage est fait.

La donnée de sortie (message déchiffré) obtenue est comme suit :

*dataout*= 0x3243f6a8885a308d313198a2e0370734 (même valeur que la donnée claire).

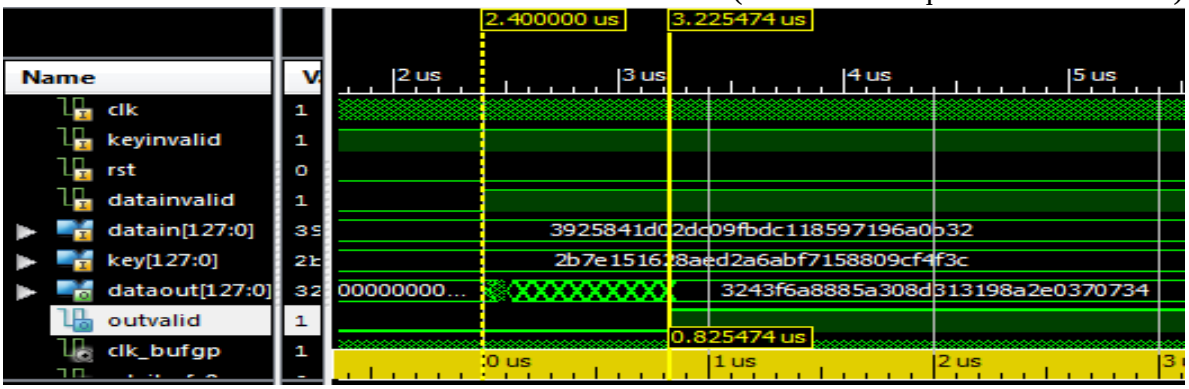



***Figure III.10 :*** *Simulation temporelle du module Data_Décncryption*

### III.2.11. Simulation de l'architecture pipeline

L'architecture en mode pipeline c'est pour l'utilisation des unités *s_box*, *mixcolumn* et *reg32_128* (pour le module *Data_Encryption*) et inv_*mixcolumn*, invs*_box* et *reg32_128* (pour le module *Data_décryption*) en même temps pour augmenter la vitesse de cryptage, les tables d'ordonnancement utilisés sont présentés dans le tableau II.9 et le tableau II.15.

La figure III.11 montre le résultat du fonctionnement en mode pipeline.

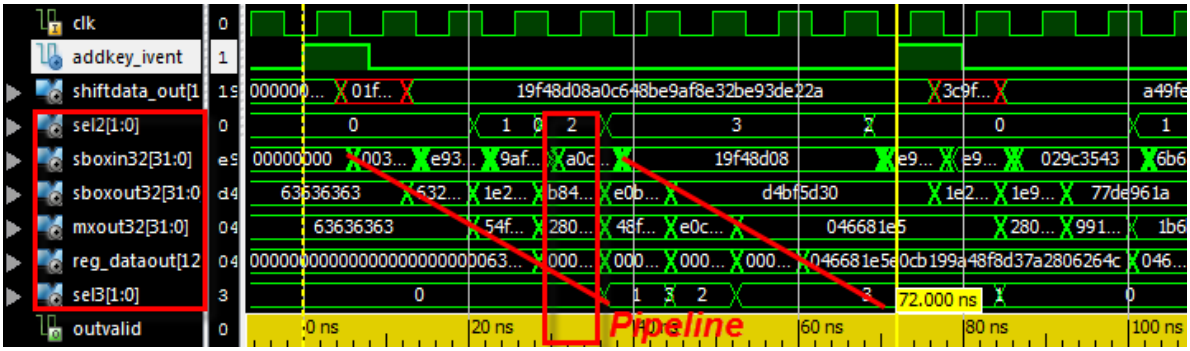



***Figure III.11 :*** *Simulation de l'architecture pipelines.*

## III.3. Résultats de synthèse et d'implémentation du circuit développé

Les fonctionnalités du circuit développé de chiffrage et de déchiffrage ont été simulées et vérifiées dans la partie précédente. La conception du circuit de chiffrage et de déchiffrage AES est synthétisée et ses résultats sont présentés dans le tableau III.1.

Dans le but de faire l'implémentation sur un FPGA, la synthèse a été faite en prenant le circuit SPARTAN 3A DSP (xc3sd1800a ; package fg676 ; vitesse -4) et avec une horloge de 125 MHZ.

### III.3.1. Rapport après la synthèse et l'implémentation

Il y a plusieurs rapports qui sont produits pendant le processus de la synthèse. Il est impératif d'examiner ces rapports pour vérifier l'opérabilité du circuit synthétisé.

Le rapport détaillé des résultats concernant l'occupation de surface et le temps nécessaire pour les différents modules de l'architecture (en mode pipeline) est comme le montre le tableau III.1.

***Tableau III.1**: Les résultats de l'implémentation des différents modules*

| Modules | Slices | 4-LUT | Flip-Flops | G-CLK | Temps |
|---|---|---|---|---|---|
| **Data_Encryption** | **1171** | **2016** | **801** | **3** | **826 ns (103 cycles)** |
| Enc_Controller | 17 | 32 | 15 | 2 | < 8 ns (1 cycle) |
| Key_Gen | 665 | 1225 | 336 | 1 | 380 ns (47 cycles) |
| S_Box | 256 | 512 | 32 | 1 | <8 ns (1 cycle) |
| Mix_Col | 40 | 76 | 32 | 1 | <8 ns (1 cycle) |
| Shift_Row | 66 | / | 128 | 1 | <16 ns (2 cycles) |
| RotWord_ForKey | 18 | / | 32 | 0 | <8 ns (1 cycle) |
| AddRoundKey | 74 | 128 | / | 1 | <8 ns (1 cycle) |
| Reg32_128 | 66 | / | 128 | 1 | <8 ns (1 cycle) |
| **Data_Decryption** | **1249** | **2106** | **798** | **3** | **825 ns (103 cycles)** |
| Dec_Controller | 12 | 21 | 17 | 2 | <8 ns (1 cycle) |
| InvS_Box | 256 | 512 | 32 | 1 | <8 ns (1 cycle) |
| InvMix_Col | 87 | 173 | 32 | 1 | <8 ns (1 cycle) |
| InvShift_Row | 68 | / | 128 | 1 | <16 ns (2 cycles) |

### III.3.2. Comparaison des résultats après l'implémentation du circuit développé avec d'autres circuits réalisés

Le tableau III.2 montre une comparaison entre notre circuit et d'autres circuits développés par d'autres personnes dans ce domaine.

***Tableau III.2** : Comparaison du circuit développé avec d'autres circuits réalisés*

| | Dispositif | Algo | Clé (bits) | Slice | Vitesse (Mb/s) | Performance (Vitesse/Slice) |
|---|---|---|---|---|---|---|
| Notre architecture | (virtex-6) xc6vlx240t | Enc<br>Dec | 128<br>128 | 515<br>459 | 248*<br>248* | 0.48<br>0.54 |
| | Spartan 3A dsp xc3sd1800a-4fg676 | Enc<br>dec | 128<br>128 | 1171<br>1249 | 155<br>155 | 0.13<br>0.12 |
| P. Chodowiec et al[15] | Virtex-6 | Enc/ Dec | 128/192/ 256 | 1230 | 577 | 0.47 |
| A. Dandalis et al[17] | Virtex-6 | Enc/ Dec | 128 | 5673 | 353 | 0.06 |
| A.J. Elbirt et al[16] | Virtex-6 | Enc | 128 | 3528 | 294.2 | 0.08 |
| Pawel Chodowiec et Kris Gaj[18] | SpartanII-6 | Enc/ dec | 128 | 222 | 166 | 0.75 |
| K.Gaj et al [19] | Virtex-6 | Enc/ Dec | 128 | 2902 | 331.5 | 0.11 |

* le résultat est calculé pour le virtex-6 LX240T, l'horloge est de 200 Mhz (5 ns/cycle), donc pour 103 cycle on à 515 ns, ce qui donne une vitesse de cryptage/décryptage = $\frac{128*10^9}{515}$ = 248 Mb/s.

### III.3.3. les schémas RTL générés après la synthèse

Les schémas RTL des principaux modules *Data_Encryption et Data_Decryption* sont représentés dans la figure III.12, et le module global (*top*) dans la figure III.13.

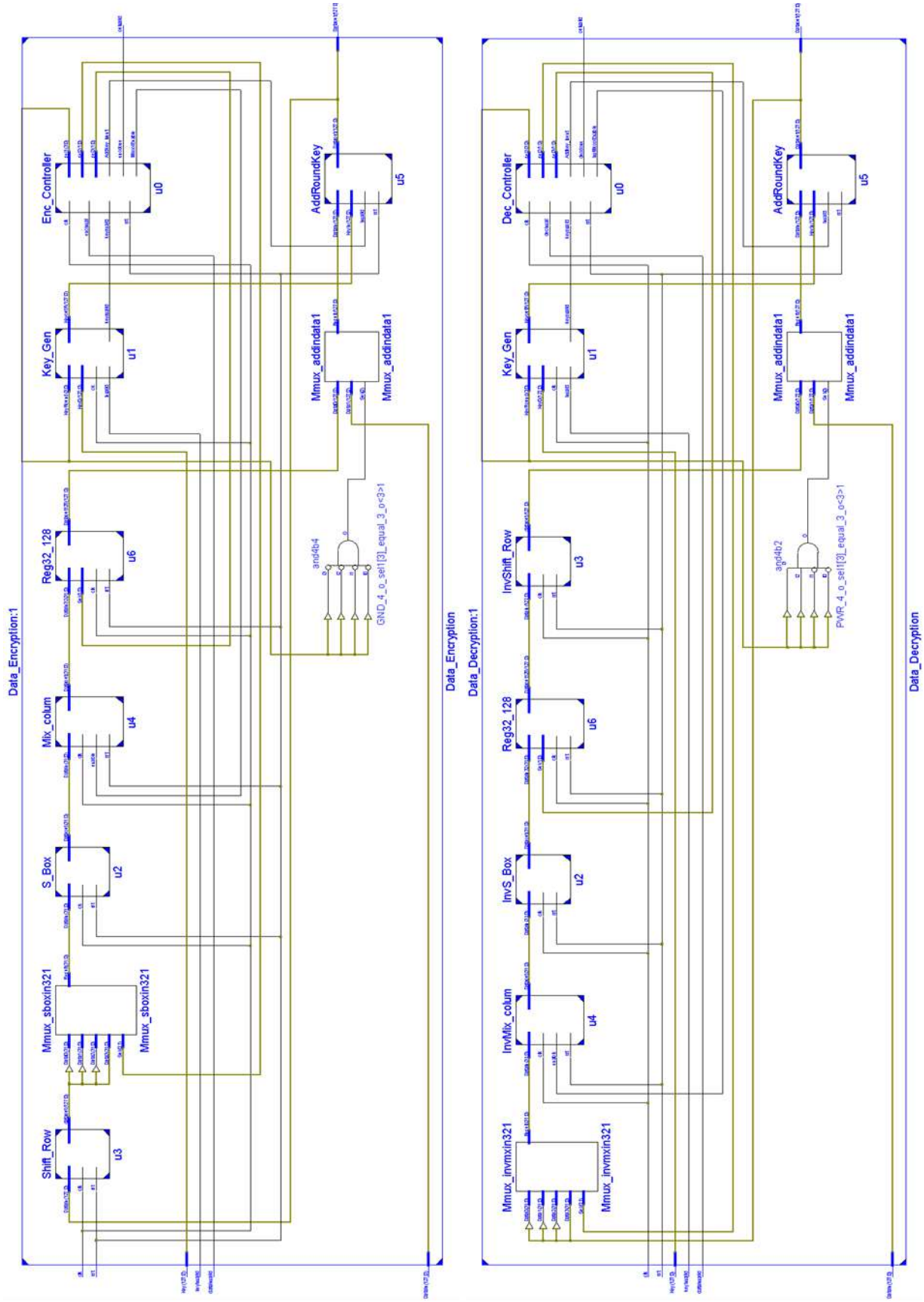



***Figure III.12 :*** *Les schémas RTL des modules Data_Encryption et Data_Decryption*

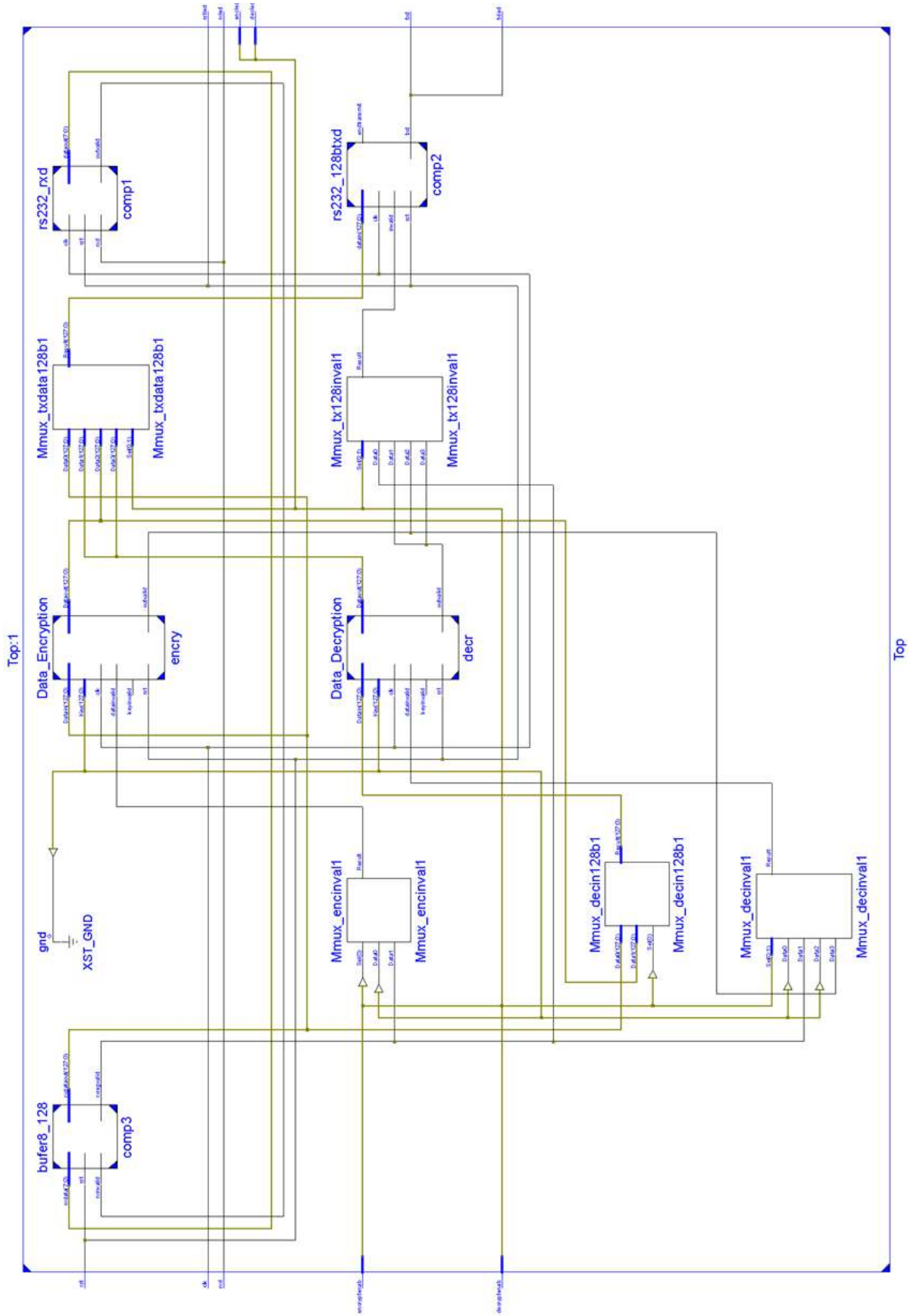



***Figure III.13 :*** *Le schéma RTL du circuit développé global*

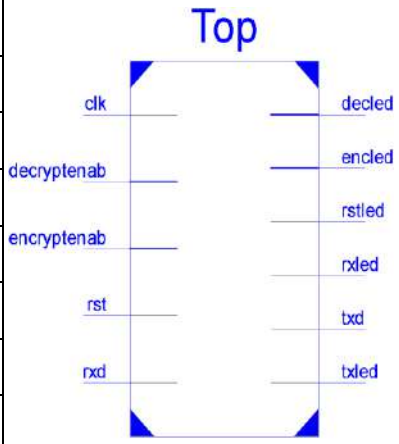
## III.4. Résultats de test pratique sur la carte FPGA

Pour faire le test du circuit développé sur la carte FPGA, les outils utilisés sont les suivants :

- La carte FPGA.
- Un ordinateur.
- Un câble liaison série RS232 non croisé.
- Un hyper Terminal HERCULES 3.2.5 pour l'envoi et la réception des données en hexadécimal, et un autre (REALTERM 2.0) pour l'envoi et la réception des fichiers via le port série RS232.

La configuration choisie des pins pour tester le circuit sur la carte est montrée dans le tableau III.3.

***Tableau III.3 :*** *La configuration des pins de la carte FPGA*

| Les signaux | Description | Pin | L'entité |
|---|---|---|---|
| clk | L'horloge (freq = 125 MHZ) | F13 |  |
| Decryptenab | Activer/Désactiver décryptage | G16 | |
| Encryptenab | Activer/Désactiver cryptage | A7 | |
| Rst | Remise à zéro du circuit | J17 | |
| Rxd | Donnée reçue à travers RS232 | N21 | |
| Txd | Donnée transmise à travers RS232 | P22 | |
| Decled | LED indique le décryptage est activé | D25 | |
| Encled | LED indique le cryptage est activé | D24 | |
| Rstled | LED de remise à zéro du circuit | K22 | |
| Rxled | LED indique réception des données | P18 | |
| Txled | LED indique transmission des données | P25 | |



Pour assurer la synchronisation entre l'ordinateur et la carte, nous avons utilisés la configuration suivante :

Le débit de transfert = 115200 bps ; la taille de la donnée = 8 bits ; pas de bits de parité ; 1 bit stop.

Et pour valider le fonctionnement de cryptage et de décryptage nous avons envoyés le message clair (Le même exemple du document officiel Fips-197**[6]** (Annexe B)) :

0x3243f6a8885a308d313198a2e0370734

Le message crypté reçu : 0x3925841d02dc09fbdc118597196a0b32

Après nous avons envoyés le résultat de cryptage et nous avons reçu le message décrypté identique au premier message clair.

La clé de test utilisée : 0x2b7e151628aed2a6abf7158809cf4f3c

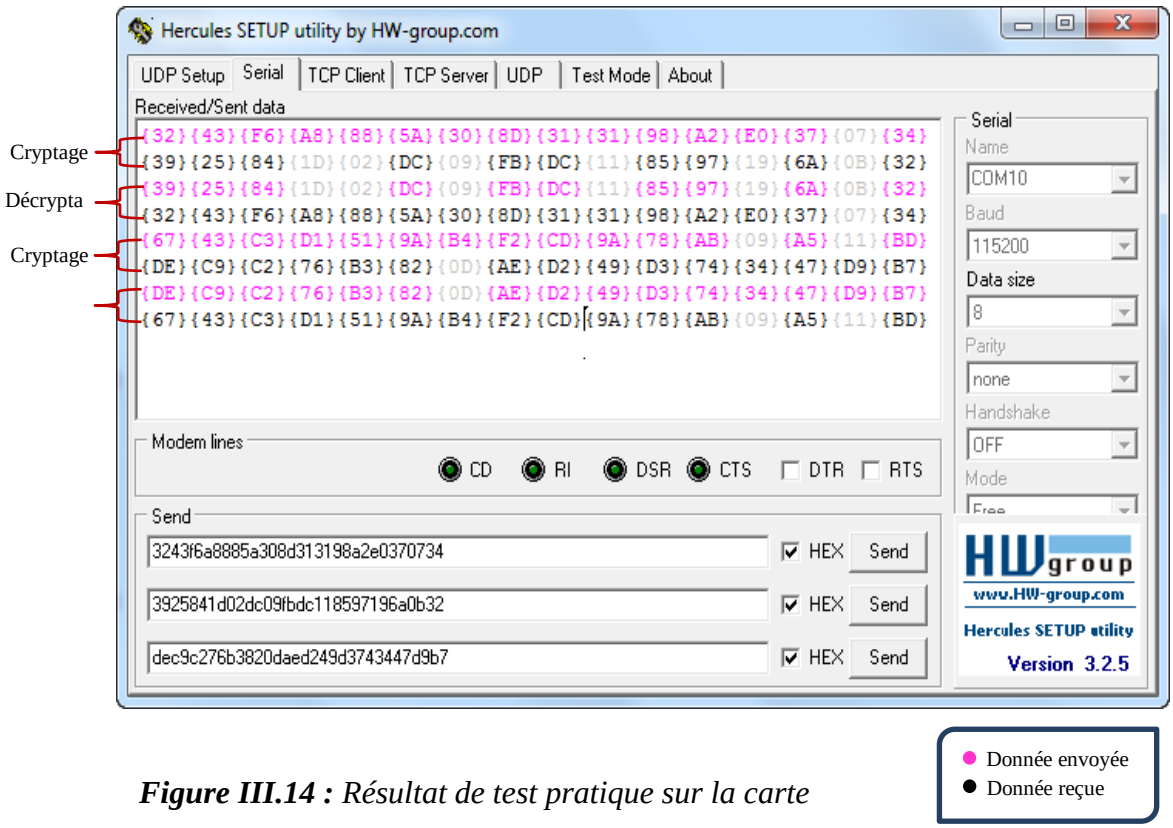



***Figure III.14 :*** *Résultat de test pratique sur la carte*

## III.5. Conclusion

Notre conception pour le chiffrage/déchiffrage avec l'algorithme AES 128 bits, a été simulée, synthétisée, et implémentée en utilisant la carte SPARTAN 3A DSP XC3SD1800A.

En plus, le circuit développé a été examiné pratiquement, les résultats du test sont entièrement fonctionnels. Le circuit de l'algorithme de cryptage et de décryptage AES a atteint une période du temps de 826 ns avec une vitesse de 155 Mb/s, et on peut augmenter jusqu'à 248 Mb/s avec le circuit virtex-6.

# Conclusion générale

Dans ce projet, nous avons développé une conception d'architecture de l'algorithme de cryptage et décryptage AES de Rijndael en langage VHDL. L'utilisation des systèmes reconfigurables est une solution qui permet la validation de façon matérielle de nouveaux principes de sécurité.

Notre conception a été validée, les résultats de simulation sont vérifiés, les différents modules sont entièrement fonctionnels. Le circuit développé est pris par l'outil de Xilinx et enfin fait l'exécution en utilisant le circuit FPGA SPARTAN 3A DSP XC3SD1800A. Les résultats des tests pratiques sont validés avec les exemples du document officiel.

Notre circuit peut chiffrer et déchiffrer des séquences de données jusqu'à 155 Mbps. La vitesse, la fonctionnalité, et le coût de chiffrage et déchiffrage rendent cette solution parfaitement pratique en fournissant un niveau de sécurité satisfaisant pour les systèmes de communications actuels, ou d'autres processus de transfert de données électroniques dont la sécurité est nécessaire.

Et comme perspective, il y a certains aspects de ce projet qui peuvent être explorés à l'avenir. D'autant plus que nous sommes parvenus d'appliquer ce projet sur un réseau de communication, cependant pour manque de temps, nous n'avons pas pu développer cette partie.

# Bibliographie

[1] http://www.final-yearproject.com/, http://www.finalyearthesis.com/ Consulté le 20/02/2013.

[2] Ridene Youssef. *Étude et Implémentation de primitives cryptographiques*, Rapport de stage, Département MIME 2 rue de la liberté 93524 Saint Denis, Université Paris VIII, 2006-2007. 2-5.

[3] RENAUD Dumont. *Introduction à la Cryptographie et à la Sécurité informatique*, Faculté des Sciences Appliquées, Université de Liège, 2006-2007. 46,47, 66-71.

[4] Daniel, Barsky. *Cours de Cryptographie*, Editions Cassini, 34.

[5] Jean-Philippe Gaulier. *Advanced Encryption Standard Présentation de l'AES*, Mémoire de synthèse soumis dans le cadre d'un probatoire en vue de l'acquisition d'un diplôme en Ingénierie Et Intégration Informatique Systèmes d'Information, Conservatoire National des Arts et Métiers Centre Régional Associe de Limoges, 2004. 11, 12, 43.

[6] National Institute of Standards and Technology, Federal Information Processing Standard 197, *The Advanced Encryption Standard (AES)*, P5-34, http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf, November 26, 2001. Télécharger le 7 Février 2013,

[7] Bruno Martin. *Codage, Cryptographie et Applications*, COLLECTION TECHNIQUE ET SCIENTIFIQUE DES TÉLÉCOMMUNICATIONS, PRESSES POLYTECHNIQUES ET UNIVERSITAIRES ROMANDES, 2004. 203.

[8] Hamdan.O.Alanazi, B.B.Zaidan, A.A.Zaidan, Hamid A.Jalab, M.Shabbir and Y. Al-Nabhani. *New Comparative Study Between DES, 3DES and AES within Nine Factors*, JOURNAL OF COMPUTING, VOLUME 2, ISSUE 3, MARCH 2010, ISSN 2151-9617

[9] H.Bessalah, M.Issad, L.Mahiddine, K.Messaoudi, N.Anane, M.Anane. *Implémentation d'un UART sur FPGA pour des Applications en Systèmes Embarqués*, Centre de Développement des Technologies Avancées CDTA, Institut National d'Informatique, Oued Smar, Alger, 2,3.

[10] Daniel Barsky et GhislainDartois. *Cryptographie*, Résumé d'après un cours, Université Paris 13, 2010. 52.

[11] NACHEF Toufik. *Implémentation d'un instrument sur un FPGA*, Mémoire de Magister en vue d'acquisition d'un diplôme en Electronique option Micro-électronique, Département d'Electronique, Université Mouloud Mammeri de Tizi-Ouzou Algérie, 2011. 81-83.

[12] *"Getting Started with the Spartan-3A DSP S3D1800A Starter Platform User Guide"*, UG485, version 1.1, June 12, 2008, p7- 8. www.xilinx.com. Télécharger le 12/01/2013.

[13] "Spartan-3A DSP Starter Platform User Guide" , P46, UG454, version 1.1, January 30, 2009. www.xilinx.com. Télécharger le 10/02/2013.

[14] SOWRIRAJAN SAMPATH. *FPGA based hardware implementation of Advanced Encryption Standard*, A thesis of Master of Science in Engineering, University of Mumbai, India, 2007. 6, 7.

[15] Chodowiec P., Gaj K., Bellows P., Schott B. (2001). *Experimental Testing of the Gigabit IPSec-Compliant Implementations of Rijndael and Triple DES Using SLAAC-1 FPGA Accelerator Board*, Information Security Conference (ISC 2001), Malaga,Spain.

[16] Elbirt A.J., Yip W., Chetwynd B., Paar C. *An FPGA-based performance evaluation of the AES block cipher candidate algorithm finalists, IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, Volume: 9 Issue: 4. August 2001.

[17] Dandalis A., Prasanna V.K., Rolim J.D. *A Comparative Study of Performance of AES Final Candidates Using FPGAs, Cryptographic Hardware and Embedded Systems Workshop (CHES 2000)*, Worcester, Massachusetts. 2000.

[18] Pawel Chodowiec and Kris Gaj. *Very Compact FPGA Implementation of the AES Algorithm*, George Mason University 4400 University Drive Fairfax, VA 22030 U.S.A, 2003. http://ece.gmu.edu/crypto-text.htm. Télécharger le 20/05/2013

[19] Gaj K. and Chodowiec P.:Comparison of the hardware performance of the AES candidates using reconfigurable hardware, Third Advanced Encryption Standard (AES3) Candidate Conference, New York, 2000

# Annexe A

## A.1. Termes et Définitions [6]

| Termes | Définitions |
|---|---|
| **AES** | Advanced Encryption Standard (Norme avancée de chiffrage) |
| **VHDL** | Very High Speed Integrated Circuit Hardware Description Language |
| **FPGA** | Field Programmable Gate Array |
| **CPLD** | Complex Programmable Logic Device |
| **Rangé** *(Array)* | Une collection énumérée d'entités identiques (par exemple, un vecteur d'octets. *(An enumerated collection of identical entities (e.g., an array of bytes).* |
| **Bit** *(bit)* | Un élément binaire ayant une valeur de 0 ou de 1. *(A binary digit having a value of 0 or 1).* |
| **Bloc** *(block)* | Séquence des bits binaire qui comportent l'entrée, la sortie, numéro de ronde, et la clef ronde. La longueur d'une séquence est le nombre de bits qu'il contient. Des blocs sont également interprétés en tant que vecteurs de bytes. *(Sequence of binary bits that comprise the input, output, State, and Round Key. The length of a sequence is the number of bits it contains. Blocks are also interpreted as arrays of bytes.)* |
| **Octet** *(byte)* | Groupe de Huit bits qui est traité comme entité simple ou comme vecteur de 8 bits individuels. *(A group of eight bits that is treated either as a single entity or as an array of 8 individual bits).* |
| **Chiffrement** *(Cipher)* | Série de transformations qui convertit le texte clair en texte chiffré en utilisant la clé de chiffrement. *(Series of transformations that converts plaintext to cipher-text using the Cipher Key).* |
| **Clé de chiffrement** *(Cipher Key)* | Secret, clef cryptographique qui est utilisée par la routine d'expansion de clé pour générer d'un ensemble de clefs rondes ; peut être décrit comme vecteur rectangulaire de bytes, ayant quatre rangées et colonnes de Nk. *(Secret, cryptographic key that is used by the Key Expansion routine to generate a set of Round Keys; can be pictured as a rectangular array of bytes, having four rows and Nk columns).* |

| | |
|---|---|
| **Texte chiffré** *(Ciphertext)* | Données de sortie du chiffrement ou entrée au déchiffrement. *(Data output from the Cipher or input to the Inverse Cipher).* |
| **Déchiffrement** *(InverseCipher)* | Série inverse de chiffrement de transformations qui convertit le texte chiffré en texte clair en utilisant la clé de chiffrement. *(Series of transformations that converts ciphertext to plaintext using the Cipher Key).* |
| **Expansion de clé** *(Key Expansion)* | Routine utilisé pour générer une série de clefs rondes de la clé de chiffrement. *(Routine used to generate a series of Round Keys from the Cipher Key).* |
| **Texte clair** *(Plaintext)* | Donnée d'entrée au chiffrement ou donnée de sortie du déchiffrement. *(Data input to the Cipher or output from the Inverse Cipher).* |
| **Rijndael** *(Rijndael)* | Algorithme cryptographique spécifique dans cette norme avancée de chiffrage (AES). *(Cryptographic algorithm specified in this Advanced Encryption Standard (AES)).* |
| **Clef ronde** *(Round Key)* | Les clefs rondes sont des valeurs dérivées de la clé de chiffrement en utilisant la routine d'expansion de clé ; elles sont appliquées à l'état dans le chiffrement et le déchiffrement. *(Round keys are values derived from the Cipher Key using the Key Expansion routine; they are applied to the State in the Cipher and Inverse Cipher).* |
| **Etat** *(State)* | Résultat intermédiaire de chiffrement qui peut être décrit comme vecteur rectangulaire d'octets, ayant quatre lignes et Nb columns. *(Intermediate Cipher result that can be pictured as a rectangular array of bytes, having four rows and Nb columns).* |
| **S-box** | Table non linéaire de substitution utilisée dans les plusieurs substitution d'octet transformations et dans la routine d'expansion de clé pour exécuter une substitution d'un pour l'autre d'une valeur d'octet. *(Non-linear substitution table used in several byte substitution transformations and in the Key Expansion routine to perform a one-for-one substitution of a byte value.)* |
| $GF(2^8)$ | Champ fini ou champ de Galois qui contient tous les éléments 0 à $2^8$ |
| **Mot** *(Word)* | Un groupe de 32 bits qui est traité comme entité simple ou comme vecteur de 4 bytes. *(A group of 32 bits that is treated either as a single entity or as an array of 4 bytes).* |

***Tableau A.1 :*** *Glossaire des termes et des acronymes*

## A.2. Paramètres, symboles, et fonctions d'algorithme AES [6]

Les paramètres, les symboles, et les fonctions suivants d'algorithme sont utilisées dans l'algorithme AES:

| ***Paramètres, symboles, et fonctions*** | ***Définition*** |
|---|---|
| ***AddRoundKey*** | Transformation dans le chiffrement et le déchiffrement, dans lesquels une clef ronde est ajoutée à l'état en utilisant une opération de XOR. La longueur d'une clef ronde égale la taille de l'état (i.e., pour Nb=4, la longueur de clef ronde est 128 bits (16 bytes)). |
| ***MixColumns*** | Transformation dans le chiffrement qui prend toutes les colonnes de l'état et mélange leurs données (indépendamment d'une une autre) pour produire des nouvelles colonnes. |
| ***ShiftRows*** | Transformation dans le chiffrement qui traite l'état en décalant cycliquement les trois dernières rangées de l'état par différents ex-centrages. |
| ***SubBytes*** | Transformation dans le chiffrement qui traite l'état en utilisant une table non linéaire de substitution de byte (boîte de substitution) qui opère chacun des bytes d'état indépendamment |
| ***InvMixColumns*** | Transformation dans le déchiffrement qui est l'inverse de ***MixColumns .*** |
| ***InvShiftRows*** | Transformation dans le déchiffrement qui est l'inverse de ***ShiftRows .*** |
| ***InvSubBytes*** | Transformation dans le déchiffrement qui est l'inverse de ***SubBytes .*** |
| ***K*** | La clé de chiffrement. |
| ***Nb*** | Nombre de colonnes (mots de 32-bits) comportant l'état. Nb=4 pour une clé de longueur de 128 bits. |
| ***Nk*** | Nombre de mots à 32 bits comportant la clé de chiffrement. Pour cet algorithme ***Nk*** = 4, 6, ou 8. |
| ***Nr*** | Nombre de ronds, qui est une fonction de ***Nk*** et de ***Nb***: (qui est fixe). Pour cet algorithme AES, ***Nr***= 10, 12, ou 14. |
| **Rcon[ ]** | Vecteur constant ronde de mot. |
| ***RotBytes*** | Fonction utilisée dans la routine l'expansion de clé qui prend un mot de quatre octets et exécute une permutation cyclique. |
| ***SubWord*** | Fonction utilisée dans la routine d'expansion de clé qui prend un mot d'entrée de quatre octets et applique une boîte de substitution à chacun des quatre bytes pour produire un mot de sortie. |

| | |
|---|---|
| ***XOR*** | L'opération OU-Exclusif. |
| $\oplus$ | L'opération OU-Exclusif. |
| $\otimes$ | Multiplication de deux polynômes(chacune avec le degré <4) modulo $\boldsymbol{x^4 + 1}$. |
| ● | Multiplication dans champ fini GF($2^8$). |

***Tableau A.2 :*** *Paramètres, symboles et fonctions.*

## A.3. Formalisme mathématique de base pour l'AES [6]

***1-Byte (octet = 8-bits)***

L'AES est un algorithme orienté "byte", concaténation de 8 bits :

$\boldsymbol{\{b_7, b_6, b_5, b_4, b_3, b_2, b_1, b_0\}}$**.** Les valeurs de ces "bytes" peuvent être interprétées comme des éléments d'un Groupe.

Trois représentations d'une même information :

1. 01100011 en binaire.
2. $\boldsymbol{x^6 + x^5 + x + 1}$ sous forme d'un polynôme. (A.3.1)
3. {63} en hexadécimal

L'octet $\boldsymbol{\{b_7, b_6, b_5, b_4, b_3, b_2, b_1, b_0\}}$ peut être représenté sous la forme suivante :

$$\boldsymbol{b_7x^7 + b_6x^6 + b_5x^5 + b_4x^4 + b_3x^3 + b_2x^2 + b_1x^1 + b_0} \quad \text{(A.3.2)}$$

Toutes les opérations effectuées dans AES sont les opérations modulo-2. Ces opérations ne sont pas les mêmes opérations utilisées dans le système de numération général. Les opérations de base sur lesquelles les maths entières de l'algorithme d'AES sont basées sur addition, multiplication. Ces opérations sont expliquées dans les sous-sections suivantes :

***2-L'Addition (***$\oplus$)

Dans les additions modulo-2, deux éléments sont ajoutés en ajoutant les coefficients des puissances correspondantes dans le polynôme. L'opération d'addition ici est le XOR noté par opération de $\oplus$. Soustraction des polynômes est exactement identiques que l'addition.

Notons la propriété intéressante :

$$\boldsymbol{a \oplus a = 0} \text{ Et donc. } \boldsymbol{(a \oplus b) \oplus b = a} \quad \text{(A.3.3)}$$

L'addition de deux bytes $\boldsymbol{\{a_7, a_6, a_5, a_4, a_3, a_2, a_1, a_0\}}$ et $\boldsymbol{\{b_7, b_6, b_5, b_4, b_3, b_2, b_1, b_0\}}$ est : $\boldsymbol{\{c_7, c_6, c_5, c_4, c_3, c_2, c_1, c_0\}}$ tel que $\boldsymbol{c_i = a_i \oplus b_i}$ **.** (A.3.4)

Par exemple si on a $A = (87)_{10}$ **et** $B = (131)_{10}$ alors :

$$\boldsymbol{A + B} = (87)_{10} \oplus (131)_{10} = (212)_{10} \quad \text{et même} \quad \boldsymbol{A - B} = (212)_{10} \qquad \text{(A.3.5)}$$

Ceci peut être représenté dans différentes formes comme montré ci-dessous :

$$\mathbf{01010111} \oplus \mathbf{1000011} = \mathbf{11010100} \quad \ldots\ldots\ldots \textit{représentation binaire} \qquad \text{(A.3.6)}$$

$$\{\mathbf{57}\} \oplus \{\mathbf{83}\} = \{\boldsymbol{d4}\} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \textit{représentation} \text{ hexadécimal} \qquad \text{(A.3.7)}$$

$$\boldsymbol{x^6 + x^4 + x^2 + x^1 + 1} \oplus \boldsymbol{x^7 + x^1 + 1} = \boldsymbol{x^7 + x^6\, x^4 + x^2} \ldots. \textit{Représentation polynômial} \text{ (A.3.8)}$$

***3-Multiplication (•)***

La multiplication notée par (•) est la multiplication d'un polynôme par un polynôme irréductible du degré 8. Un polynôme est irréductible, si les seuls diviseurs du polynôme irréductible sont un et le nombre lui-même. Le polynôme irréductible désigné sous le nom d'un nombre premier dans les mathématiques. Pour l'algorithme d'AES ce polynôme irréductible est:

$$\boldsymbol{m(x) = x^8 + x^4 + x^3 + x^1 + 1} \qquad \text{(A.3.9)}$$

L'équation du polynôme irréductible peut être également s'écrite en tant que $\{\mathbf{1_00011011}\}_{\mathbf{2}}$ ou $\{\mathbf{01_1b}\}_{\boldsymbol{h}}$ format binaire et hexadécimal respectivement.

Par exemple : $\{\mathbf{72}\} \bullet \{\mathbf{18}\} = \{\boldsymbol{dc}\}$

$$(\boldsymbol{x^6 + x^5 + x^4 + x})(\boldsymbol{x^4 + x^3}) = \boldsymbol{x^{10} + x^9 + x^8 + x^5 + x^9 + x^8 + x^7 + x^4}$$

$$= \boldsymbol{x^{10} + x^7 + x^5 + x^4}$$

$$\boldsymbol{x^{10} + x^7 + x^5 + x^4\ modulo\ (x^8 + x^4 + x^3 + x + 1) = x^7 + x^6 + x^4 + x^3 + x^2}$$

La multiplication modulaire au-dessus du polynôme m(x) assure que le produit résultant de toute la multiplication a le degré plus moins de huit et peut être représentée par un octet (byte) .

À la différence de l'addition la multiplication ne peut pas être comprise par une opération simple.

# Annexe B

## Exemple de génération (expansion) des clés et de chiffrement [6]

### B.1. Génération (expansion) des clés

Cet annexe illustre la méthode d'extention (génération) des clés de chiffrement de longeur 128-bits, la méthode d'expansion est illustré dans le tableau suivant, les valeurs sont représentées en format héxadecimal.

Clé de chiffrement initiale :

KEY = 0x_2b7e1516_28aed2a6_abf71588_09cf4f3c

Avec **Nk** = 4 :

W0 =2b7e1516, W1 =28aed2a6, W2 =abf71588, W3 =09cf4f3c

| i (dec) | temp | After RotWord() | After SubWord() | Rcon[i/Nk] | After XOR with Rcon | w[i-Nk] | w[i]= temp XOR w[i-Nk] |
|---|---|---|---|---|---|---|---|
| 4 | 09cf4f3c | cf4f3c09 | 8a84eb01 | 01000000 | 8b84eb01 | 2b7e1516 | a0fafe17 |
| 5 | a0fafe17 | | | | | 28aed2a6 | 88542cb1 |
| 6 | 88542cb1 | | | | | abf71588 | 23a33939 |
| 7 | 23a33939 | | | | | 09cf4f3c | 2a6c7605 |
| 8 | 2a6c7605 | 6c76052a | 50386be5 | 02000000 | 52386be5 | a0fafe17 | f2c295f2 |
| 9 | f2c295f2 | | | | | 88542cb1 | 7a96b943 |
| 10 | 7a96b943 | | | | | 23a33939 | 5935807a |
| 11 | 5935807a | | | | | 2a6c7605 | 7359f67f |
| 12 | 7359f67f | 59f67f73 | cb42d28f | 04000000 | cf42d28f | f2c295f2 | 3d80477d |
| 13 | 3d80477d | | | | | 7a96b943 | 4716fe3e |
| 14 | 4716fe3e | | | | | 5935807a | 1e237e44 |
| 15 | 1e237e44 | | | | | 7359f67f | 6d7a883b |
| 16 | 6d7a883b | 7a883b6d | dac4e23c | 08000000 | d2c4e23c | 3d80477d | ef44a541 |
| 17 | ef44a541 | | | | | 4716fe3e | a8525b7f |
| 18 | a8525b7f | | | | | 1e237e44 | b671253b |
| 19 | b671253b | | | | | 6d7a883b | db0bad00 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 20 | db0bad00 | 0bad00db | 2b9563b9 | 10000000 | 3b9563b9 | ef44a541 | d4d1c6f8 |
| 21 | d4d1c6f8 | | | | | a8525b7f | 7c839d87 |
| 22 | 7c839d87 | | | | | b671253b | caf2b8bc |
| 23 | caf2b8bc | | | | | db0bad00 | 11f915bc |
| 24 | 11f915bc | f915bc11 | 99596582 | 20000000 | b9596582 | d4d1c6f8 | 6d88a37a |
| 25 | 6d88a37a | | | | | 7c839d87 | 110b3efd |
| 26 | 110b3efd | | | | | caf2b8bc | dbf98641 |
| 27 | dbf98641 | | | | | 11f915bc | ca0093fd |
| 28 | ca0093fd | 0093fdca | 63dc5474 | 40000000 | 23dc5474 | 6d88a37a | 4e54f70e |
| 29 | 4e54f70e | | | | | 110b3efd | 5f5fc9f3 |
| 30 | 5f5fc9f3 | | | | | dbf98641 | 84a64fb2 |
| 31 | 84a64fb2 | | | | | ca0093fd | 4ea6dc4f |
| 32 | 4ea6dc4f | a6dc4f4e | 2486842f | 80000000 | a486842f | 4e54f70e | ead27321 |
| 33 | ead27321 | | | | | 5f5fc9f3 | b58dbad2 |
| 34 | b58dbad2 | | | | | 84a64fb2 | 312bf560 |
| 35 | 312bf560 | | | | | 4ea6dc4f | 7f8d292f |
| 36 | 7f8d292f | 8d292f7f | 5da515d2 | 1b000000 | 46a515d2 | ead27321 | ac7766f3 |
| 37 | ac7766f3 | | | | | b58dbad2 | 19fadc21 |
| 38 | 19fadc21 | | | | | 312bf560 | 28d12941 |
| 39 | 28d12941 | | | | | 7f8d292f | 575c006e |
| 40 | 575c006e | 5c006e57 | 4a639f5b | 36000000 | 7c639f5b | ac7766f3 | d014f9a8 |
| 41 | d014f9a8 | | | | | 19fadc21 | c9ee2589 |
| 42 | c9ee2589 | | | | | 28d12941 | e13f0cc8 |
| 43 | e13f0cc8 | | | | | 575c006e | b6630ca6 |

***Tableau B.1 :*** *Exemple de génération des clés*

### B.2. Exemple de chiffrement [6]

Le diagramme suivant illustre les étapes de chiffrement d'une de donnée de 128-bits et une clé de 128-bits, (i.e Nb=4,Nk=4).

**DATA_IN** = 0x_3243f6a8_885a308d_313198a2_e0370734

**Key** = 0x_2b7e1516_28aed2a6_abf71588_09cf4f3c

| Round Number | Start of Round | After SubBytes | After ShiftRows | After MixColumns | Round Key Value |
|---|---|---|---|---|---|
| input | 32 88 31 e0<br>43 5a 31 37<br>f6 30 98 07<br>a8 8d a2 34 | | | | ⊕ 2b 28 ab 09<br>7e ae f7 cf<br>15 d2 15 4f<br>16 a6 88 3c = |
| 1 | 19 a0 9a e9<br>3d f4 c6 f8<br>e3 e2 8d 48<br>be 2b 2a 08 | d4 e0 b8 1e<br>27 bf b4 41<br>11 98 5d 52<br>ae f1 e5 30 | d4 e0 b8 1e<br>bf b4 41 27<br>5d 52 11 98<br>30 ae f1 e5 | 04 e0 48 28<br>66 cb f8 06<br>81 19 d3 26<br>e5 9a 7a 4c | ⊕ a0 88 23 2a<br>fa 54 a3 6c<br>fe 2c 39 76<br>17 b1 39 05 = |
| 2 | a4 68 6b 02<br>9c 9f 5b 6a<br>7f 35 ea 50<br>f2 2b 43 49 | 49 45 7f 77<br>de db 39 02<br>d2 96 87 53<br>89 f1 1a 3b | 49 45 7f 77<br>db 39 02 de<br>87 53 d2 96<br>3b 89 f1 1a | 58 1b db 1b<br>4d 4b e7 6b<br>ca 5a ca b0<br>f1 ac a8 e5 | ⊕ f2 7a 59 73<br>c2 96 35 59<br>95 b9 80 f6<br>f2 43 7a 7f = |
| 3 | aa 61 82 68<br>8f dd d2 32<br>5f e3 4a 46<br>03 ef d2 9a | ac ef 13 45<br>73 c1 b5 23<br>cf 11 d6 5a<br>7b df b5 b8 | ac ef 13 45<br>c1 b5 23 73<br>d6 5a cf 11<br>b8 7b df b5 | 75 20 53 bb<br>ec 0b c0 25<br>09 63 cf d0<br>93 33 7c dc | ⊕ 3d 47 1e 6d<br>80 16 23 7a<br>47 fe 7e 88<br>7d 3e 44 3b = |
| 4 | 48 67 4d d6<br>6c 1d e3 5f<br>4e 9d b1 58<br>ee 0d 38 e7 | 52 85 e3 f6<br>50 a4 11 cf<br>2f 5e c8 6a<br>28 d7 07 94 | 52 85 e3 f6<br>a4 11 cf 50<br>c8 6a 2f 5e<br>94 28 d7 07 | 0f 60 6f 5e<br>d6 31 c0 b3<br>da 38 10 13<br>a9 bf 6b 01 | ⊕ ef a8 b6 db<br>44 52 71 0b<br>a5 5b 25 ad<br>41 7f 3b 00 = |
| 5 | e0 c8 d9 85<br>92 63 b1 b8<br>7f 63 35 be<br>e8 c0 50 01 | e1 e8 35 97<br>4f fb c8 6c<br>d2 fb 96 ae<br>9b ba 53 7c | e1 e8 35 97<br>fb c8 6c 4f<br>96 ae d2 fb<br>7c 9b ba 53 | 25 bd b6 4c<br>d1 11 3a 4c<br>a9 d1 33 c0<br>ad 68 8e b0 | ⊕ d4 7c ca 11<br>d1 83 f2 f9<br>c6 9d b8 15<br>f8 87 bc bc = |

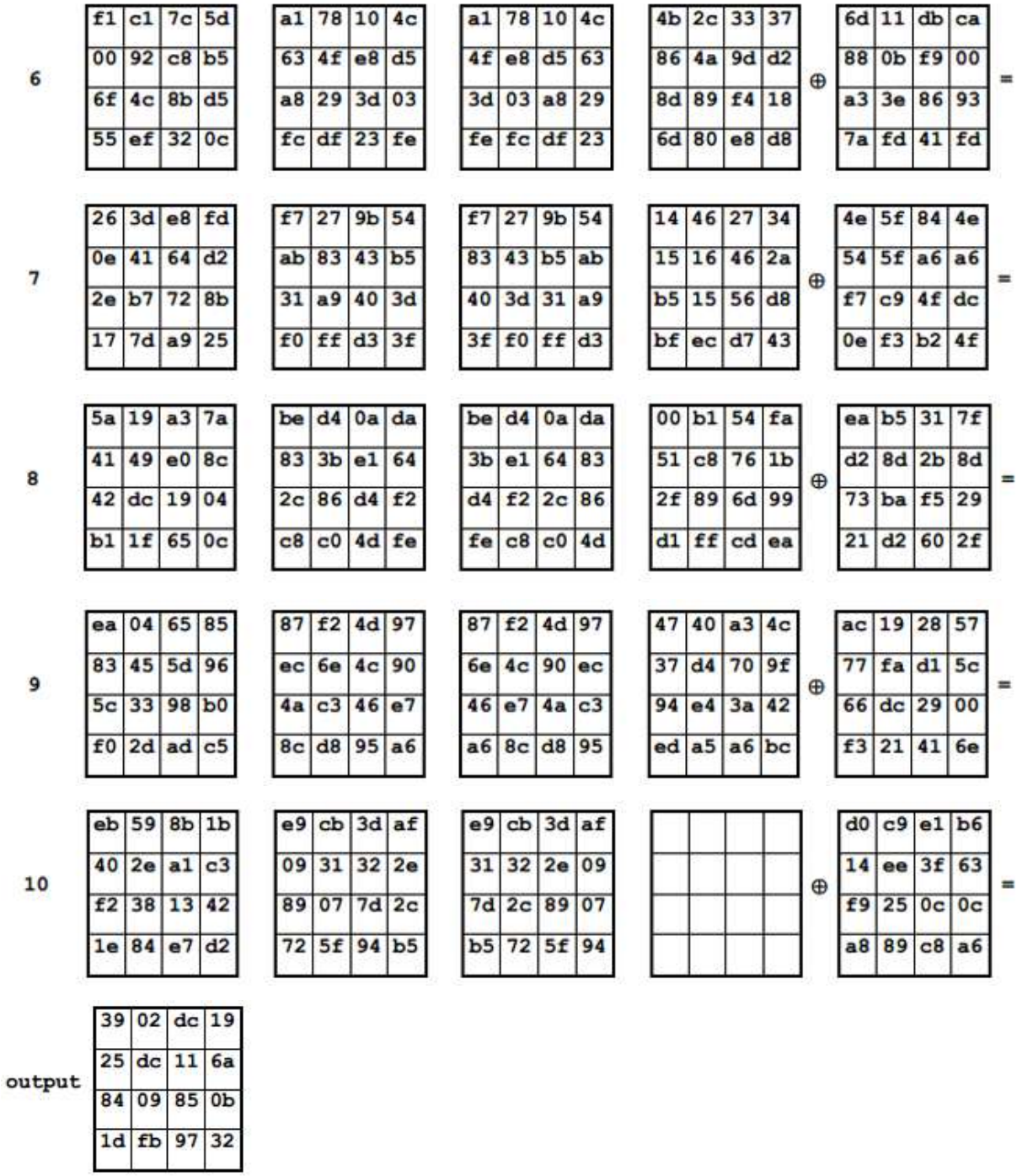



***Tableau B.2 :*** *Exemple de chiffrement*