الجمهورية الجزائرية الديمقراطية الشعبية

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR
ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE DR. YAHIA FARES DE MEDEA

FACULTE DES SCIENCES ET DE LA TECHNOLOGIE

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



# Mémoire de Fin d'Études

**Présenté pour l'obtention du Diplôme**
**de Master en Génie Électrique**

*Option* : Systèmes Intelligents et Communicants

## THEME

# Contribution à la sécurité des communications d'un réseau WSN appliqué dans un système de contrôle du trafic routier

**Encadré par :**

Dr. OULD ZMIRLI Mohamed

**Préparé par :**

Hamzi Imane

# *Remerciement*

*Je tiens à remercier d'abord mon encadreur Dr Ould Zmirli Qui m'a aidé tout au long de ce projet, Merci pour votre disponibilité et votre gentillesse.*

*Je remercie aussi mon amie « Khalladi Amira » qui m'a aidé énormément.*

*Je remercie également mes parents pour leur support et leur croyance en moi et qui m'ont poussé à dépasser tous les problèmes et aller jusqu'au bout, mes grands-mères et mes grands-pères pour leur soutien indéfectible durant mes années d'études Merci*

*à mes sœurs, mon petit frère, mes amies «Samia Guettar» «Bouchou Djamila », «Samia Bachouche », « Duifi Nadia » et «Fayza Guettar » .*

*Enfin, je remercie toute personne ayant contribuée de prés ou de loin à la progression de ce projet.*

# Dédicace

Je tiens à dédier ce mémoire :

à mes chers .....

« Mes Parents », pour Leur soutien et leur amour ;

À mes sœurs « Mariem et Basma » et mon petit frère

« Ahmed elAmine » ;

À mes grands pères « Houcine et Abdelkader » ;

À mes grands-mères « khadidja et kheira » ;

À mes oncles et mes tantes ;

À mes cousins et mes cousines «Djamila»«Ahlem»«Mariem» et

« Zineb »;

À toutes mes amies : Nadia, les 2 Samia, Djamila, Hassiba, Amina,

Fayza, Sara, Soumia, Radia, Fatma, Fatima, Fadila,

Imane, Et Amel, Salima,

Houria.

**الملخص:**

هذا المشروع يشمل تصميم و تنفيذ خوارزمية SEA 128 bits الخاصة بعمليات التشفير وفك التشفير للبيانات على الرقاقة الإلكترونية AGPF باستعمال النواة PSARTAN 3E لضمان امن الاتصالات في شبكات المجسات اللاسلكية.

**الكلمات المفتاحية : أمن** ، نظام التشفير AES ، التشفير، فك التشفير،FPGA ، RcSF

**Résumé**

Ce projet consiste à la conception et l'implémentation software de l'algorithme de chiffrement et de déchiffrement AES 128 bits sur un circuit FPGA de type Spartan 3E. Le but du cryptage est d'assurer la sécurité des communications dans les réseaux de capteurs sans fil.

**Mot clé :** Sécurité, AES, Chiffrement, Déchiffrement, Cryptage, Décryptage, FPGA, RcSF

**Abstract**

This project involves the design and software implementation of the AES 128 bits encryption and decryption algorithm on a Spartan 3E FPGA circuit. The purpose of encryption is to ensure the security of communications in wireless sensor networks.

**Key Words:** Security, AES, cipher, Inverse Cipher, Encryption, Decryption, FPGA, WSN

# Sommaire

Résumé……i

Sommaire……ii

Listes des Figures……...vi

Listes des Tables……..viii

Listes des Acronymes……ix

Introduction générale……..1

## Chapitre I : Sécurité des réseaux de capteurs sans fil.

I.1. Introduction ..... 3

I.2. Sécurité dans les RCSF ..... 3

I.3. Politiques de sécurité dans les RcSF ..... 5

I.4. Mécanismes de sécurité ..... 5

I.4.1. Partitionnement de données ..... 6

I.4.2. La cryptographie ..... 7

I.5. Le chiffrement ..... 8

I.5.1. Le chiffrement symétrique ..... 8

I.5.2. Le chiffrement asymétrique ..... 9

I.5.3. Le chiffrement mixte ..... 10

I.6. Cryptographie symétrique Vs Cryptographie asymétrique ? ..... 10

I.7. L'AES (Advanced Encryption Standard) ..... 11

I.7.1. Les résultats d'un concours ..... 11

I.7.2. Le choix : Rijndael ..... 12

I.7.2.1. Chiffrement et Déchiffrement ..... 14

I.7.2.1.1. Chiffrement……..…….16

I.7.2.1.2. Déchiffrement……..……21

I.7.2.2. Calcul de la clé ..... 24
I.7.3. Avantages et limites ..... 26
I.7.4. Attaques ..... 27
I.8. Conclusion ..... 27

## Chapitre II : Conception des systèmes numériques embarqués

II.1. Introduction ..... 28
II.2. Conception des systèmes numériques ..... 28
II.3. Architectures à base d'un processeur ..... 28
II.3.1. Les processeurs d'usage général (GPP) ..... 28
II.3.2. Les processeurs de traitement du signal ..... 29
II.3.3. Les processeurs reconfigurables ..... 29
II.4. Les circuits logiques programmables ..... 30
II.4.1. Les PLDs et les CPLDs ..... 30
II.4.2. Les FPGAs ..... 31
II.4.3. Les ASICs ..... 32
II.5. Les systèmes numériques embarqués ..... 32
II.5.1. Codesign : Conception conjointe logicielle/matérielle ..... 33
II.5.2. Flot de la conception conjointe logicielle/matérielle ..... 34
II.5.3. Les composants virtuels (IPs) ..... 35
II.5.4. Les accélérateurs matériels ..... 36
II.6. Les systèmes sur puce (SoC) ..... 37
II.6.1. Définitions d'un SoC ..... 37
II.6.2. Les contraintes liées aux systèmes embarqués ..... 38
II.6.3. Les SoC et les SoPC ..... 38
II.7. Communications sur puce ..... 39
II.7.1. Exemple : Le Bus CoreConnect ..... 40

II.8. Plateformes multi-composants ..... 41
II.8.1 Les deux grandes familles des FPGA ..... 42
II.8.1.1 La famille Xilinx ..... 42
II.8.1.2 La famille Altéra ..... 43
II.8.2 Les processeurs embarqués ..... 43
II.8.2.1 Les processeurs hardcore ..... 44
II.8.2.2 Les processeurs softcore ..... 45
II.8.3. Les processeurs softcore Open Source ..... 47
II.8.4 Les Outils de développement de Xilinx ..... 48
II.8.4.1 Xilinx ISE ..... 48
II.8.4.2 Xilinx EDK ..... 48
II.9 Conclusion ..... 49

## Chapitre III : conception et Implémentation de l'algorithme AES sur le circuit FPGA

III.1. Introduction ..... 49
III.2. Carte FPGA utilisé : ..... 49
III.3. Outil Software utilisé : ..... 52
III.4. Réalisation du projet sous EDK ..... 52
III.4.1. Utilisation du Base System Builder de XPS ..... 52
III.4.2. Implémentation du matériel ..... 58
III.4.3. Utilisation de l'outil SDK ..... 60
III.5. Organigrammes des fonctions de test de cryptage/décryptage implémentée ..... 60
III.6. Résultats du test pratique sur la carte FPGA : ..... 63
III.7. Conclusion : ..... 64
Conclusion générale……..……65

Bibliographie…….66

# Listes des Figures

Figure I-1 : Technique de partitionnement de données……..6

Figure I-2 : Principe de la cryptographie……7

Figure I-3 : Chiffrement symétrique……...9

Figure I-4 : Chiffrement asymétrique……...10

Figure I-5 : Différents participants au concours AES……...11

Figure I-6 : Schéma générale de Réjindel……13

Figure I-7 : Schéma de différentes étapes……14

Figure I-8 : Table d'état de texte…….....15

Figure I-9 : Le chiffrement…….…...16

Figure I-10 : illustrer l'effet du SubBytes () transformation sur l'état……..17

Figure I-11 : Table d'état des clés…….....17

Figure I-12 : Les S-box : la subtitution évolue pour le xy du multiplet (format hexadécimal)…….18

Figure I-13 : Fonction SubBytes applique S-box sur chaque multiplet de l'état……..18

Figure I-14 : Schéma de l'étape ShiftRow…….....19

Figure I-15 : Etape de MixColumn…….....20

Figure I-16 : AddRoundKey…….….....20

Figure 1-17 : Le déchiffrement AES……21

Figure I-18 : InvShiftRows () cycliquement changement les dernières trois ligne dans l'état……..……23

Figure I-19 : boite S inverse : valeur de substitution de l'octet (au format hexadécimal)……23

Figure I-20 : Schéma des opérations effectuées sur la clé…….....24

Figure I-21 : Expansion de la clé avec bloc "commun"…….....25

Figure I-22 : Expansion de la clé avec les blocs "multiples de Nk"…….25

Figure II-1 : Les différents circuits programmables……30

Figure II-2 : Architecture conceptuelle d'un FPGA (Architecture, Interconnections et CLB).....32

Figure II-3 : Flot de conception standard…….....35

Figure II-4 : Objectif du SoC……...37

Figure II-5 : Flot de conception Xilinx……39

Figure II-6 : Le bloc PLB dans un système à base de MicroBlaze……..41

Figure II-7 : Détails d'une plateforme multi-composants…….....41

Figure II-8 : Schéma-bloc fonctionnel du noyau MicroBlaze…….46

Figure II-9 : Les processeurs NIOS et NIOS- II de Altera……..47

Figure III-1: système de test pratique……..50

Figure III-2: plateforme Xtreme DSP Starter Kit……....51

Figure III-3 : Modules du système Microblaze utilisés……...52

Figure III-4: Démarrage application XPS……....53

Figure III-.5: création nouveau projet XPS……..53

Figure III-6 : Choix carte FPGA…….....54

Figure III-7 : Sélection solution mono processeur……...54

Figure III-8 : sélection taille mémoire et fréquence d'horloge microblaze……..55

Figure III-9 : sélection modules utiles……..55

Figure III-10 **:** Le navigateur des projets de Xilinx Platform Studio……...56

Figure III-11 : Capture d'écran des ports utilisés…….56

Figure III-12 : Capture d'écran des adresses réservées aux différents modules……...57

Figure III-13 **:** fichier de spécification de matériel MHS…….....57

Figure III14 **:** fichier de contraintes UCF…….....58

Figure III-15 **:** Schéma RTL du système à base du MicroBlaze implémenté……..58

Figure III-16 **:** Le navigateur des projets de Software Development Kit…….....60

Figure III-17 : Organigramme de la fonction de test de l'algorithme de cryptage AES128…….....61

Figure III-18 : Organigramme de la fonction de test de l'algorithme de décryptage AES128…….....62

Figure III-19 : Résultat du test pratique sur la carte…….....63

# Listes des Tables

Table I-1: Décalage selon la taille des blocs de messages..……..19

Table I-2 : Nombres des rondes à effectuer..……....20

Table I-3 : Table de correspondance des Rcon []…….……25

Table III-1: Rapport des résultats concernant l'occupation de surface ……...……59

# Listes des Acronymes

AES: Advanced Encryption Standard.

ASIC: Application Specific Integrated Circuit.

CBC: Cipher Block Chaining.

CLB: Configurable Logic Block.

CPLDs : Complex Logic Programmable Device.

DES: Data Encryption Standard.

DCM: Digital Clock Manager.

DCR: Device Control Register.

DSP: Digital Signal Processors.

ECB: Electronic Code Book.

ECC: Elliptic Curve Cryptography.

ECS: Engineering Capture System.

EDK: Embedded Development Kit.

ESA: Agence Spatiale Européenne.

FPGA: Field Programmable Gâte Array.

FPU: Floating Point Unit.

GAL: Généric Array Logic.

GPP: General Purpose Processor.

HDL: Hardware Design Language.

HW/SW: Hardware/Software.

IP: Internet Protocol.

ISE: Integrated Software Environment.

LUT: Look Up Table.

NIST: National Institute of Standards and Technology.

OFB: Output Feed Back.

OPB: On-chip Peripheral Bus.

PAL: Programmable Array Logic.

PDA: Personal Digital Assistants.

PLB: Processor Local Bus.

PLD : Programmable Logic Device.

Rcon : Règle de construction.

RcSF : Réseaux de capteurs sans fil.

RSA: Rivest Shamir Adleman.

RC4: Rivest Cipher 4.

RC6: Rivest Cipher 6.

SoC: System on Chip.

S-Box: Subtitution Box.

SoPC: System on Programmable Chip.

SPARC-V8: Scalable Processor ARChitecture.

SPLD: Simple Programmable Logic Devices.

VHDL: Very High Speed Integrated Circuit Hardware Description Language.

XPS: Xilinx Platform Studio.

XST: Xilinx Synthesis Technology.

# Introduction générale

# 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.

Dans un contexte où les échanges d'information dématérialisées se développent, il est indispensable de pouvoir bénéficier de systèmes sécurisés pour protéger les données a caractère personnel ou confidentiel. Il est donc, nécessaire d'avoir accès à des outils permettant une meilleure protection contre les intrusions arbitraires de la confidentialité des données. Le chiffrement est souvent le seul moyen efficace pour répondre à ces Exigences.

La cryptographie est l'ensemble des processus de verrouillage visant à protéger l'accès à certaines données afin de les rendre incompréhensible aux personnes non autorisées. Les technologies cryptographiques sont ainsi reconnues comme étant des outils essentiels de sécurité des données et de la confiance dans les communications et le commerce électroniques**.** 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.

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 C standard de

Xilinx. Et finalement, le code 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 chapitres a été appliquée:

Le premier chapitre fournit un aperçu sur la sécurité dans les réseaux de capteurs sans fil et 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.

Le deuxième chapitre présente la conception des systèmes numériques embarqués et l'environnement de la description de l'architecture matérielle de notre propre conception du circuit développé sur un circuit FPGA.

Dans le dernier chapitre, nous utiliserons l'environnement de développement EDK de Xilinx englobant les deux outils : le XPS pour créer l'architecture matérielle à base du Microblaze, et le SDK pour concevoir le programme de l'algorithme de l'AES qui sera exécuté sur le système embarqué à base du Microblaze. Et enfin, nous présentons les résultats des tests pratiques du circuit développé sur la plate-forme de type SPARTAN 3A DSP1800A starter kit.

# Chapitre I :

## Sécurité des réseaux de capteurs sans fil

## I.1. Introduction

Ce chapitre propose un état de l'art sur la sécurité des réseaux de capteurs sans fil. D'autant que la confidentialité des informations échangées sur un RcSF est primordiale, la sécurité est donc une dimension importante pour ces réseaux. Les solutions proposées pour les réseaux ad hoc classiques ne peuvent pas s'appliquer aux RcSF car les capteurs sont limités par leur puissance d'énergie et de calcul ce qui rend la conception d'une politique de confiance un défi de taille. Nous allons présenter quelques défis, politiques et solutions de sécurité pour ce type de réseaux.

## I.2. Sécurité dans les RCSF [1]

Le succès des RcSF est lié, en grande partie, à leur efficacité dans le monitoring de leur environnement et la capacité de ses nœuds à intervenir, d'une façon autonome, dans des fonctions sensibles. Cette intervention consiste à protéger les transmissions des données captées, à moindre cout, dans leur intégralité sans aucun accès ou modification par une autre entité étrangère. L'environnement de déploiement des RcSF regroupe des domaines d'application très critiques où la sécurité du système représente un enjeu majeur et l'existence de failles est intolérable. En effet, les RcSF connaissent actuellement une grande extension et une large utilisation dans différents types d'applications, souvent sensibles, exigeant une sécurité renforcée. Toutefois, les nœuds capteurs disposent de capacités mémoires, énergétiques et de traitement très limitées. Ces applications font que l'introduction des mesures de sécurité classiques est restreinte. En d'autres termes, la protection des communications dans les réseaux de capteurs sans fil nécessite un niveau de sécurité élevé entre les différents liens de communication. Cependant, la sécurisation des liens n'est pas suffisante pour garantir la fiabilité du réseau. En effet, la sécurisation des entités du réseau permet d'empêcher toute attaque visant à i) compromettre les nœuds, ii) récupérer leur identité, iii) récupérer les informations captées et échangées avec les autres nœuds capteurs, iv) altérer le bon fonctionnement du réseau par l'introduction de fausses informations par le nœud compromis…etc. La sécurisation des RcSF est la source de plusieurs travaux de recherche proposant des solutions économes en énergie et qui respectent d'autres contraintes propres aux capteurs telles que la puissance de calcul et les capacités de stockage. Par ailleurs, l'absence d'infrastructure fixe, le déploiement dans des endroits ouverts et hostiles proches des territoires ennemis rendent les réseaux de capteurs sujet à différents types d'attaques qui visent à exploiter leurs vulnérabilités et se permettre des privilèges d'accès pour modifier,

d'insérer ou de supprimer une partie ou la totalité des informations qui circulent sur le réseau. Deux types de vulnérabilités sont à distinguer pour les RcSF : logiques et physiques. Les vulnérabilités physiques se résument essentiellement aux ressources limitées des nœuds capteurs notamment en termes d'énergie, de stockage et de calcul. Les vulnérabilités logiques dérivent des vulnérabilités physiques, principal handicap au développement de mécanismes complexes, complets et fiables. Par conséquent, les contraintes matérielles et énergétiques s'imposent dans le modèle de conception d'un mécanisme de sécurité fiable.

Les vulnérabilités liées à la sécurité du routage dans les réseaux de capteurs sans fil se résument en :

- ✓ La technologie sans fil sous jacente. Quiconque possédant le récepteur adéquat peut potentiellement écouter ou perturber les messages échangés ;
- ✓ Les nœuds eux-mêmes sont des points de vulnérabilité du réseau car une attaque peut compromettre un composant laissé sans surveillance ;
- ✓ L'absence d'infrastructure fixe pénalise l'ensemble du réseau dans la mesure où il faut faire abstraction de toute entité centrale de gestion pour l'accès aux ressources ;
- ✓ Les mécanismes de routage sont d'autant plus critiques dans les RcSF que chaque nœud participe à l'acheminement des paquets à travers le réseau. De plus, les messages de routage transitent sur les ondes radio.

Plusieurs solutions, basées sur des approches algorithmiques et sur des approches cryptographiques, sont proposées pour les RcSF. Nous nous intéresserons dans la suite de notre travail uniquement aux solutions cryptographiques.

Afin de mieux comprendre l'aspect sécurité dans les RcSF (contexte, difficultés, objectifs et solutions), nous allons, tout d'abord, répondre aux deux questions posées ci-dessous.

1) Pourquoi les RcSF sont vulnérables ?

- Des vulnérabilités liées aux réseaux ad hoc ;
- Contraintes matérielles et énergétiques ;
- Débit faible et une portée radio limitée ;
- Déploiement sur des zones à risque proches ou au cœur des zones ennemies ;
- Absence d'infrastructures fixes ;
- Absence de sécurité physique ;

- De nouvelles technologies émergentes ;
- Les attaquants sont souvent mieux équipés ;
- La sécurité ne peut pas être sure à 100 %.

2) Pourquoi les solutions de sécurité ne peuvent jamais être sures à 100 % ?

- Difficulté de remplacer la politique de sécurité en cas d'attaques massives sur le système ;
- Différents types d'attaques simultanées sur un même réseau ;
- Plus les mécanismes de sécurité sont confrontés à des contraintes, moins ils sont efficaces.

### I.3. Politiques de sécurité dans les RcSF

Une politique de sécurité dans les réseaux de capteurs sans fil est un ensemble de mécanismes, d'algorithmes, de procédures et de schémas cohérents conçus pour maintenir un certain niveau de sécurité. Une politique de sécurité doit assurer au minimum : i) un contrôle d'accès : détecter et interdire l'ajout de nouveaux éléments corrompus au réseau ; ii) l'intégrité et iii) la confidentialité des données transmises. Les protocoles de sécurité reposent sur les politiques suivantes :

- Sécurité des communications : choisir des outils adéquats (comme la cryptographie symétrique) pour l'établissement de liens sécurisés entre les nœuds communicants. Ces outils doivent respecter les limites des nœuds capteurs.
- Sécurité du routage et des données agrégées : l'accès aux données pour les agréger, les filtrer, les compresser ou décompresser doit se faire uniquement par les nœuds autorisés du réseau.
- Sécurité physique des nœuds : protéger les nœuds contre toute tentative de compromission, détecter et mettre en quarantaine les nœuds compromis.

### I.4. Mécanismes de sécurité

Plusieurs solutions ont été développées pour contrer les attaques sur les RcSF. On distingue des solutions spéciales au routage où la fonction de sécurité concerne la protection de la donnée transmise, d'autres solutions sont conçues pour protéger les communications entre les nœuds capteurs en assurant la fiabilité des liaisons par des fonctions

cryptographiques, et enfin des solutions qui se chargent de protéger les nœuds du réseau, de détecter et d'exclure les nœuds malicieux par des mécanismes de détection d'intrusions.

### I.4.1. Partitionnement de données

C'est une solution proposée dans le but d'empêcher les attaquants de récupérer l'intégralité des informations qui circulent sur le réseau. Le principe est de découper, au niveau du nœud source, l'information à transmettre en plusieurs paquets de tailles fixes et chaque paquet de données sera envoyé sur un chemin différent. L'entité de destination finale, et après la réception de tous les paquets de données, procède à la reconstitution du message initial émis par le nœud source. Afin de saisir intégralement l'information échangée entre la source et sa destination, un attaquant doit scruter tous les chemins utilisés dans la transmission du message, ce qui est difficile voire impossible vu la taille des réseaux et le nombre important de chemins possibles entre chaque paire de nœuds. Cependant, cette solution est gourmande en énergie car elle implique un grand nombre de nœuds pour acheminer chaque paquet vers l'entité destinatrice. Un exemple d'une telle solution est donné par la figure I-1.

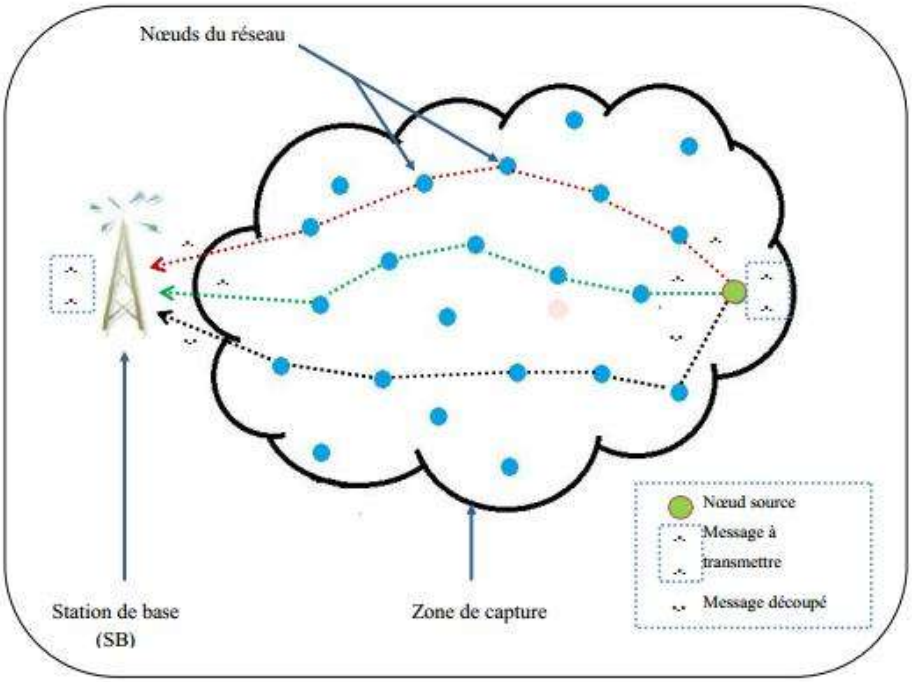



Figure I-1 : Technique de partitionnement de données. **[1]**

### I.4.2. La cryptographie

Le mot cryptographie est composé de deux mots grecs : « crypto » qui signifie *caché* et « graphie » qui signifie *écrire*, d'où la signification complète de la cryptographie est « *l'écriture secrète* ». La cryptographie est définie comme la science qui convertit les informations *en clair* en informations *cryptées* c'est-à-dire codées. Le principe est donné par la Figure I-1. La plupart des mécanismes de sécurité des communications sont basés sur des outils cryptographiques utilisant des informations secrètes représentées par des nombres premiers, dites clés, combinées, en entrée d'une opération cryptographique, au message à coder pour produire le message crypté. On distingue quatre types de clés utilisées dans les opérations cryptographiques : clé individuelle, clé par paire, clé de groupe et clé globale. Plusieurs outils cryptographiques sont utilisés pour assurer la sécurité du routage, des entités du réseau et des liens. Pour assurer un système cryptographique fiable, deux communicants doivent choisir soigneusement leur clé de chiffrement/déchiffrement et doivent appliquer les principes suivants, car un attaquant peut essayer un certain nombre de possibilités et par chance tomber rapidement sur la solution :

- La sécurité doit se reposer sur le secret de la clé et non pas sur la sécurité de l'algorithme.
- Le déchiffrement sans connaissance préalable de la clé doit être impossible en un temps raisonnable.
- Calculer la clé à partir du texte en clair et du texte chiffré doit être impossible en un temps raisonnable.

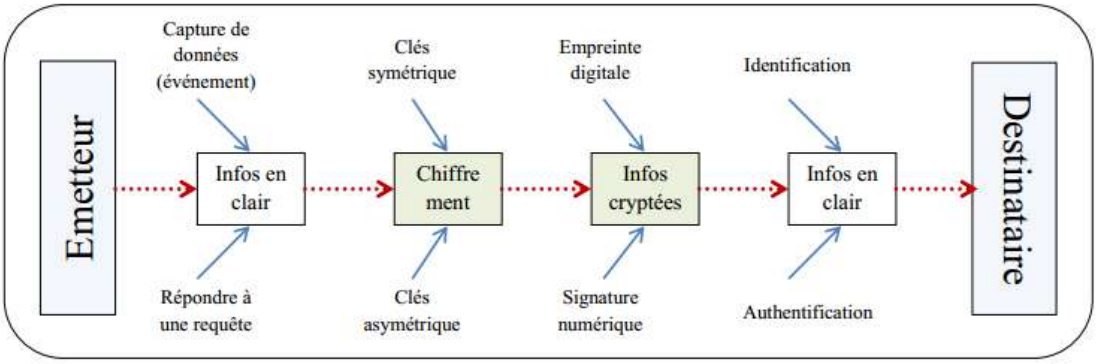



Figure I-2 : Principe de la cryptographie. **[1]**

Selon les applications et le niveau de sécurité désiré, la clé est peut être un nombre premier qui varie de 32 (25) bits à plusieurs milliers (220) ou plus. On utilise les nombres premiers car il n'existe, du moins pour le moment, aucune méthode connue pour les retrouver rapidement. Egalement, la sécurité ne se repose plus uniquement sur le secret partagé mais sur des problèmes, notamment mathématiques, connus de tous (exemple : la factorisation) et d'une information connue de tous (la clé publique).

## I.5. Le chiffrement

Le chiffrement est un système basé sur des clés cryptographiques pour assurer la confidentialité des données. Les clés peuvent être privées (symétriques) ou publiques (asymétriques).

### I.5.1. Le chiffrement symétrique

Une même clé est utilisée par une paire de nœuds pour chiffrer et déchiffrer les messages échangés entre eux en utilisant un algorithme de chiffrement symétrique. Le chiffrement symétrique se fait soit en fractionnant les données en bit à bit (chiffrement par flots) ou bien en blocs de taille fixe (chiffrement par blocs).

Les algorithmes de chiffrement par blocs : Ce sont les algorithmes les plus utilisés dans les systèmes cryptographiques. Les algorithmes de chiffrement par blocs (Block Cipher en anglais) décomposent les données à transmettre en blocs de tailles généralement égales (comprises entre 64 et 512 bits) et sont ensuite chiffrés les uns après les autres selon différents modes (ECB, CBC, OFB, CTR…etc.). Outre la confidentialité, les algorithmes de chiffrement par blocs peuvent être utilisés dans d'autres outils de cryptographie comme les fonctions de hachage et les codes MAC (Message Authentication Code).

Les algorithmes de chiffrement par flots : ils convertissent les données en clair en texte chiffré bit par bit en combinant les flux de bits en clair avec un flux de bits aléatoires. L'opération de combinaison est souvent bijective (un ou exclusif (XOR)). Le déchiffrement est réalisé par l'inversement de l'opération bijective en combinant les données chiffrées avec le même flux aléatoire de bits pour retrouver les données en clair. Le flux de bits aléatoire est généré avec un générateur pseudo-aléatoire initialisé avec la clé de chiffrement et un vecteur public d'initialisation (IV) différent pour chacun des messages à chiffrer. L'inconvénient des

algorithmes de chiffrement symétrique est la difficulté de distribuer les clés car chaque nœud a besoin d'une clé partagée avec un autre nœud du réseau, et si on considère que le nombre de nœuds du réseau est *n*, le système aura à gérer *n*(n-1)/2* clés. Malgré cela, les algorithmes de chiffrement à clé privée ne requièrent aucune opération mathématique complexe et gourmande en énergie durant les phases de chiffrement et de déchiffrement. Exemples d'algorithmes symétriques : AES (Advanced Encryption Standard), DES (Data Encryption Standard), RC4 (Rivest Cipher 4).

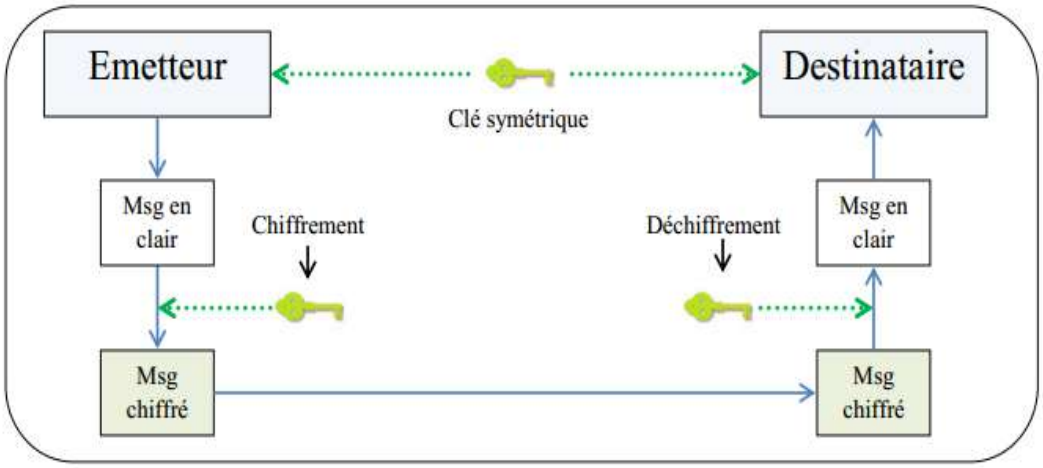



Figure I-3 : Chiffrement symétrique. **[1]**

### I.5.2. Le chiffrement asymétrique

Deux clés sont générées par le récepteur du message : une clé privée maintenue chez le récepteur et une clé publique diffusée à tous les nœuds émetteurs. La clé publique sert au chiffrement d'un message et la clé privée sert au déchiffrement du même message. La puissance de ce mode de chiffrement repose sur l'impossibilité de déduire la clé privée à partir de la clé publique. La fonction de distribution de clés est fortement simplifiée où chaque nœud a besoin uniquement d'une paire de clés, et si on considère que le nombre de nœuds du réseau est *n*, le système aura à gérer *2*n* clés. Cependant, la complexité des opérations mathématiques et le stockage de toutes les clés publiques de tous les autres nœuds exigent des capacités de traitement et de stockage importantes synonyme d'une consommation conséquente d'énergie. Exemples d'algorithmes asymétriques : RSA (Rivest Shamir Adleman), ECC (Elliptic Curve Cryptography).

### I.5.3. Le chiffrement mixte :

Bien que le chiffrement asymétrique soit pratique et le chiffrement symétrique est performant, la combinaison des deux techniques s'avère intéressante et elle a tendance à devenir un système cryptographique moderne. Pour cela, on génère et on échange avec une clé publique une clé de session symétrique et on utilise cette dernière pour le chiffrement des messages échangés.

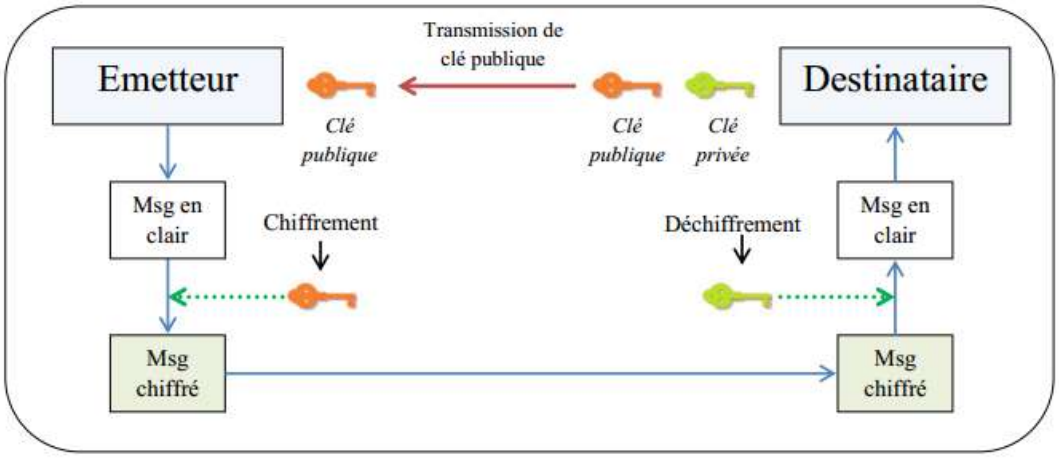



Figure I-4 : Chiffrement asymétrique. **[1]**

### I.6. Cryptographie symétrique Vs Cryptographie asymétrique ?

Dans les systèmes basés sur la cryptographie symétrique, les faiblesses des nœuds capteurs sont contournées par la réduction des opérations de calculs et de stockage de clés. Contrairement aux systèmes symétriques, la cryptographie asymétrique nécessite des temps de calcul importants et une consommation d'énergie énorme due au calcul des algorithmes à clés publiques et à la transmission des certificats numériques utilisés par le destinataire pour authentifier la clé publique reçue. Cependant, la gestion de clé dans les systèmes à clé privée (symétriques) est beaucoup plus compliqué à cause des difficultés rencontrées lors de la distribution, l'échange et la découverte des clés entre nœuds voisins. Un problème que résout le système à clés publiques par l'introduction de certificats numériques pour une seule paire de clés (privée / publique) par nœud utilisée dans le chiffrement et le déchiffrement des messages. Bien que le chiffrement à clé publique possède de gros avantages, il demeure inapproprié pour les RcSF pour sa lenteur d'exécution et son cout considérable en termes de

ressources physiques et énergétiques. Cependant et malgré que le chiffrement à clé privée possède des inconvénients liés au problème de l'établissement de clés entre les nœuds, il demeure le système le plus approprié pour les RcSF. Les caractéristiques spéciales aux réseaux de capteurs sans fil ne permettent pas d'utiliser des méthodes cryptographiques complexes. En effet, le temps de calcul et la consommation d'énergie dans les traitements doivent être raisonnables. Pour ces deux raisons, l'emploi de techniques de cryptographie simples est indispensable. Dans ce qui suit, nous allons étudier en détail la fonction de gestion de clés pour la conception des systèmes cryptographiques simples et efficaces pour les RcSF.

## I.7. L'AES (Advanced Encryption Standard) [2]

L'AES (Advanced Encryption Standard), est un algorithme de chiffrement symétrique établi par deux chercheurs belges entre 1998 et 2000 sous le nom de code Rijndael. Cet algorithme a été choisi suite à un concours international organisé par la NIST (National Institute of Standards and Technology), une agence américaine, afin de désigner le successeur de l'algorithme DES qui commençait à montrer ses limites.

### I.7.1. Les résultats d'un concours :

La progression de la puissance des ordinateurs a causé la mort du DES. Ce dernier n'est plus jamais utilisé lorsque la sécurité demandée est forte (utilisation militaire, documents "secrets", etc.). Pour cette tâche, on préfère utiliser l'algorithme connu sous le nom générique d'AES (Advanced Encryption Standard), issu d'un concours créé en raison des faiblesses avérées du DES. Le véritable nom de l'AES est le rijndael, nom résultant de la contraction des noms de ses inventeurs : Rijmen et Deamen.

Le Triple DES demeure toutefois 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 rechiffrer les documents existants.

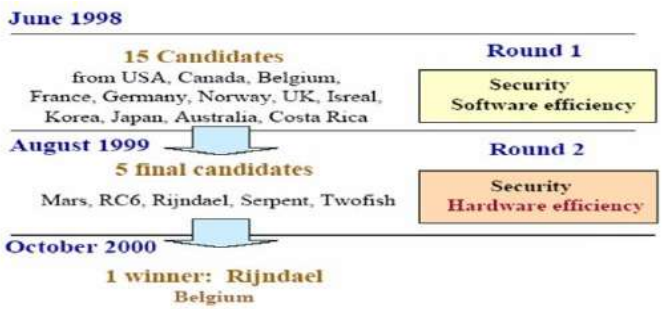



Figure I-5: Différents participants au concours AES. **[2]**

**Cahier des charges :**

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

- ✓ La sécurité générale,
- ✓ Le coût en terme 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.

Au niveau du chiffrement/déchiffrement, les résultats varient assez fortement. Cependant, Serpent reste le moins bon pour la majorité des plates-formes, Rijndael et RC6 étant les meilleurs.

### I.7.2. 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.

Il possède les propriétés suivantes :

– Plusieurs longueurs de clef et de bloc sont possibles : 128, 192, ou 256 bits;

– Le nombre de cycles ("rondes") varie en fonction de la longueur des blocs et des clés (de 10 à 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

À chaque ronde, quatre transformations sont appliquées :

1. substitution d'octets dans le tableau d'état.
2. décalage de rangées dans le tableau d'état
3. déplacement de colonnes dans le tableau d'état (sauf à la dernière ronde).
4. addition d'une "clef de ronde" qui varie à chaque ronde.

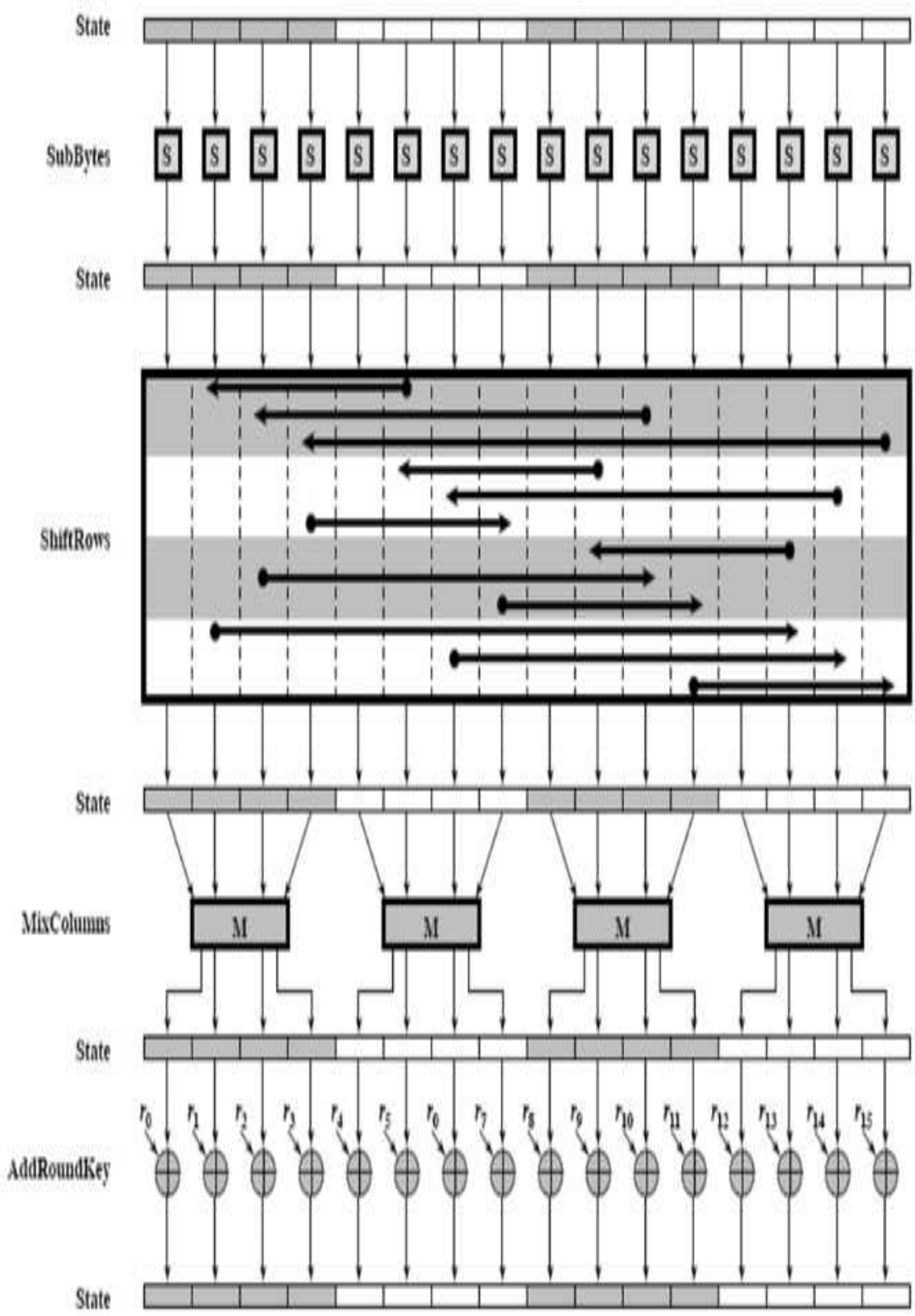



Figure I-6: Schéma générale de Réjindel. **[2]**

### I.7.2.1. Chiffrement et Déchiffrement :

L'ordonnancement des étapes est illustré à la figure I-7 :

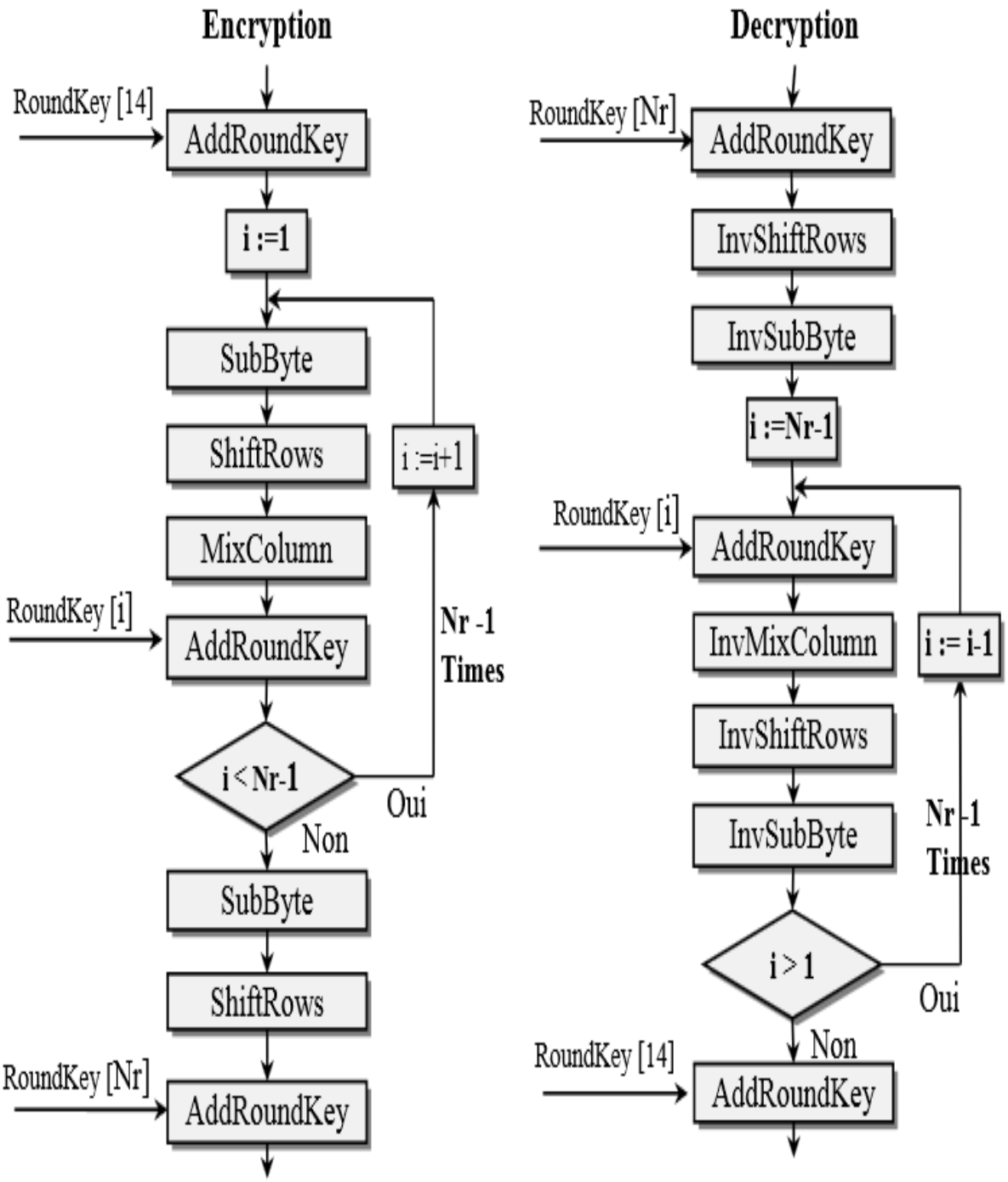



Figure I-7: Schéma de différentes étapes. **[2]**

### Table d'état du texte et des clés :

Le message et la clé sont conservés sous forme de tables représentées respectivement aux 2 figures suivantes. Le nombre de colonnes dépend des tailles des textes et clés.

Nb=Lbloc/32

Nk=Lclef/32

Une colonne du tableau correspond à un mot de 32 bits. Ainsi, chaque petit bloc représente 8 bits, donc 1 octet. L'input et l'output sont donc gérés comme des séquences linéaires d'octets.

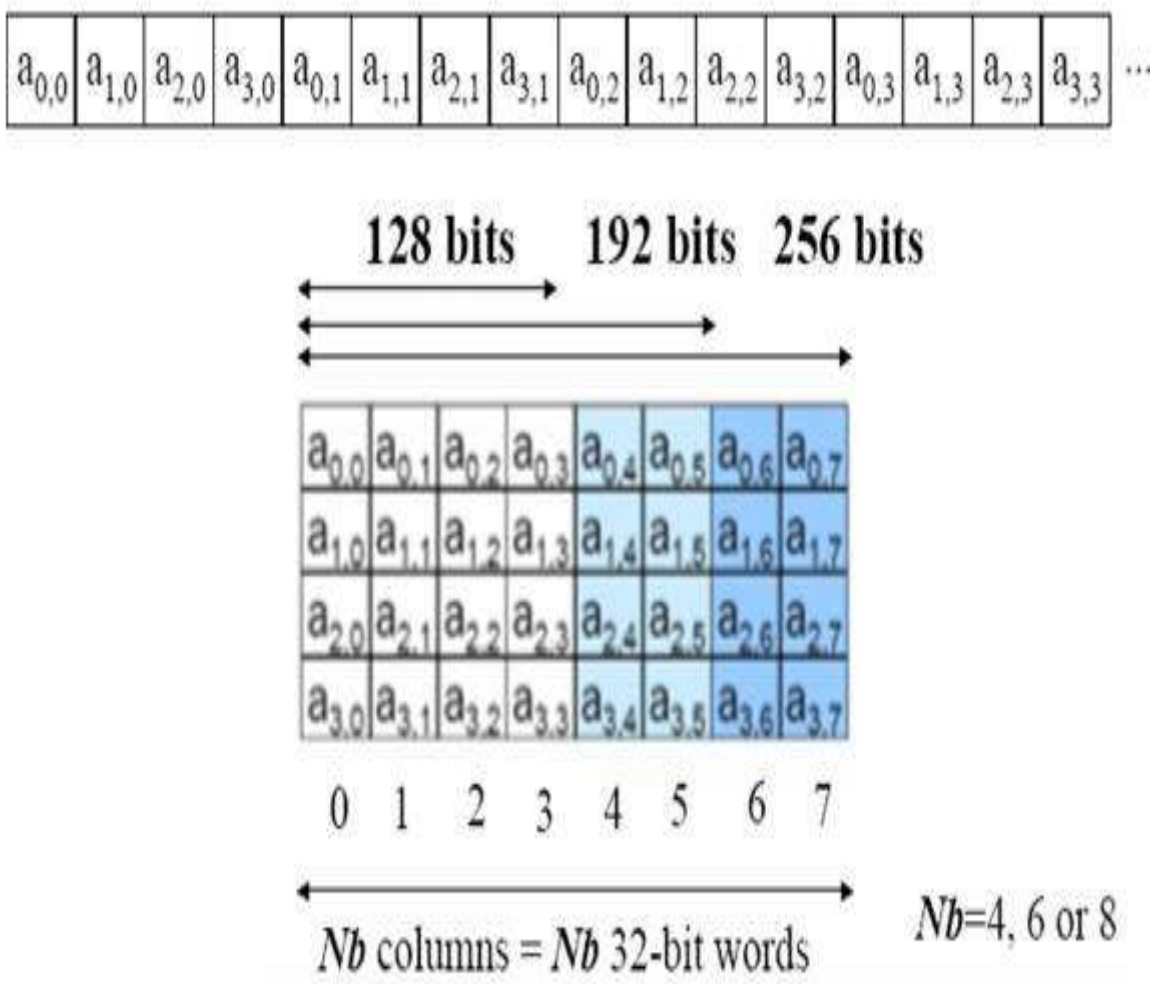



Figure I-8: Table d'état du texte. **[2]**

### I.7.2.1.1. Chiffrement :

**Cipher [3]**

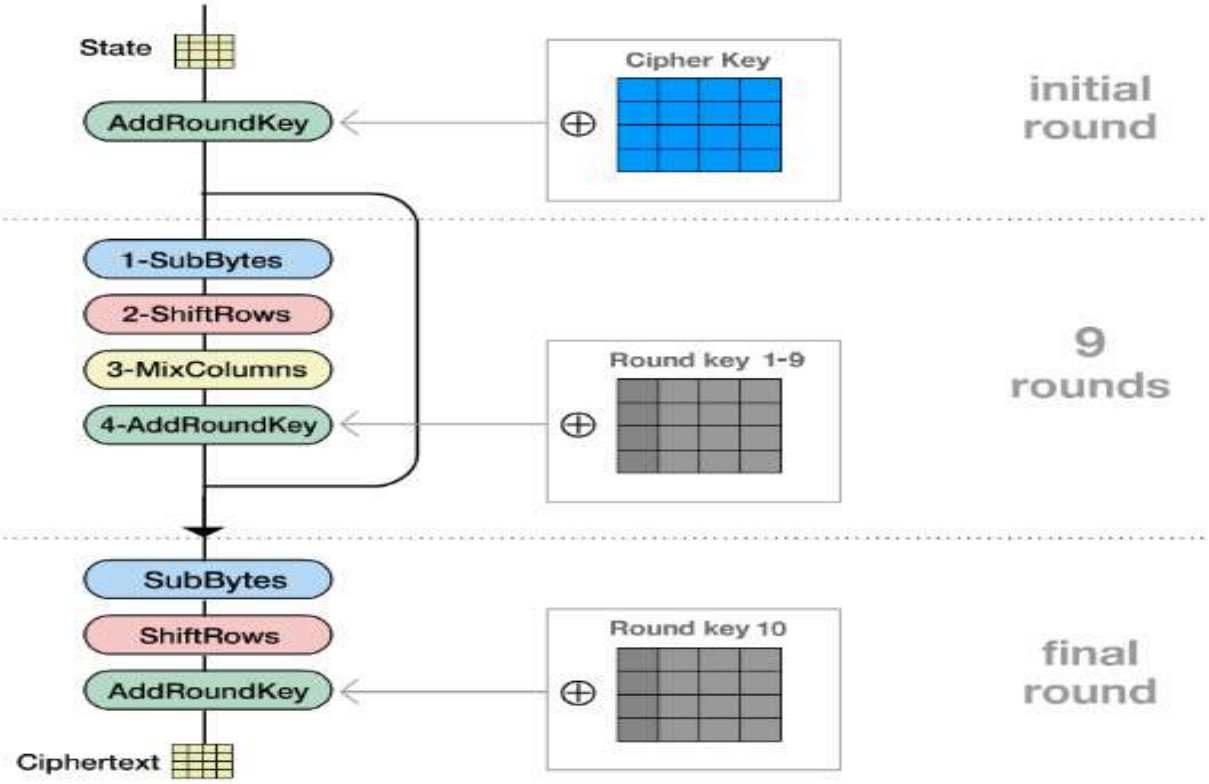



Figure I-9 : Le chiffrement.

Au début du Chiffre, l'entrée est copiée à la collection de l'État qui utilise les conventions décrite dans :

$$out[r + 4c] = s[r, c] \qquad \text{for } 0 \le r < 4 \text{ and } 0 \le c < Nb.$$

Après une addition de la Clef Ronde initiale, la collection de l'État est transformée en rendant effectif une fonction 10 ronde, 12, ou 14 fois (selon la longueur clé), avec le finale qui est différent des premiers ronds Nr-1 légèrement. La fonction ronde est paramétré qui utilisent un programme clé qui consiste en une collection unidimensionnelle de mots de quatre multiplets ont dérivé utiliser la routine de l'Expansion Clé décrit dans :

Figure I-10 : illustre l'effet du SubBytes () transformation sur l'État.

Les transformations individuelles - SubBytes (), ShiftRows (), MixColumns (), et AddRoundKey () –processus l'État et est décrit dans les subdivisions suivantes.

### SubByte

Les octets sont transformés en appliquant une S-Box inversible (afin de permettre un déchiffrement unique). Une seule S-Box est suffisante pour toute la phase de chiffrement.

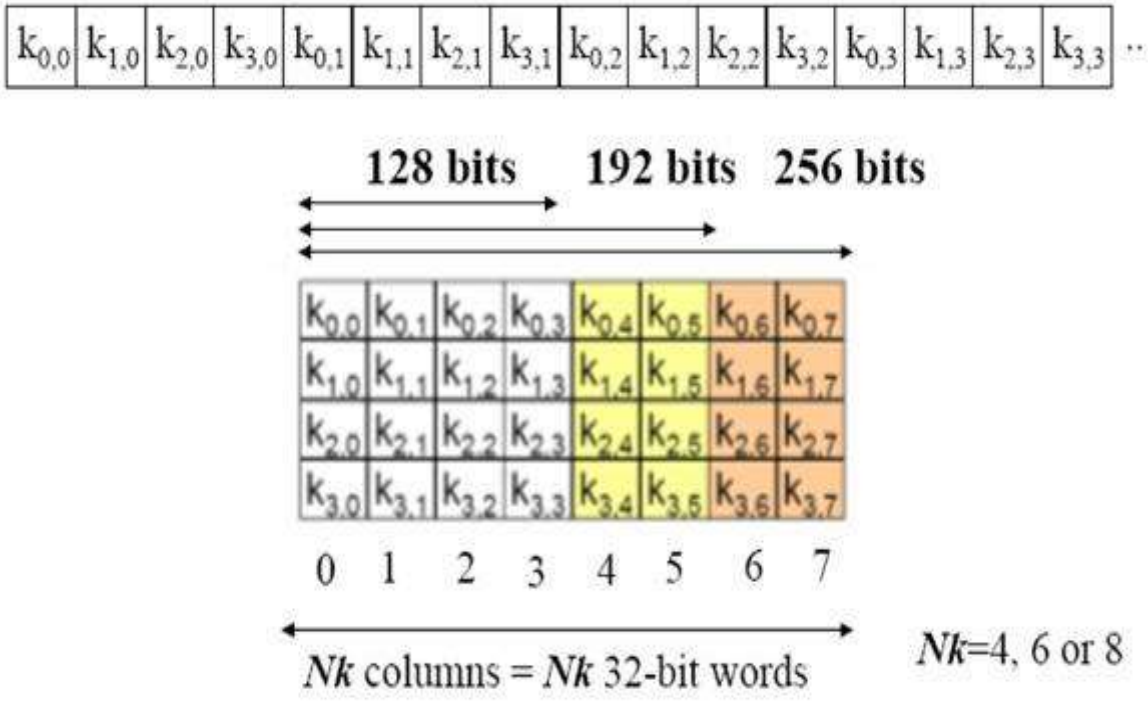



Figure I-11:Table d'état des clés. **[2]**

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

Figure I-12 : Les S-box: la substitution évalue pour le xy du multiplet (format hexadécimal).

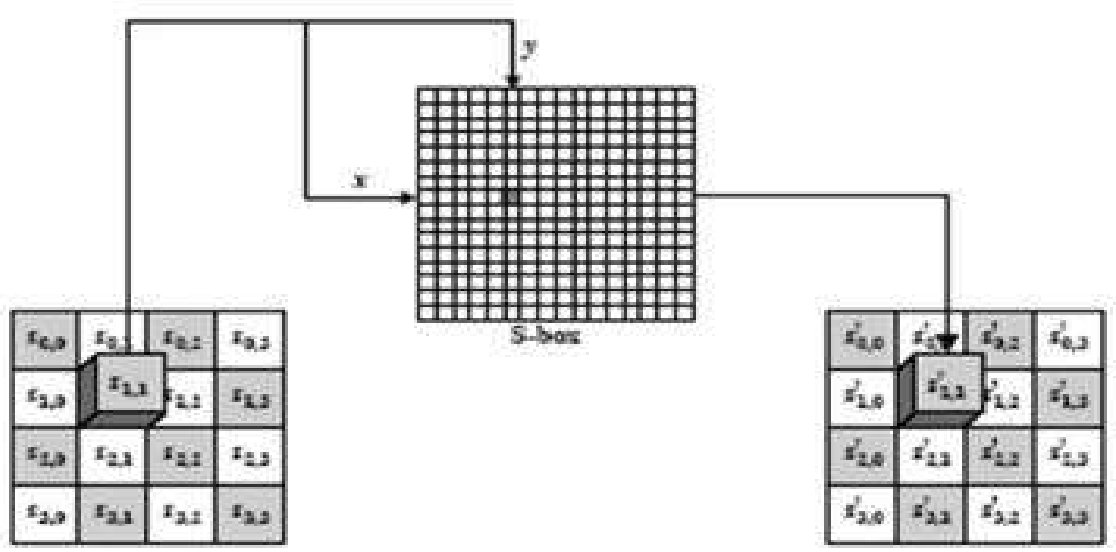



Figure I-13: Fonction SubByte applique S-box sur chaque multiplet de l'état. **[2]**

### ShiftRow

Cette étape augmente la diffusion dans la ronde.

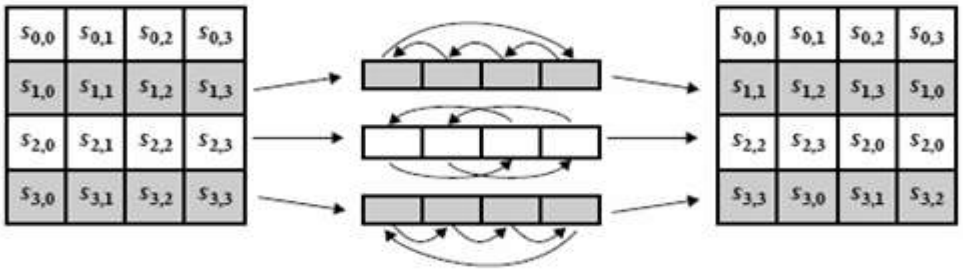



Figure I-14: Schéma de l'étape ShiftRow. **[2]**

Selon la taille des blocs de message (c'est-à-dire la valeur de Nb), les décalages ne seront pas toujours identiques.

– La ligne 0 n'est jamais décalée,

– La ligne 1 est décalée de C1,

– La ligne 2 est décalée de C2,

– La ligne 3 est décalée de C3.

| | $C_1$ | $C_2$ | $C_3$ |
|---|---|---|---|
| $N_B$=4 | 1 | 2 | 3 |
| $N_B$=6 | 1 | 2 | 3 |
| $N_B$=8 | 1 | 3 | 4 |

Table I-1: Décalage selon la taille des blocs de messages. **[2]**

### MixColumn

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.

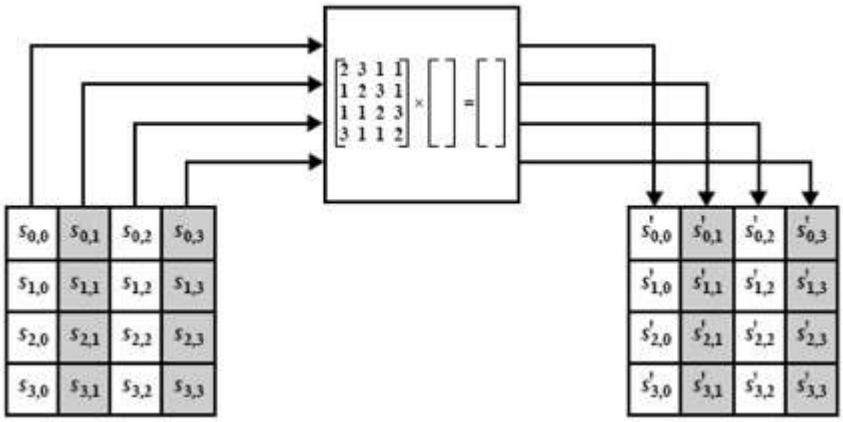


Figure I-15: Etape du MixColumn. **[2]**

### Add Round Key

C'est un simple $\oplus$ des clés. Il s'agit d'additionner des sous-clés aux sous-blocs correspondants.

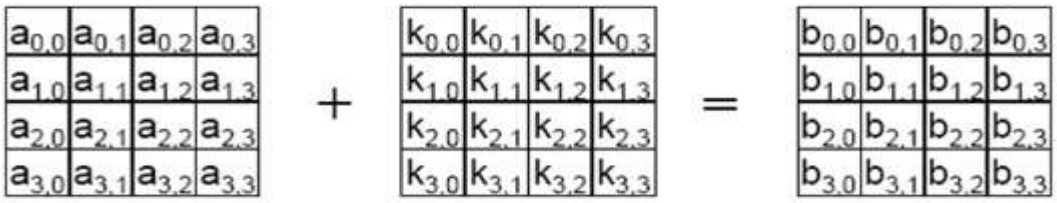


Figure I-16: AddRoundKey. **[2]**

### Nombre de rondes

Selon la taille des blocs à traiter et la taille de la clé, le nombre de rondes évolue.

| | Key length | | |
|---|---|---|---|
| **Block length** | **128 bits Nk=4** | **192 bits Nk=6** | **256 bits Nk=8** |
| **128 bits Nb=4** | **10** | **12** | **14** |
| **192 bits Nb=6** | **12** | **12** | **14** |
| **256 bits Nb=8** | **14** | **14** | **14** |

Table I-2: Nombres de rondes à effectuer. **[2]**

### I.7.2.1.2. Déchiffrement : [3]

La routine de chiffrement peut être inversée et réordonnée pour produire un algorithme de déchiffrement utilisant les transformations InvSubBytes, InvShiftRows, InvMixColumns, et AddRoundKey. Dans cette version du déchiffrement, la séquence des transformations diffère de celle du chiffrement, le traitement de la clé restant inchangé. Certaines propriétés du Rijndael permettent d'implémenter une routine de déchiffrement équivalente qui respecte la séquence de transformations de la routine AES_Encrypt, la structure de celle-ci étant la plus efficace. [1]

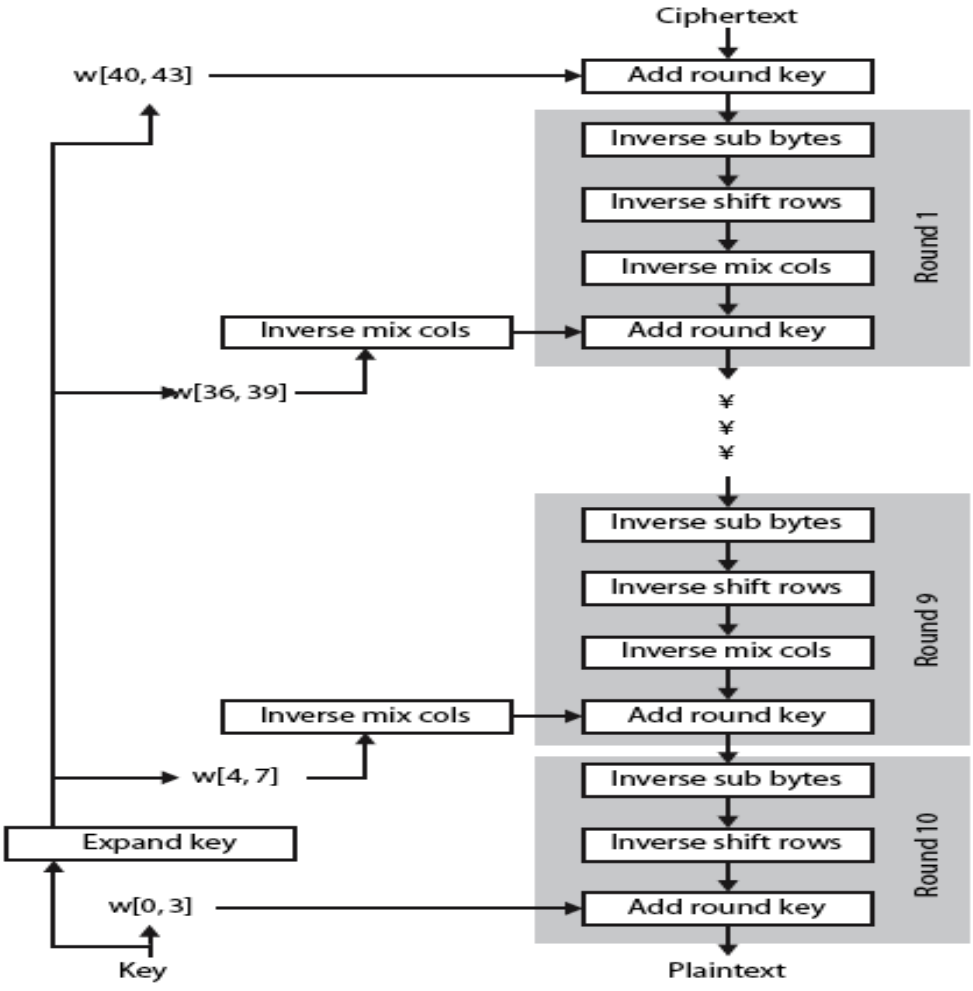



Figure I-17 : Le déchiffrement AES.

[1] https://varrette.gforge.uni.lu/download/teaching/crypto/enonce_Projet_AES_HTML/node4.html.

### Key Expansion :

L'algorithme AES prend la clé de chiffrement, K, et exécute une routine de l'Expansion Clé pour produire un programme clé. L'expansion Clé génère un total de Nb (Nr + 1) mots: l'algorithme nécessite une première série de Nb mots, et chacun des Nr tours nécessite Nb mots de données clés. Le résultant programme clé consiste en une collection linéaire de 4 multiplet rédige, a dénoté [wi], avec i dans la gamme 0 £ i <Nb (Nr +1).

### SubWord :

SubWord () est une fonction qui prend un quatre multiplet est entrée le mot et applique les S Empaquettent (Figure I-11) à chacun des quatre multiplets produire un mot de la production. La fonction RotWord () prend un mot [a0, a1, a2, a3] comme entrée, exécute une permutation cyclique, et recettes le mot [a1, a2, a3, a0]. La collection du mot constante ronde, Rcon[i], contient les valeurs données par [xi-1, {00}, {00}, {00}], avec xi-1 qui est pouvoirs de x (x est dénoté comme {02}) dans le champ GF($2^8$), comme discuté dans b(x) a(x) + m(x) c(x)=1.

### Inverse Cipher :

Les transformations du Chiffre peuvent être inversés et alors peuvent être rendus effectif dans revers ordonnez de produire un Chiffre Inverse simple pour l'algorithme AES. Les transformations individuelles ont utilisé dans le Chiffre Inverse - InvShiftRows (), InvSubBytes (), InvMixColumns (), et AddRoundKey ().

### InvShiftRows () Transformation :

InvShiftRows () est l'inverse du ShiftRows () transformation. Les multiplets dans les dernières trois lignes de l'État sont changées sur nombres différents de multiplets cycliquement (compensations). Le premier ligne, r = 0, n'est pas changée. Les trois lignes inférieures sont changées par Nb-shift(r, Nb cycliquement) multiplets où le shift (r, Nb de la valeur du changement) dépend du nombre de la ligne, et est donné dans équation :

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

La figure I-18 illustre l'InvShiftRows () transformation.

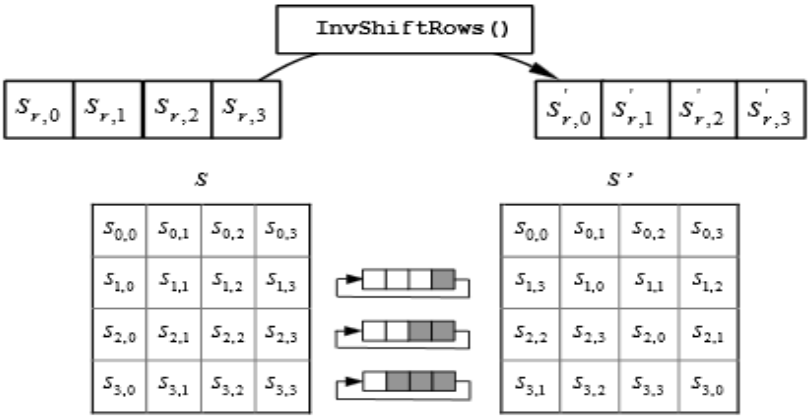



Figure I-18: InvShiftRows () cycliquement changements les dernières trois lignes dans l'État.

### InvSubBytes () Transformation :

InvSubBytes () est l'inverse de la transformation de substitution d'octets, dans lequel la boîte de S-inverse est appliquée à chaque octet de l'Etat. Cela est obtenu en appliquant l'inverse de la transformation affine :

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

, puis en prenant l'inverse multiplicatif GF ($2^8$) .Le S-box inverse a utilisé dans les InvSubBytes () .la transformation est présentée dans la figure I-19.

| | | 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 |

Figure I-19: Boîte S inverse: valeurs de substitution pour l'octet (au format hexadécimal).

### InvMixColumns () Transformation :

InvMixColumns () est l'inverse du MixColumns () transformation. InvMixColumns () opère sur la colonne de l'État colonne par colonne, en traitant chaque colonne comme un quatre terme polynôme comme décrit $b^{-1}(x) = a(x) \bmod m(x)$. Les colonnes sont considérées comme polynômes sur $GF(2^8)$ et a multiplié le modulo $x^4 + 1$ avec un polynomial $a^{-1}(x)$ fixe), donnée par $a^{-1}(x) = \{0b\}x^3 + \{0d\}x^2 + \{09\}x + \{0e\}$.

Comme décrit dans $b^{-1}(x) = a(x) \bmod m(x)$, cela peut être écrit comme une multiplication de la matrice. Laissez le $s'(x) = a^{-1}(x) \oplus s(x)$ :

$$\begin{pmatrix} S'_{0,c} \\ S'_{1,c} \\ S'_{2,c} \\ S'_{3,c} \end{pmatrix} = \begin{pmatrix} 0e & 0b & 0d & 09 \\ 09 & 0e & 0b & 0d \\ 0d & 09 & 0e & 0b \\ 0b & 0d & 09 & 0e \end{pmatrix} \bullet \begin{pmatrix} S_{0,c} \\ S_{1,c} \\ S_{2,c} \\ S_{3,c} \end{pmatrix} \text{ Pour } 0 \le c \le Nb$$

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

### I.7.2.2. Calcul de la clé :

Après avoir subi une extension (Key Expansion), la clé sera découpée en sous clés (appelées clés de rondes), comme indiqué à la figure I-20 :

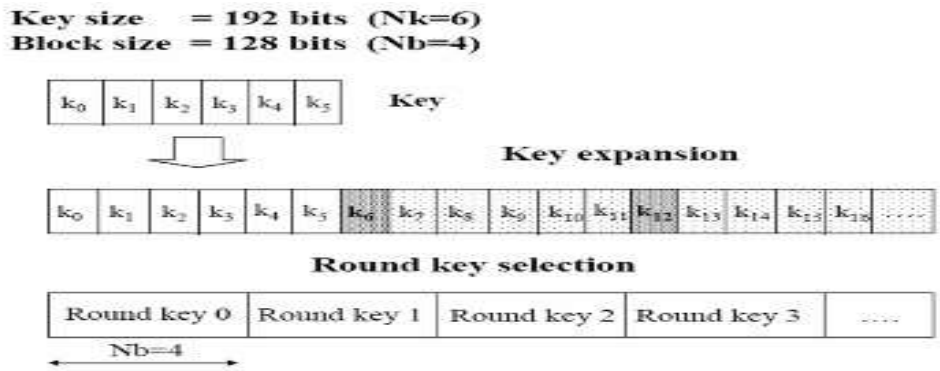



Figure I-20: Schéma des opérations effectuées sur la clé. **[2]**

Le nombre de sous-blocs ki dépendra bien sûr de la taille des clés et bloc du message.

### Extension de la clé

Le calcul de l'expansion de la clé se fait de deux manières distinctes selon le sous-bloc de la clé concerné, comme l'illustrent les figures I-21, I-22.

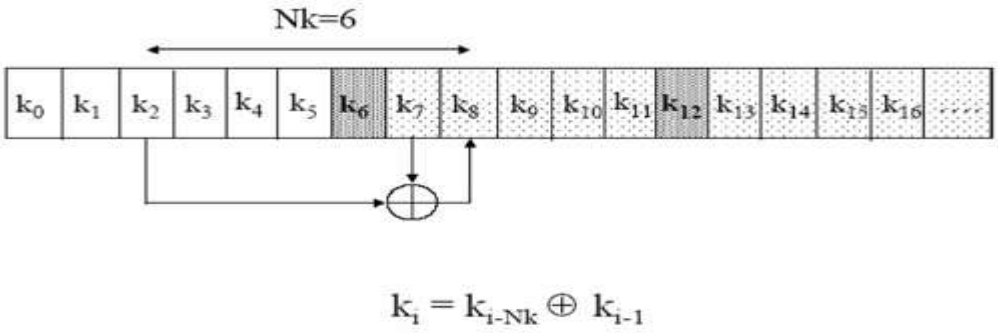



Figure I-21: Expansion de la clé avec bloc "commun".

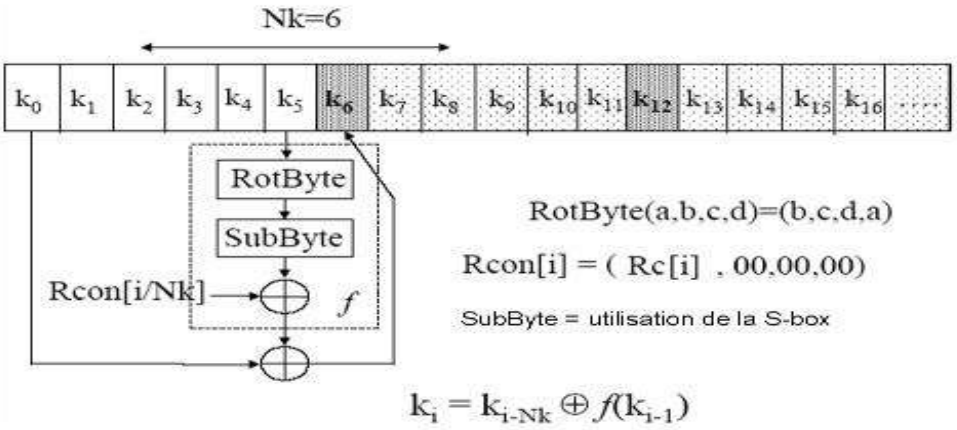



Figure I-22: Expansion de la clé avec les blocs "multiples de Nk". **[2]**

### Remarques concernant la figure I-22:

L'ajout de "Rcon[x]" donne comme résultat un $\oplus$ sur les bits les plus significatifs. La table utilisée pour donner les valeurs de Rcon [] est donnée à la figure I-22.

| j | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | ... |
|---|---|---|---|---|---|---|---|---|---|---|
| RC[j] | 01 | 02 | 04 | 08 | 10 | 20 | 40 | 80 | 1B | ... |

Table I-3: Table de correspondance des Rcon []. **[2]**

La règle de construction3 de cette table est

RC [1] = 1 RC[j] = 2.RC [j−1]

- Corps finis : GF($2^8$) :

•Cet objet mathématique est utilisé pour définir la boîte S et dans MixColumn ()

•Unique corps fini à 256 éléments

–Addition = XOR

–Multiplication = ?

Définition :

•GF(2) = F2 = unique corps fini à 2 éléments= {0,1} avec les opérations booléennes usuelles

•2[X] = ensemble des polynômes à coefficients dans F2

•Soit P(X) un polynôme irréductible de degré 8 appartenant à F2[X]

•Par définition, GF($2^8$) = F2[X] / P

•Prenons P(X) = $X^8+ X^4+ X^3+ X + 1$

•Éléments dans GF($2^8$) = polynômes réduits « modulo P»= polynômes de degré < 8

•Exemple a = $X^6+ X^4+ X^2$

•Soient a et b dans GF($2^8$)

a = $X^6+ X^4+ X^2$

b = $X^2$

•Addition:

a+b = $(X^6+ X^4+ X^2) + (X^2 ) = X^6+ X^4$

•Multiplication:

a x b = $(X^8+ X^6+ X^4)$ modulo $(X^8+ X^4+ X^3+ X + 1)$

a x b = $(X^6+ X^4) + X^4+ X^3+ X + 1$

•Chaque élément de GF($2^8$) est représenté $b7.X^7+ b6.X^6+ ... + b1X + b0$

•On le stocke sur l'octet représenté en binaire par (b7,..., b0)

•Représentation entre 0 et 255

### I.7.3. Avantages et limites

Les principaux avantages sont :

- ✓ des performances très élevées,
- ✓ la possibilité de réalisation en "Smart Card" avec peu de code,
- ✓ la possibilité de parallélisme,

- ✓ il ne comprend pas d'opérations arithmétiques : ce sont uniquement des décalages et des XOR,
- ✓ il n'utilise pas de composants d'autres cryptosystèmes,
- ✓ il n'est pas fondé sur des relations obscures entres opérations,
- ✓ le nombre de rondes peut facilement être augmenté si c'est requis,
- ✓ il ne possède pas de clés faibles,
- ✓ il est résistant à la cryptanalyse différentielle et linéaire.

Il possède pourtant quelques inconvénients et limites :

- le code et les tables sont différents pour le chiffrement et déchiffrement,
- le déchiffrement est plus difficile à implanter en "Smart Card",
- dans une réalisation matérielle, il y a peu de réutilisation des circuits de chiffrement pour effectuer le déchiffrement.

### I.7.4. Attaques

L'AES est sensible à certaines attaques par canal auxiliaire, n'impliquant donc pas directement l'algorithme.

Toutefois, en 2009, un article[2] met en évidence des problèmes de sécurité sur des versions allégées de l'AES-256. Bien que non applicables à l'algorithme standard (sur 128 bits), il est fort probable qu'à moyen terme, l'AES soit mis à mal.

## I.8. Conclusion :

Le développement de mécanismes de sécurité spécifiques aux réseaux de capteurs sans fil a attiré une grande part d'intention parmi les chercheurs de la communauté scientifique du domaine. En outre, les solutions de sécurité proposées pour les autres réseaux sans fil (ad hoc, Wifi, Bluetooth…etc.) ne sont pas applicables directement dans les RcSF. En effet, l'absence de sécurité physique et les limites en ressources énergétiques des capteurs, entre autres, s'imposent comme contraintes importantes dans la conception des solutions de sécurité efficaces et économes en énergie pour les réseaux de capteurs sans fil. Dans ce chapitre, nous avons présenté un état de l'art sur la sécurité du routage dans les RcSF. D'autre part, les moyens et les mécanismes de sécurité existants dans la littérature pour garantir un niveau de sécurité élevé pour les échanges de données sur le réseau. D'autres points abordés dans ce chapitre traitent les techniques de la cryptographie, et en particulier l'algorithme de chiffrement AES (Advanced Encryption Standard).

[2] http ://eprint.iacr.org/2009/374.

# Chapitre II :

## Conception des systèmes numériques embarqués

## II.1. Introduction

Les systèmes électroniques et informatiques sont devenus omniprésents dans la vie de tous les jours : téléphones portables, baladeurs, cartes de paiement, etc., et même dans des applications moins « voyantes » mais pas moins répandues (conduite assistée pour voitures, monitorage du trafic, contrôles d'accès, systèmes de sécurité, etc.).

L'évolution des technologies de fabrication ont permis d'augmenter non seulement la capacité d'intégration, mais aussi la fréquence de fonctionnement des composants. En pratique, l'architecture interne des processeurs se révèle de plus en plus complexe, de sorte à améliorer et accélérer l'exécution des programmes. Pour répondre aux exigences du marché, avec une diversité d'applications, l'électronique a dû progressivement s'adapter pour devenir portable, puis ultra miniaturisée : c'est la naissance des systèmes (numériques) embarqués. Ces systèmes ont vu leur importance progresser au rythme de l'importance prise par les microprocesseurs. Le marché des microprocesseurs est un marché qui croît de façon exponentielle. Ce marché a aussi tiré le marché des systèmes embarqués (et des télécommunications). En effet, selon quelques statistiques, 90% des processeurs sont destinés à l'industrie des systèmes embarqués.

## II.2. Conception des systèmes numériques :

Pendant longtemps, les calculs sont effectués par les ordinateurs et entièrement pris en charge par le processeur central (CPU), qui s'avère insuffisant dans un certain nombre de domaines. Des circuits plus efficaces ont été créés et travaillant en parallèle avec le processeur. En plus, les évolutions technologiques acquises dans le monde industriel actuel ont permis l'intégration de plus en plus de fonctions sur le même circuit intégré numérique. Aujourd'hui, on arrive à embarquer plusieurs microprocesseurs, des accélérateurs matériels, des systèmes de communications diverses, des systèmes d'exploitation, etc. Ceci permet donc de réaliser toutes les fonctions nécessaires pour effectuer des traitements informatiques complexes sur la même puce, d'où la naissance de la notion des systèmes sur puce (SoC pour System on Chip).

## II.3. Architectures à base d'un processeur

### II.3.1. Les processeurs d'usage général (GPP)

La puissance de calcul et la flexibilité sont les objectifs premiers d'un GPP. Ils sont atteints grâce à une fréquence de fonctionnement élevée et des architectures matérielles et logicielles adaptées : super scalaire, fort pipeline, prédiction de branchement, caches multi-niveaux,

instructions spécialisés (par exemple l'instruction multimédia : MMX pour le Pentium et AltiVec pour le PowerPC) et extension SIMD (Single-Instruction Multi-Data). Cependant, un GPP n'est pas bien adapté aux applications embarquées du traitement du signal et de l'image, et cela à cause de deux principales raisons : dissipation d'énergie généralement élevée dans les GPP, et les GPP ne sont pas adaptés au temps réel pour les applications complexes.

### II.3.2. Les processeurs de traitement du signal :

Certains systèmes numériques intègrent également des processeurs à usages spécifiques pour des applications de traitement du signal et des images (DSP pour Digital Signal Processors). Ces processeurs spécialisés sont surtout destinés aux traitements d'images de type «bas niveau» pour lesquels les calculs sont réguliers (les traitements sont identiques pour toutes les données d'entrée) et doivent être réalisés à hautes fréquences.

Les DSP sont des processeurs simplifiés relativement semblables aux GPP. Leur particularité essentielle est qu'ils sont conçus pour effectuer des calculs en temps réel et intègrent donc de nombreux opérateurs. En effet leur architecture interne est dimensionnée pour le calcul intensif.

### II.3.3. Les processeurs reconfigurables

En plus des GPP et des DSP, on retrouve également des processeurs reconfigurables qui peuvent intégrer des unités fonctionnelles selon le besoin. Ces processeurs peuvent être taillés sur mesure selon les besoins de l'application et de gagner par conséquent énormément de ressources. En effet, un processeur softcore (synthétisable) est un processeur implémenté sur un système reprogrammable comme un FPGA (Field Programmable Gâte Array). Si on intègre plusieurs processeurs sur la même puce, on parle alors de système sur puce programmable (System on Programmable Chip ou SoPC). Architecture très flexible, une implémentation softcore d'un processeur peut être reconfigurée en tout temps contrairement à un processeur dit hardcore qui ne peut être modifiée. Un processeur softcore s'adapte donc aux besoins de ses développeurs et aux contraintes matérielles (périphériques, performances, consommation, etc.). Toutefois, ces performances sont inférieures à celles d'un processeur hardcore. Un softcore est en général programmé dans un langage de description matérielle comme le VHDL (acronyme de VHSIC-HDL : Very High Speed Integrated Circuit Hardware Description Language) ou le Verilog. Parmi les processeurs softcore les plus connus, on peut citer : le Microblaze de Xilinx, le NIOS de la société Altera2, Openrisc d'Opencore3 et le LEON de chez Gaisler Research4.

## II.4. Les circuits logiques programmables

Un circuit logique programmable, est un circuit logique intégré qui peut être reprogrammé après sa fabrication. Il est composé de nombreuses cellules logiques élémentaires librement assemblables. Un circuit logique programmable permet la réalisation de circuits logiques à grand nombre de variables d'entrée et de sortie, selon la spécification de l'utilisateur et en utilisant un même support matériel. Actuellement, on trouve différentes familles de circuits programmables tels que les PLD (Programmable Logic Device), les CPLDs (Complex Logic Programmable Device) et les FPGAs. La figure II-1 montre les différentes familles des circuits programmables.

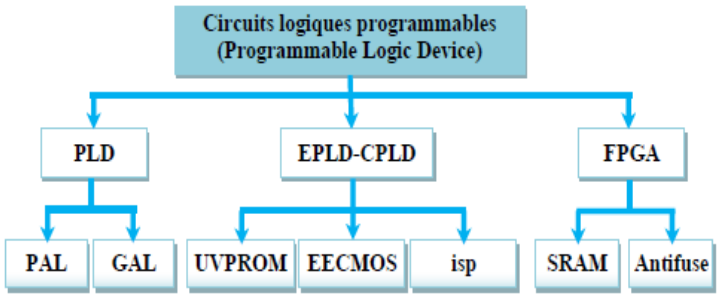



Figure II-1 : Les différents circuits programmables. **[4]**

### II.4.1. Les PLDs et les CPLDs :

Les PLDs sont proposés avec deux grandes familles : PAL (Programmable Array Logic), et GAL (Généric Array Logic). Les PLDs présentent les avantages suivants : personnalisation et mise en oeuvre simple, outils de développement peu coûteux (souvent gratuits). Les inconvénients des PLDs sont : une consommation globale accrue par les circuits de configuration, les PLDs sont peut-être chers pour de grandes séries, la performance globale est dépendante pour beaucoup de l'outil utilisé. Toujours dans un souci de performances, les PALs ont été regroupés sous le terme de SPLD (Simple Programmable Logic Devices). Pour encore augmenter les capacités de l'architecture, plusieurs SPLDs peuvent être intégrés dans la même puce pour constituer un CPLD. Avec les CPLDs, les concepteurs ont atteint la densité d'intégration maximale pour ce type de circuits, de nouvelles études ont conduit à l'apparition des FPGAs.

### II.4.2. Les FPGAs

Les FPGAs, ou bien les réseaux logiques programmables, sont des composants électroniques programmables de la famille des PLDs. Un FPGA est un ensemble de blocs logiques élémentaires que l'utilisateur peut interconnecter pour réaliser les fonctions logiques de son choix. La densité des portes est importante et sans cesse en évolution. L'avantage d'un FPGA est leur grande souplesse dans leur technologie permettant une réutilisation à volonté et en un temps très court (quelques millisecondes) dans des algorithmes différents. Le progrès technologique permet de faire des composants toujours plus rapides et à plus haute intégration, autorisant la programmation d'applications importantes.

Grâce à l'évolution des procédés de fabrication, ces composants peuvent actuellement supporter des applications complexes. Ils sont constitués d'un réseau de blocs logiques, de blocs mémoires, de blocs dédiés et d'entrées/sorties. L'ensemble est relié par un réseau d'interconnexions programmable. Les blocs logiques permettent de réaliser des opérations avec quelques variables à travers une LUT (Look Up Table). Le résultat peut être éventuellement stocké dans un registre, les blocs RAM permettent d'implanter des mémoires adressables et des FIFO (First In, First Out), les blocs dédiés permettent de réaliser facilement de nombreuses opérations de traitement (blocs DSP), de gérer l'horloge, ou des interfaces de communication (RocketIO, Ethernet, PCI Express). Les nombreux ports permettent également de connecter des périphériques de la plateforme matérielle à base de FPGA.

Les FPGA se programment grâce à leurs LUT et leur réseau d'interconnexion. La programmation se fait avec un langage de programmation hardware tel que le VHDL ou bien le Verilog. L'outil de développement transforme cette description en un fichier de configuration du FPGA en plusieurs étapes : le HDL (Hardware Design Language) doit d'abord être synthétisé (transformé en éléments logiques de base), puis les éléments doivent être placés sur le composant (placement) et enfin interconnectés (routage). La Figure II.2 montre la structure interne d'un FPGA de type matrice symétrique. Il s'agit de l'architecture que l'on retrouve dans les FPGA actuels. L'utilisateur peut programmer la fonction réalisée par chaque cellule (CLB : Configurable Logic Block). On programme aussi les interconnexions entre les cellules. Les FPGA les plus récents sont configurables en une centaine de millisecondes. Les FPGA sont utilisés pour un développement rapide et bon marché des ASIC.

### II.4.3. Les ASICs

Un ASIC est un circuit électronique dédié. Il peut regrouper sur une même puce tous les éléments actifs nécessaires à la réalisation d'un ensemble électronique. Ce composant n'est pas modifiable. Comparé à un FPGA, il offre une capacité d'intégration plus élevée, une vitesse plus rapide et un coût plus faible. Cependant, le développement d'un ASIC est très long. Il s'agit d'un circuit intégré conçu exclusivement pour le projet ou l'application qui l'utilise. Les ASICs n'ont d'intérêt que pour les grandes séries pour compenser les coûts de développement et de production. Il n'est pas possible de modifier le composant après sa production, une erreur peut provoquer la nécessité de repasser par la phase de développement et coûte par conséquent très cher.

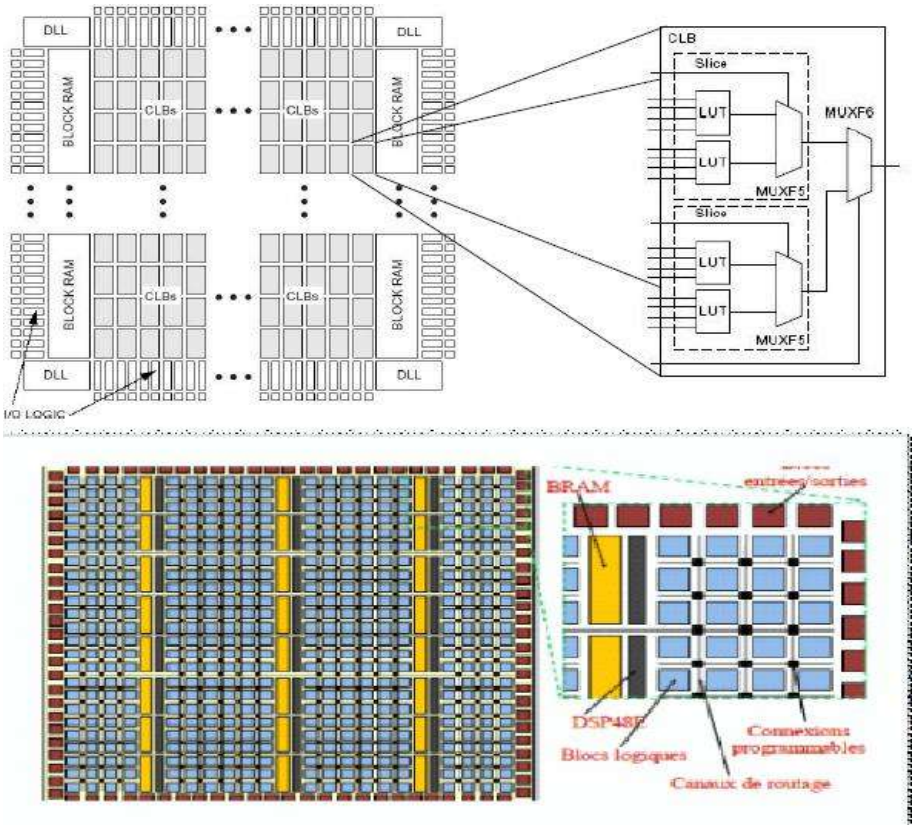



Figure II-2 : Architecture conceptuelle d'un FPGA (Architecture, Interconnexions et CLB).

**[4][7]**

### II.5. Les systèmes numériques embarqués

De nos jours, les systèmes électroniques embarqués ont été introduits dans de nombreux domaines d'applications tels que l'automobile, l'avionique, les systèmes multimédia, les appareils électroménagers, les terminaux de communication sans fil, etc. Les concepteurs doivent donc gérer les développements de systèmes électroniques embarqués

multidisciplinaires et multiprocesseurs. L'intégration électronique permettra, à titre d'exemple, de combiner les fonctions d'un téléphone, d'un navigateur Internet, d'un appareil photo numérique, d'un écran couleur, de lecteurs multimédia et d'un PDA (Personal Digital Assistants) au sein d'un unique objet portable. L'architecture générale d'un système sur puce intègre trois principaux types de composants qui sont : les unités de traitements, les mémoires et les systèmes de communication.

### II.5.1. Codesign : Conception conjointe logicielle/matérielle :

La conception des systèmes contenant des composants hardware et software n'est pas un problème nouveau. Les techniques classiques de conception de systèmes imposent de séparer le hardware du software dans le cycle de conception. En outre, le consensus grandissant pour un modèle de systèmes informatiques «en couches» semble éviter aux développeurs software de telles préoccupations. Ainsi, les concepteurs de hardware spécifient leurs produits sans totalement apprécier les besoins, en ressources matérielles, des logiciels qui s'exécuteront dessus. De leur côté, les concepteurs de logiciels ne tirent pas suffisamment profit des ressources matérielles de la plate-forme hardware sur laquelle seront implantés leurs produits (Le software et le hardware sont deux activités qui ont toujours été menées de manières relativement indépendantes). Finalement, la phase d'intégration des deux parties nécessite souvent des modifications parfois majeures du logiciel et/ou du matériel. Cette approche présente peu de flexibilité dans l'évaluation des différentes options de conception et de répartition du hardware et du software. La tendance actuelle, principalement lorsqu'il s'agit de systèmes temps réel complexes (systèmes critiques embarqués par exemple), est de tenir compte dès la phase de spécification des interactions entre les deux parties. Ainsi, la séparation est effectuée le plus tard possible dans le cycle de conception tendant à unifier la spécification pour qu'elle englobe le hardware et le software. Cette approche est nommée "hardware/software codesign", ou "codesign". Le système est conçu à partir d'une spécification unique décrivant son architecture et/ou son comportement. Après la phase de spécification survient une phase de partitionnement du système ayant pour but de décomposer ce dernier en partitions comportant trois parties :

- Une partie matérielle implantée sous forme de circuits hardware. Cette partie peut être un IP récupéré à partir d'une bibliothèque.
- Une partie logicielle implantée sous forme d'un programme exécutable sur un processeur.
- Une interface de communication entre les deux parties.

Les partitions obtenues doivent ensuite être vérifiées et validées avant de passer à la phase de synthèse et de mise en œuvre. Un feed-back permet de revenir à la phase de partitionnement en affinant différemment les poids associés aux contraintes si les partitions obtenues ne s'avèrent pas satisfaisantes.

### II.5.2. Flot de la conception conjointe logicielle/matérielle

Le développement de systèmes complexes fait de plus en plus appel à des composants logiciels et spécifiques. La conception conjointe logicielle/matérielle apporte des solutions à ce type de développement. Un flot de conception standard est généralement constitué de trois étapes principales : la spécification, le partitionnement, et la vérification conjointe HW/SW (Figure II-3).

D'après cette figure, les étapes principales peuvent être résumées comme suit :

**1.** Le point de départ est une spécification au niveau du système de l'application à réaliser. Cette spécification permet de décrire le comportement fonctionnel du système sans tenir compte de l'aspect architectural de celui-ci. A ce niveau de la conception, on ne pourra en aucun cas obtenir des résultats pertinents au niveau des performances et timing.

**2.** L'étape qui suit la vérification fonctionnelle du modèle au niveau système, sera donc le partitionnement logiciel/matériel. A ce stade de conception, on intègre les détails architecturaux de communication et d'ordonnancement des opérations. Plusieurs tentatives de simulation seront effectuées pour choisir la meilleure répartition entre les parties logicielles et les parties matérielles.

**3.** Après le découpage logiciel/matériel de la spécification d'entrée, l'étape de synthèse appelée aussi d'implémentation aura lieu. Dans cette étape, nous obtenons une description au niveau RTL pour la partie matérielle et un code source (e.g. C ou C++) pour la partie logicielle du système. Bien évidemment, la vérification et la validation de la fonctionnalité du modèle ainsi généré devrait être faite. A ce stade de conception, nous pouvons effectuer des analyses de performances de l'architecture au niveau cycle et au niveau bit à travers des Co-simulations.

**4.** La dernière étape du flot de conception consiste en premier lieu à la synthèse logique de la partie RTL du système. Ensuite les fonctions logiques qui ont été synthétisées seront placées et routées sur le plan de la puce. Ce processus est réalisé à l'aide des outils de synthèses commerciaux comme par exemple : Synplify, XST, Leonardo Spectrum, etc. Quant à la partie logicielle du système, elle sera compilée pour générer une image hexadécimale. Enfin, une Co_simulation sera établie après la synthèse logique afin d'obtenir les performances en

surface et en consommation d'énergie. Une fois l'architecture est validée, nous passons aux tests réels sur les cartes reconfigurables de type FPGA.

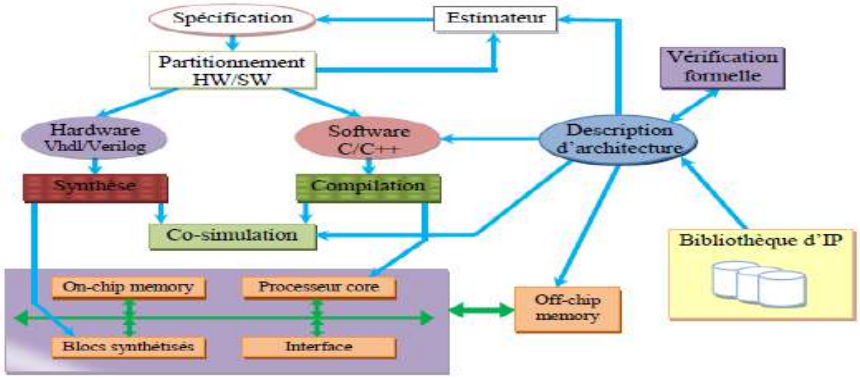



Figure II-3 : Flot de conception standard. [4]

Plusieurs outils et compagnies ont adopté cette approche de codesign par le fait de sa simplicité de synthèse et d'intégration de nouveaux composants. Parmi ces outils de développement notons celui proposé par Xilinx (EDK pour Embedded Development Kit).

### II.5.3. Les composants virtuels (IPs) :

Les outils de développement et de conception, des circuits spécialisés, sont de plus en plus performants malgré l'augmentation considérable de la capacité des composants matériels. Cependant il est indispensable de réutiliser des conceptions déjà réalisées dans ces outils de développement. En effet, afin de réduire le temps de conception et de prototypage des systèmes sur puce, une approche de réutilisation des composants a été introduite. Cette approche consiste à reprendre des modèles de composants au niveau RTL qui sont considérés comme étant des propriétés intellectuelles (IP). Les IPs peuvent être définis comme des composants virtuels, ce sont des circuits génériques ainsi que des outils associés que l'on peut trouver sur le marché ou bien sous une forme téléchargeable (Open Source).

Un IP peut être une fonction de base ou bien un bloc plus complexe. L'intégration de cœurs de processeurs est également possible, tel que le MicroBlaze ou bien le NIOS. Ces processeurs sont relativement simples, et personnalisables en fonction de l'application. L'assemblage des IPs permet de décrire rapidement des applications complexes. On peut utiliser des IP sans les avoir conçus, en ayant uniquement une description haut-niveau (vue externe et fonctionnelle) de leur comportement. Un composant IP peut apparaître sous différentes formes :

- **IP Logiciel** (softcore) : Le composant est livré sous sa forme HDL synthétisable. Son principal avantage est sa portabilité. La propriété du fichier source est en soi la meilleure documentation, on peut ainsi maintenir le produit pendant des années et éventuellement modifier à volonté la source et même changer de technologie cible. L'inconvénient majeur est qu'il ne peut être prédictif en termes de superficie, puissance et temps. Le travail d'optimisation du circuit final est à la charge de l'intégrateur du système.
- **IP Matériel** (hardcore) : Dans ce cas, la description du bloc IP est au niveau porte logique, optimisée pour une technologie particulière (optimisation garantie). Cela englobe la netlist entière, le routage et l'optimisation pour une librairie technologique spécifique et un lay-out personnalisé. L'inconvénient est qu'il est moins flexible car le processus est dépendant de la technologie, par contre il a l'avantage d'être prédictif.
- **IP Firm** (firmcore) : Le bloc IP firmcore offre un compromis entre les deux autres IPs, plus flexible que le hardcore, plus prédictif en termes de performance et de surface que le softcore (par exemple une interface USB où la vitesse de la norme ne pourra être assurée que dans une technologie bien définie). En général, le travail de synthèse HDL est déjà réalisé pour une technologie cible donnant lieu à une description par netlist (format EDIF par exemple).

Dans un système complexe, les IPs communicants à travers un réseau de communication sont aussi susceptibles d'être réutilisables après reconfiguration. La conception se résume à l'assemblage de l'ensemble des unités de calculs qui ont été choisies au préalable et les faire communiquer à travers un réseau de communication. L'inconvénient majeur de cette approche réside dans le fait que le concepteur est confronté très souvent aux problèmes d'adaptation des entrées/sorties des composants réutilisés avec le réseau d'interconnexions. Ce problème d'adaptation amène le concepteur à intégrer des adaptateurs de bus autrement dit, des interfaces de bus entre le réseau de communication et les composants du système.

### II.5.4. Les accélérateurs matériels

Un accélérateur matériel consiste à confier une fonction spécifique effectuée par le processeur à un circuit dédié qui effectuera cette fonction de façon plus efficace. En effet, les accélérateurs matériels représentent des unités de calculs intensifs. Ils peuvent être des composants matériels issus de la synthèse de haut niveau (HLS), des propriétés intellectuelles (IPs) réutilisables ou bien un composant matériel conçu spécialement pour réaliser une tâche

précise. Les accélérateurs matériels peuvent intégrer des fonctions de calculs (FFT, encodeur vidéo, etc.), des contrôleurs d'entrée/sortie (UART, VGA, PS2, etc.), des réseaux de communication (bus partagé, bus matriciel, réseau sur puce, etc.) et également des mémoires qui peuvent être considérées comme étant des IPs. Plusieurs méthodologies de conception des systèmes embarqués ont été proposées pour la réutilisation des IPs. Sur le site **[6]** plusieurs accélérateurs matériels sont proposés, allant d'une simple fonction jusqu'aux processeurs reconfigurable et même des accélérateurs pour des systèmes de traitement complets (exemple la compression d'image JPEG). Ces blocs IPs sont open source et peuvent ainsi être modifiés à volonté.

## II.6. Les systèmes sur puce (SoC)

### II.6.1. Définitions d'un SoC

Un système embarqué est système électronique et informatique autonome ne possédant pas des entrées/sorties standards comme un clavier ou un écran d'ordinateur. Une définition plus appropriée de système mono puce serait : un système complet sur une seule pièce de silicium, résultant de la cohabitation de nombreuses fonctions déjà complexes en elles-mêmes : processeurs, DSP, mémoires, bus, convertisseurs, blocs analogiques, etc. Il doit comporter au minimum une unité de traitement de logiciel (un processeur qui exécute un logiciel dédié pour la réalisation d'une fonctionnalité précise) et doit dépendre d'aucun (ou de très peu) des composants externes pour exécuter sa tâche, en conséquence, il nécessite à la fois du matériel et du logiciel. En plus un système embarqué n'exécute pas une application scientifique ou commerciale. L'objectif est diminué au minimum le nombre de composants sur une carte pour mettre tout sur une seule puce.

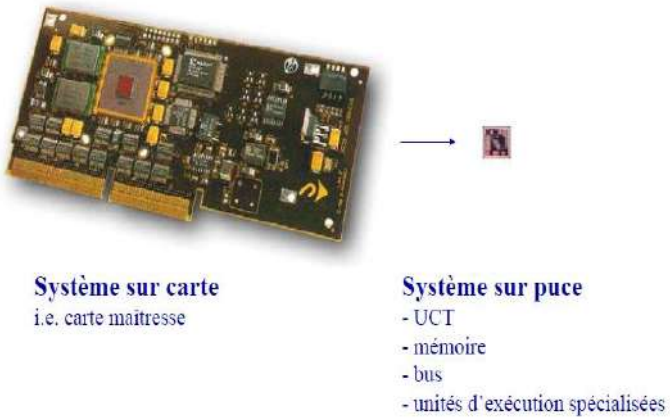



Figure II-4 : Objectif du SoC.

### II.6.2. Les contraintes liées aux systèmes embarqués

Les contraintes lors de la construction d'un système embarqué sont relatives au temps, imposés par l'interaction du système avec son environnement et aussi au caractère limité et/ou périssable des ressources dont dispose le système (énergie, mémoire, etc.). A ces contraintes s'ajoutent des contraintes de sûreté de fonctionnement plus ou moins importantes selon le type d'utilisation du système embarqué. En plus, un système embarqué exécute des tâches prédéfinies et a un cahier des charges contraignant à remplir, qui peut être de l'ordre de coût, d'espace compté ou bien les contraintes physiques (le système doit présenter un faible encombrement et un faible poids) ; de puissance de calcul ; de consommation énergétique (la plus faible possible), etc. Des choix doivent donc être réalisés pour privilégier un ou plusieurs de ces critères. Par exemple, pour un ordinateur portable, l'autonomie est souvent négligée par rapport aux contraintes de performances et de taille. En effet, un ordinateur peu performant perd tout son intérêt, d'autant plus qu'il est souvent possible de travailler à proximité d'une prise de courant. Pour un téléphone portable au contraire, l'intérêt étant de pouvoir téléphoner de partout, l'autonomie est un critère primordial.

### II.6.3. Les SoC et les SoPC

L'approche SoC a été créée dans un premier temps pour le développement d'ASIC mais a été étendue pour le développement de FPGA. On parle alors de SoPC pour System on Programmable Chip. Le SoPC (acronyme a été inventé par Synopsys) correspond à l'intégration d'un ou plusieurs cœurs de processeurs et de ses périphériques sur une même puce programmable de type FPGA. Le logiciel est alors situé soit dans une mémoire du circuit FPGA si l'empreinte mémoire le permet, soit dans une mémoire externe le plus souvent **[7]**. Les deux approches peuvent être résumées à :

- ✓ **L'approche SoC** (généralement en technologie ASIC) : répond aux besoins de performances et d'intégration mais elle est peu adaptée à l'évolutivité des systèmes et reste réservée aux grands volumes de production. En effet, un SoC peut être retenu pour les applications destinées au grand public. Il permet de meilleures performances en termes de consommation, de vitesse et de surface, mais, la fabrication et le test sont des étapes longues et coûteuses. De plus, un SoC est figé et n'est donc pas réutilisable pour une autre application.

- ✓ **L'approche SoPC** (technologie FPGA) : Spécialement pour résoudre les problèmes de développement et les prototypages rapides, ainsi que les composants reconfigurables en quelques « ms » et à volonté. Il permet donc un développement et

prototypage rapide du système. En contrepartie, la densité d'intégration est généralement moindre (~10 Millions de portes) et la consommation d'énergie est plus grande avec une performance plus faible que celle du SoC.

Les composants FPGA permettent d'implanter deux types de SoPC. Le premier type de SoPC utilise uniquement des IP soft de processeur tels que le MicroBlaze et le NIOS. Le nombre de processeurs intégrés sur un même composant dépend des ressources du composant utilisé. Le deuxième type de composants en plus de la logique reconfigurable, implémente en dur un ou plusieurs cœurs de processeurs (PowerPC pour Xilinx, ARM pour Altera). Ceci n'empêche en rien l'utilisation de processeurs Soft (netlist). Ces deux types de SoPC utilisent les mêmes flots de conception. Chez Xilinx, tout le flot de conception est regroupé sous la forme d'un seul environnement dit Xilinx Platform Studio (XPS).

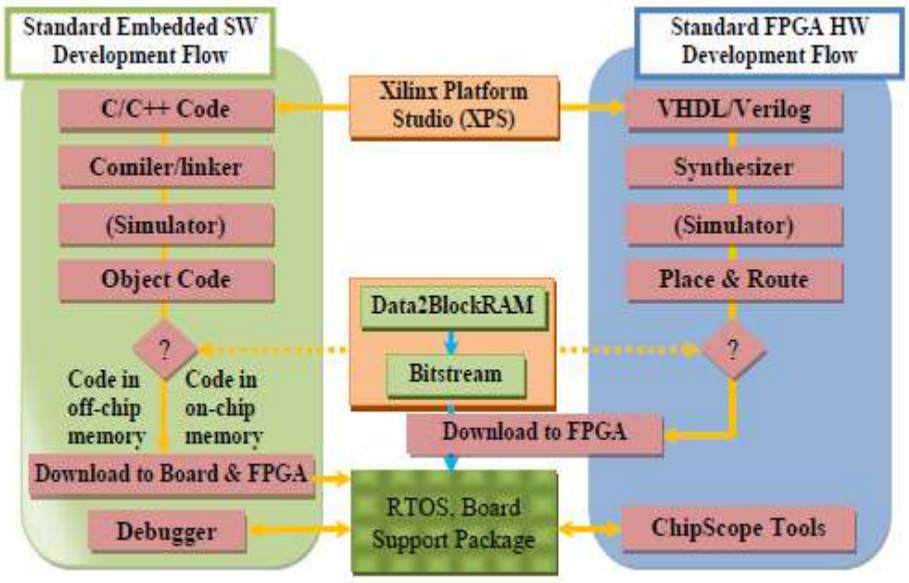



Figure II-5 : Flot de conception Xilinx. **[4]**

## II.7. Communications sur puce

Les systèmes logiques sur puce sont souvent conçus de manière à être dimensionnés pour une seule application particulière. Chaque application a sa propre architecture, de ce fait l'extension ou la réutilisation de telle architecture est difficile et nécessite un effort de réadaptation et pour cette raison plusieurs types de communication sur puce ont été proposés. Un système de communication permet d'établir des connexions entre tous les composants dans un SoC. Il se présente sous forme de bus ou bien de réseau de communication.

### II.7.1. Exemple : Le Bus CoreConnect

Spécialement pour des systèmes sur puce, IBM a proposé l'architecture CoreConnect composée de trois bus qui permettent la connexion : de cœurs de processeur, de logique dédiée, et différents autres accélérateurs matériels. Les trois bus de CoreConnect sont :

- **Processor Local Bus (PLB)** : Généralement, le PLB fournit un chemin de données large bande. En effet il est utilisé pour connecter des composants tels que les cœurs de processeurs, les interfaces avec les mémoires externes, et les contrôleurs DMA (Direct Memory Accès).
- **On-chip Peripheral Bus (OPB)** : L'OPB est utilisé pour réduire la charge du bus PLB. Il est plus adapté aux composants tels que les ports série, les ports parallèles, UARTs, GPIO et tous les composants à faible largeur de bande. Un Maître sur le PLB peut accéder aux composants connectés à l'OPB via une macro passerelle (Bridge). Cette passerelle est vue en tant que maître par l'OPB et en tant qu'esclave par le PLB. Les registres d'état et de configuration de faible performance sont généralement lus et écrits via le bus DCR.
- **Device Control Register (DCR)** : Le DCR est principalement utilisé pour accéder aux registres de l'état et de contrôle au sein des différents bus PLB et OPB.

Xilinx a repris le comportement de tous les bus CoreConnect d'IBM et les a décrits en VHDL, ce qui permet d'avoir des bus paramétrables. En effet, pour simplifier la connexion d'un module logique utilisateur à un des bus CoreConnect, Xilinx a ajouté une interface de bus préconçue paramétrable appelée IPIF (IP InterFace). L'IPIF gère les signaux du bus, le protocole de communication et plus généralement, l'ensemble des caractéristiques du bus. L'IPIF intègre aussi une interface de gestion, du logique utilisateur, appelée IPIC (IP InterConnect). Lorsque le logique utilisateur est conçu avec un IPIC, elle peut être portable et facilement réutilisable sur d'autres bus en changeant seulement l'IPIF. Des fichiers VHDL qui instancient l'IPIF et fournissent le code nécessaire à l'utilisateur pour ajouter ses modules, simplifient la tâche de connexion du logique utilisateur. Ces fichiers sont les User Core Reference Design. Un système de traitement réalisé par EDK doit être connecté le bus OPB et/ou le bus PLB, le périphérique ajouté au système (accélérateur matériel) doit être connecté à OPB ou bien PLB. La figure II-7 montre la connexion du bus PLB dans un système à base d'un MicroBlaze, d'une DDR2 et d'une mémoire Compact Flash.

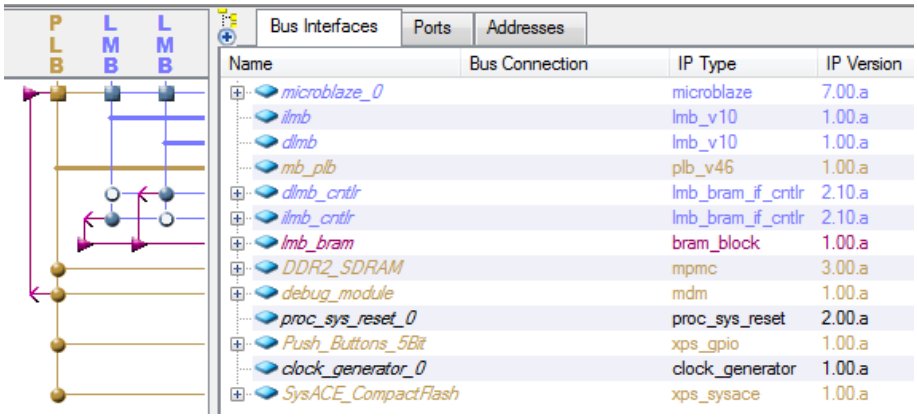



Figure II-6 : Le bus PLB dans un système à base de MicroBlaze. **[4]**

## II.8. Plateformes multi-composants :

La maîtrise des Socs nécessite des compétences en électronique et en informatique d'un niveau assez élevé. Pour aider les nouveaux utilisateurs de SoCs à bien maîtriser l'ensemble de la chaîne de mise en place d'applications, des plateformes de prototypage rapide existent aujourd'hui sur le marché. Ces plateformes utilisent un gros FPGA comme cœur et disposent, en plus, de mémoires SRAM, de convertisseurs A/N et N/A, des interfaces (boutons poussoirs, voyants LED, connecteurs d'usage général, connecteurs pour debug, zone de prototypage, connecteur RS232 ainsi qu'un connecteur JTAG), d'oscillateurs, sortie VGA, etc. Généralement, la configuration des composants se fait par liaison JTAG.

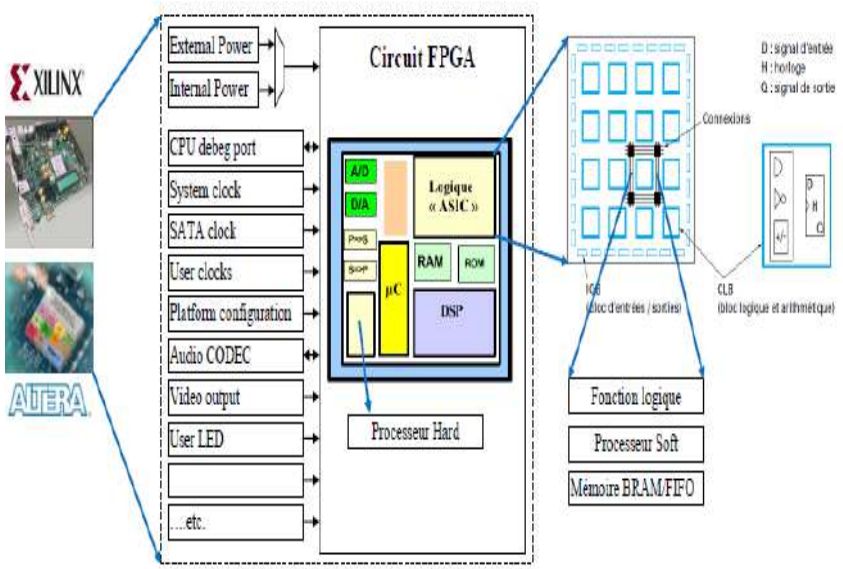



Figure II-7 : Détails d'une plateforme multi-composants. **[4]**

### II.8.1 Les deux grandes familles des FPGA :

#### II.8.1.1 La famille Xilinx

Xilinx, Inc. (1984) est une entreprise américaine de semi-conducteurs. Inventeur du FPGA avec un premier produit en 1985, Xilinx fait partie des plus grandes entreprises spécialisées dans le développement et la commercialisation de composants logiques programmables, et des services associés tels que les logiciels de CAO électroniques ; création des blocs IP et formation. En effet, Xilinx vend également des spécifications d'architectures ("IP Corés"). Cela peut aller de fonctions très simples comme des compteurs jusqu'à des systèmes complets comme des microcontrôleurs. Xilinx est également à l'origine du processeur softcore MicroBlaze. Xilinx fabrique une large gamme de FPGA et de CPLD pour diverses applications. En effet, l'offre commerciale de Xilinx est découpée en plusieurs gammes : (FPGA hautes performances : gamme Virtex, FPGA pour la fabrication en grande série : gamme Spartan et CPLD : gammes XC9500 et Coolrunner). Les plus onéreux sont les FPGA Virtex.

Chaque CLB est constitué de 6 slices. Chaque slice contient un générateur de fonctions, une logique de traitement de la retenue (carry logic), des portes logiques arithmétiques, un large multiplexeur et deux éléments de stockages, 256bits de RAM distribuée au lieu de 64 bits dans le Virtex4, un registre de 128 bits au lieu de 64 bits dans le Virtex4. Le nombre de blocs de RAM varie entre 32 et 384 blocs.

Chaque bloc RAM a une taille de 36Kbits (18Kbits pour le Virtex4) avec une largeur de bus de données maximales de 36 bits, ces mémoires sont doubles ports (Dual Port). Le nombre maximum d'I/O utilisateur est de 960. Les composants Virtex5 possèdent aussi entre 32 et 384 DSPs (DSP48E de 25×18-bit MAC à 550 MHz), jusqu'à 12 DCM (Digital Clock Manager), jusqu'à 6 PLL (Phase Lock Loop). La catégorie FXT est la seule qui dispose d'un ou bien deux processeurs embarqués type PowerPC405 (directement inclut au sein du FPGA).

En 2010, Xilinx a lancé les FPGA-Virtex7 (avec une seule catégorieXC7VxxxT) fabriqués en technologie 28nm. Afin de satisfaire à terme les applications très exigeantes en performances de calcul et en bande passante tout en limitant la puissance consommée, Xilinx va proposer dans le courant du second semestre 2011 des circuits qui couplent plusieurs FPGA Virtex7 au sein d'un même boîtier via une technologie d'interconnexion 3D. Selon Xilinx, le boîtier Virtex7 LX2000T en technologie 28 nm sera le premier FPGA multi-puce au monde et affichera une densité de portes logiques 3,5 fois supérieure au FPGA 40 nm le plus puissant actuellement disponible.

En plus du matériel, Xilinx commercialise toute une gamme d'outils de développement pour exploiter ses composants. Les principaux outils sont :

- Xilinx ISE : Environnement de développement intégré.
- Xilinx EDK : Environnement de développement intégré ciblant les processeurs intégrés au FPGA.
- Xilinx ChipScope Pro : Outil de débogage temps-réel des circuits FPGA.

#### II.8.1.2 La famille Altéra

Le concurrent de Xilinx dans le marché des FPGA est Altera. En effet, la société américaine Altera, créée en 1983, est un fabricant de composants reprogrammables (FPGA, CPLD et ASIC) et des processeurs embarqués. Pour la gamme haute performances on trouve les FPGA : Stratix et Stratix GX (2002), Stratix II (2004), Stratix II GX (2005), Stratix III (2006) et Stratix IV et HardCopy (2008). Pour les FPGA de grande série on trouve : Cyclone (2002), Cyclone II (2004), Cyclone III (2007), Cyclone IV et Arria GX (2007), etc. Enfin, pour les CPLD, on trouve les exemples les plus utilisés : MAX 3000A et MAX 7000 (1991), MAX II (2007) et MAX V (2010). Altera est aussi à l'origine des processeurs softcorés NIOS et NIOSII ainsi que le bus Avalon. Le bus Avalon est un bus informatique destiné à l'implémentation sur du matériel programmable (FPGA). Ses principaux atouts sont la simplicité de son architecture, une conception prévue pour optimiser l'utilisation des ressources matérielles et un support multi-maître.

### II.8.2 Les processeurs embarqués

Avec l'augmentation exponentielle des performances et de la densité d'intégration des portes logiques, les cœurs de processeurs sont devenus essentiels dans le développement des SoC et SoPC (Temps de mise sur le marché réduit). Un cœur de processeur est la partie matérielle qui traite les données en fonction du décodage des instructions contenues en mémoire. La grande quantité de données à prendre en compte ainsi que les contraintes en temps souhaitées, font du cœur de processeur un élément critique d'un système. Le cœur de processeur idéal doit donc fournir le maximum de performances sur une surface la plus réduite possible. Il doit être le plus flexible possible et ne pas subir de dégradation de performances en fonctions des différents environnements rencontrés (Outils de développement, taille de la cible ou électronique adjacente).

Les cœurs de processeurs peuvent être, soit des macro-cellules spécialisées, dites processeur Hardcore (PowerPC et ARM), soit des modules synthétisables, dits processeur softcore (MicroBlaze, NIOS-II, etc.). Ces soft corés sont configurables à travers plusieurs

paramètres (profondeur du pipeline, taille du cache, présence ou non de multiplieurs, etc.). Un processeur softcore est réalisé à partir de la logique programmable présente sur la puce. Ainsi, un concepteur peut mettre autant de processeurs softcore que la puce le permet. A titre illustratif, un processeur MicroBlaze occupe seulement 4% de la surface d'un FPGA de la famille Virtex5-LX50 (composant à 28.800 cellules logiques), et 1,7% de la surface d'un FPGA de la famille Virtex5-LX110T (composant à 69.120 cellules logiques). A l'inverse, le concepteur ne choisit pas le nombre de processeurs hardcore présents dans un FPGA. L'intérêt majeur des FPGA est la possibilité de faire du prototypage rapide, la proximité des cœurs de processeurs avec de la logique programmable rend possible la spécialisation de ces processeurs pour concevoir un système sur puce.

### II.8.2.1 Les processeurs hardcore

Le cœur d'un processeur hard est décrit au niveau physique (lay-out sur silicium). Dans ce cas, l'optimisation est maximale en termes de vitesse et de surface utilisée (technologie fixe). Les performances du système sont parfaitement connues et ne varient pas d'un système à l'autre (surface, consommation, délais). Cependant, le cœur du processeur hard souffre d'une trop faible flexibilité ce qui le rend très spécifique à l'application pour laquelle il a été étudié. Par exemple, la taille des microcodes utilisés est fixe ce qui interdit l'ajout de nouvelles fonctionnalités.

a. **Le processeur PowerPC405 de Xilinx :**

Le PowerPC405 d'IBM est un processeur RISC 32 bits avec une architecture Harvard. Son architecture pipeline est de 5 étages avec une fréquence de fonctionnement de 400MHz. Il dispose de 32 registres de 32 bits, une mémoire cache d'instructions et de données de 16 KB chacune et d'une unité de gestion de mémoire (Memory Management Unit). Sur les circuits FPGA Xilinx supportant le processeur PowerPC, l'accès aux mémoires internes (Blocs RAM du FPGA) se fait à travers une unité "On-Chip memory Controllers".

Le cœur de processeur matériel PowerPC est enfoui dans quelques catégories de circuits FPGA (Virtex5-FXT par exemple), c'est à l'utilisateur de rajouter les périphériques qui sont nécessaires à son fonctionnement. Le processeur PowerPC 405 ainsi que le MicroBlaze sont exploitables grâce à l'outil EDK de Xilinx.

### II.8.2.2 Les processeurs softcore

La capacité d'intégration des FPGA permet d'y implémenter des processeurs IPs soft. Ce sont des processeurs décrits à l'aide d'un langage de description matériel (VHDL ou bien Verilog) pouvant être synthétisable. Ces processeurs reprennent exactement le fonctionnement et les performances des autres processeurs implémentés en hard, l'avantage ici réside dans leur capacité d'être paramétrée. En effet, un processeur softcore s'adapte aux besoins de ses développeurs ainsi qu'aux contraintes matérielles (périphériques, performances, ressources, consommation, etc.). Le fait de paramétrer un processeur permet d'avoir une flexibilité et une facilité à faire un compromis performance-coût accélérant ainsi le cycle de conception. Cependant, les performances d'un softcore sont, généralement, inférieures à celles d'un processeur hardcore **[4]**. Parmi les produits existant en ce moment sur le marché on peut citer :

#### a. Le processeur MicroBlaze de Xilinx

Le MicroBlaze est un microprocesseur softcore RISC, d'architecture Harvard, entièrement 32 bits. En effet, il possède :

- 32 registres internes de 32 bits,
- un bus instructions interne (ILMB),
- un bus données interne (DLMB),
- un bus d'instructions externe (IOPB),
- un bus de données externe (DOPB).

Le processeur comporte 70 options de configuration2 permettant à l'utilisateur de sélectionner ou de paramétrer les composants internes selon ses besoins :

- un pipeline à 3 ou 5 niveaux,
- utilisation des multiplieurs câblés du FPGA (sauf sur Spartan II)
- opérateur de division
- opérateur de décalage (Barrel Shifter)
- FPU (Floating Point Unit)
- mémoires cache instructions et données
- logique de débug (hardware breakpoints)

En 2006, il occupait entre 900 et 2600 'logic cells' et atteignait une fréquence maximale de 80 à 180 MHz selon la plateforme FPGA et les options sélectionnées.

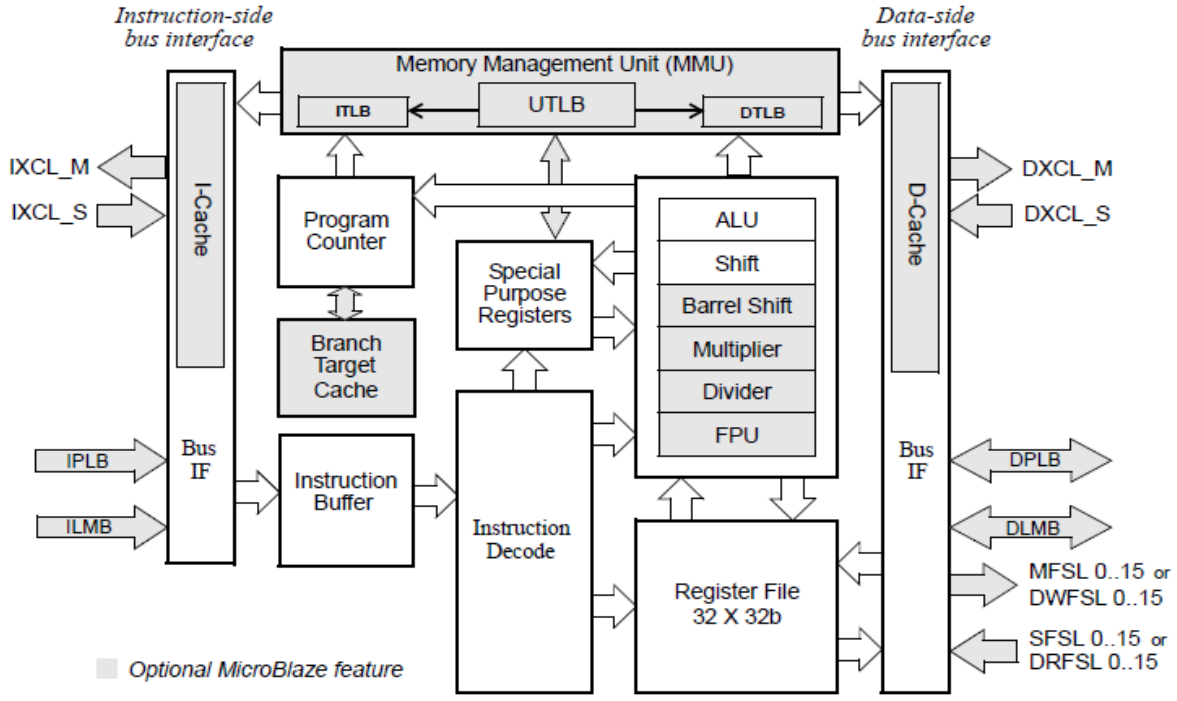



Figure II-8 : Schéma-bloc fonctionnel du noyau MicroBlaze. **[5]**

### b. Les processeurs NIOS et NIOS-II de Altera

Le NIOS est un Processeur softcore propriétaire de la société Altera. Il est basé sur un cœur RISC 32 bits et doté du bus Avalon. Le NIOS-II est la deuxième génération de processeurs embarqués à cœur logiciel d'Altera, il annonce déjà une nouvelle aire pour l'électronique et l'informatique embarqué. NIOS-II incorpore de nombreuses améliorations sur l'architecture d'origine NIOS, le rendant plus adapté pour une large gamme d'applications d'électronique embarqué et systèmes de contrôle.

Contrairement au MicroBlaze, Synopsys DesignWare offre une licence de NIOS-II pour les standardcell ASICs, et grâce à cette licence, les concepteurs peuvent utiliser NIOS pour une production de masse d'ASIC. Pour bien gérer les applications à base de processeur NIOS, Altera a développé trois principaux outils de développement :

- L'outil Quartus pour l'implémentation du NIOS dans un FPGA ;

- Le développement du cœur et de ses composants ("Briques IP") s'effectue à l'aide du "SOPC Builder" ;
- Le développement du logiciel s'effectue en C sous NIOS-IDE (Windows) ou en utilisant nios2- toolchain sous Linux.

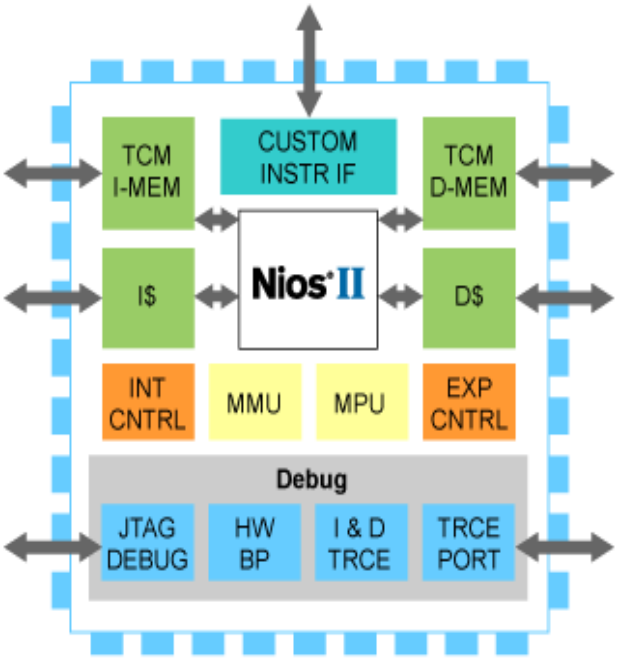



Figure II-9 : Les processeurs NIOS et NIOS-II de Altera. **[4]**

### II.8.3. Les processeurs softcore Open Source

#### a. Le processeur LEON

Le processeur Léon, développé par l'Agence Spatiale Européenne (ESA), emploie une architecture interne de 32 bits, entièrement décrite en VHDL et complètement synthétisable. Son architecture est de type SPARC-V8 (Scalable Processor ARChitecture). Cette architecture permet des économies en nombres d'instructions de type load/store en mémoire et offre au compilateur la possibilité de produire un code plus efficace. En effet, comparé à d'autres types d'architectures RISC, le compilateur possède une plus grande flexibilité d'affectation des registres aux variables de l'application. Pour des programmes complexes décrits avec des langages à haut niveau d'abstraction (type C++), la réduction en termes de nombres d'instructions exécutées est significative.

### II.8.4 Les Outils de développement de Xilinx [4]

En plus du matériel, Xilinx fourni des outils de conception des systèmes nous permettant de mettre au point des projets sur leurs différentes plateformes reconfigurables. En effet, Xilinx fournit dans son pack d'outils différents soft permettant la création de systèmes embarqués sur puce, parmi ces softs on dénombre ISE (Integrated Software Environment) et EDK. L'outil ISE est spécialement pour la réalisation des projets d'IPs matériels à partir d'un code HDL. L'outil EDK, englobant SYSTEM GENERATOR FOR PROCESSOR et Xilinx Platform Studio (XPS), va nous permettre d'établir un lien direct entre la partie matérielle et logicielle d'un système. Tous les deux nous offrent la possibilité d'avoir un bitstream pour la programmation des FPGA suivant l'application ciblée.

#### II.8.4.1 Xilinx ISE

ISE est un environnement intégré de développement de systèmes numériques ayant pour but la synthèse/implémentation matérielle sur FPGA. Les designs peuvent être décrits sous trois formes principales : sous forme de schémas, sous forme de HDL ou bien sous forme de diagrammes d'états. Pour cela, ISE intègre différents outils permettant de passer à travers tout le flot de conception d'un système numérique. Il inclut des éditeurs et outils pour la saisie du système, il dispose aussi d'un ensemble d'outils, de synthèse et d'implémentation, regroupés dans un seul flot de conception. Les détails de ces outils sont donnés comme suit :

1. Saisie de conception (Design Entry) : Editeur HDL, éditeur de Machine à Etat Fini (StateCAD), éditeur de schéma (Engineering Capture System (ECS)) et générateur d'IP (CORE Generator) ;
2. Synthèse : XST (Xilinx Synthesis Technology), Leonardo Spectrum de Mentor Graphics et Synplify de Cadence ;
3. Simulation : Générateur de TestBench HDL et intégration du simulateur ModelSim de Model

Technology ;

4. Implémentation : Translate, MAP, placement et routage (PAR), floor planner, éditeur FPGA, timing Analyzer XPower, Fit (seulement CPLD) et Chipviewer (seulement CPLD) ;
5. Programmation du composant et formatage du fichier de Bitstream : BitGen et iMPACT.

#### II.8.4.2 Xilinx EDK

L'environnement EDK permet de développer une application complète à base de processeur embarqué et de l'intégrer à un FPGA. EDK donne accès à tous les réglages

nécessaires pour l'application embarquée que l'on souhaite créer. Il permet la programmation d'un ou plusieurs MicroBlazes et de leurs périphériques. Il va s'occuper de placer les programmes des MicroBlazes en mémoire lors de la programmation du FPGA.

EDK permet d'une part la création des accélérateurs matériels au niveau RTL à partir d'une description en VHDL (il est possible d'ajouter directement le code synthétisé avec une extension .ngc), et d'autre part de générer l'architecture globale du système en incluant les processeurs logiciels (CPU) ainsi que les contrôleurs d'entrée/sortie nécessaires. L'ensemble de ces composants vont être interconnectés via un bus, qui est lui aussi généré en fonction des besoins architecturaux de l'application. EDK permet aussi :

- Génération du logiciel : permet l'extraction des parties logicielles de l'application et de générer par la suite le code machine correspondant à chaque processeur de la plateforme matérielle.
- Génération des communications : permet de générer le système de communication de l'architecture en fonction du nombre de composants constituants la plateforme matérielle.

## II.9 Conclusion :

Les systèmes sur puce (SoC) et les systèmes multiprocesseurs sur puce (MPSoC) sont deux aspects très utilisés de nos jours, ils sont motivés par l'augmentation continue de la capacité d'intégration (à cause des progrès des technologies CMOS) d'une part, et la complexité croissante des applications embarquées d'autre part, surtout en ce qui concerne les applications du multimédia. En effet, La conception de systèmes embarqués est généralement motivée par l'évolution technologique et la généralisation du multimédia.

# Chapitre III:

## Conception et implémantation de l'algorithme AES sur le circuit FPGA

## III.1. Introduction

Dans ce projet, nous avons implémenté deux algorithmes de cryptage/décryptage AES 128bits sur un circuit FPGA de type Spartan 3E. Dans ce chapitre, il sera présenté la conception et l'implémentation matérielle-logicielle du système sur une plateforme de développement Spartan 3A DSP XC3SD1800A. Et enfin le résultat du test pratique réalisé entre le kit FPGA et un PC interfacés par le port série (RS232) (Figure III.1).

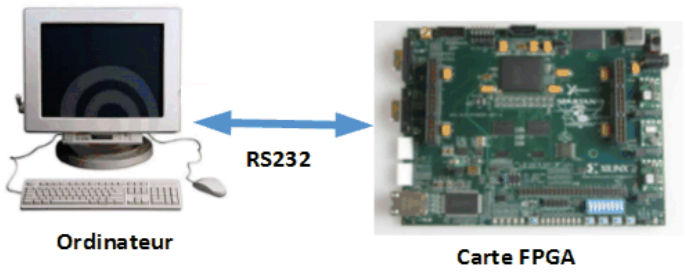



Figure III-1: système de test pratique.

## III.2. Carte FPGA utilisé :

Pour implémenter l'algorithme AES sur FPGA, nous avons utilisé la carte "Xtreme DSP Starter Kit-Spartan-3A DSP 1800A Edition", elle est présentée dans la figure III.2. Cette carte possède les caractéristiques suivantes **[9] :**

- 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)

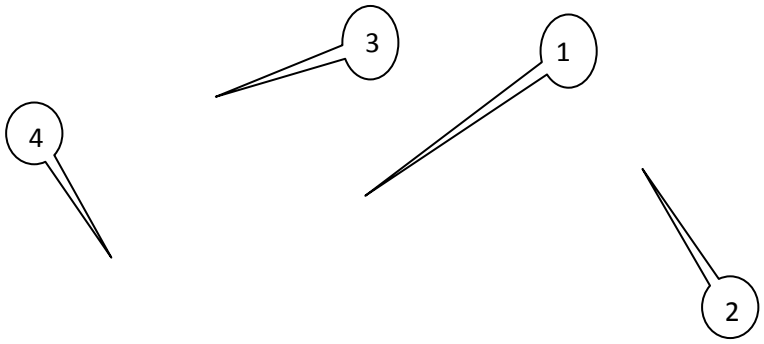



Figure III-2: plateforme Xtreme DSP Starter Kit.

1. Le Circuit FPGA XC3SD1800A-4FGG676C
2. Connecteur d'alimentation de la carte
3. Port de programmation/configuration
4. Port série RS-232

### III.3. Outil Software utilisé :

Nous avons utilisé l'environnement de développement EDK de Xilinx englobant les deux outils XPS et SDK (voir chapitre II). Avec le Base System Builder, un outil d'assistance de XPS, on pourra créer rapidement un nouveau projet. XPS utilise les outils fournis avec ISE pour synthétiser, placer et router la logique dans le matériel. Des outils GNU sont fournis avec l'EDK et utilisés à l'intérieur de XPS pour concevoir le logiciel pour le système embarqué Microblaze.

Les modules du système Microblaze, utilisés dans cette partie sont les suivantes :

- Microblaze
- Contrôleur LMB pour BRAM
- BRAM
- BUS PLB
- UART PLB
- MDM PLB

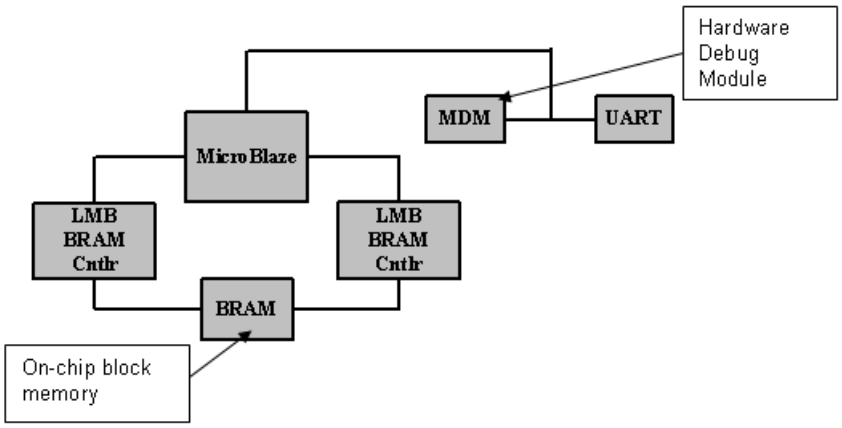



Figure III-3 : Modules du système Microblaze utilisés.

### III.4. Réalisation du projet sous EDK :

#### III.4.1. Utilisation du Base System Builder de XPS

On commence par l'ouverture de l'outil Xilinx Platform Studio (XPS). On crée un nouveau projet en utilisant « Base System Builder » (Figure III.4). On obtient la figure III.5.

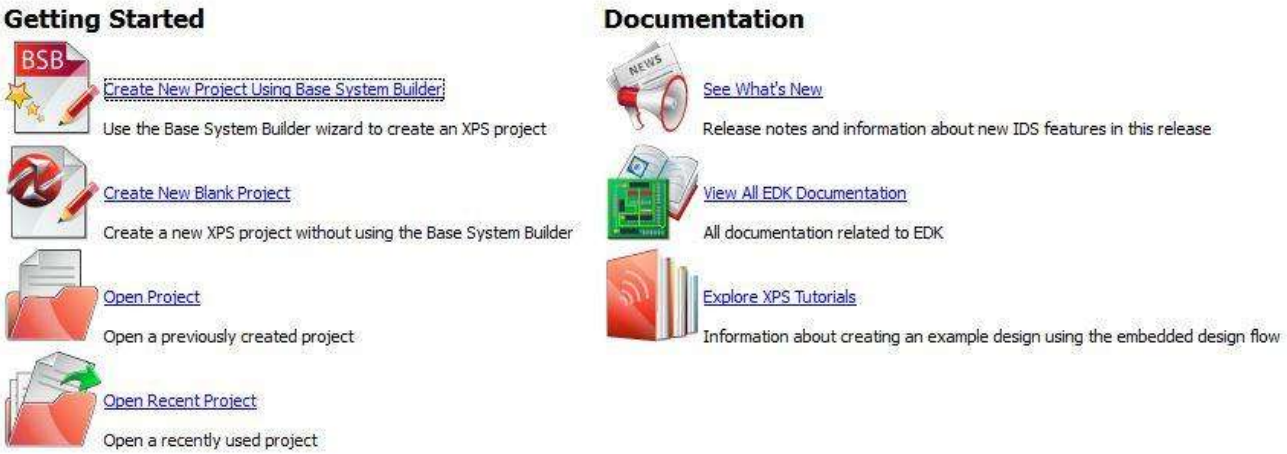



Figure III-4: Démarrage application XPS.

On indique au logiciel le chemin du répertoire du projet (projet_aes128 créer auparavant) et on sélectionne le bus d'interconnexion de type PLB.

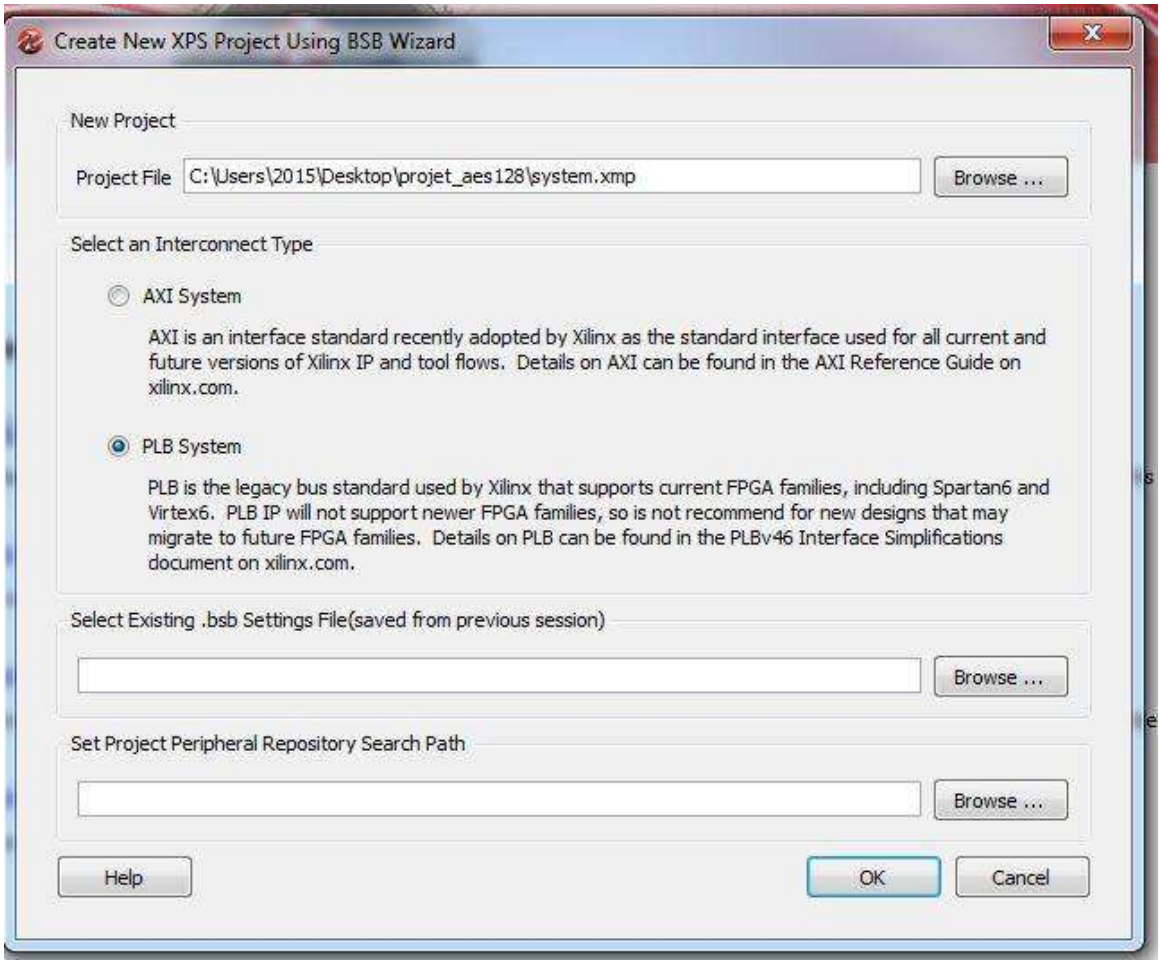



Figure III-.5: création nouveau projet XPS.

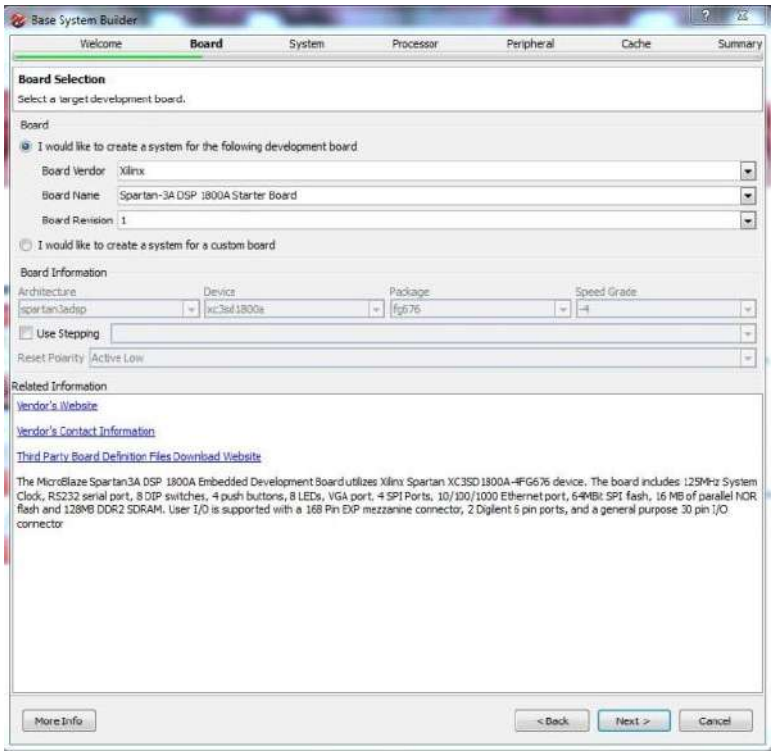



Figure III-6 : Choix carte FPGA.

Ensuite, on aura la figure III.6, où on choisit la carte de développement (Spartan 3A DSP 1800 starter board) et on valide ce choix. On obtient la figure III.7.

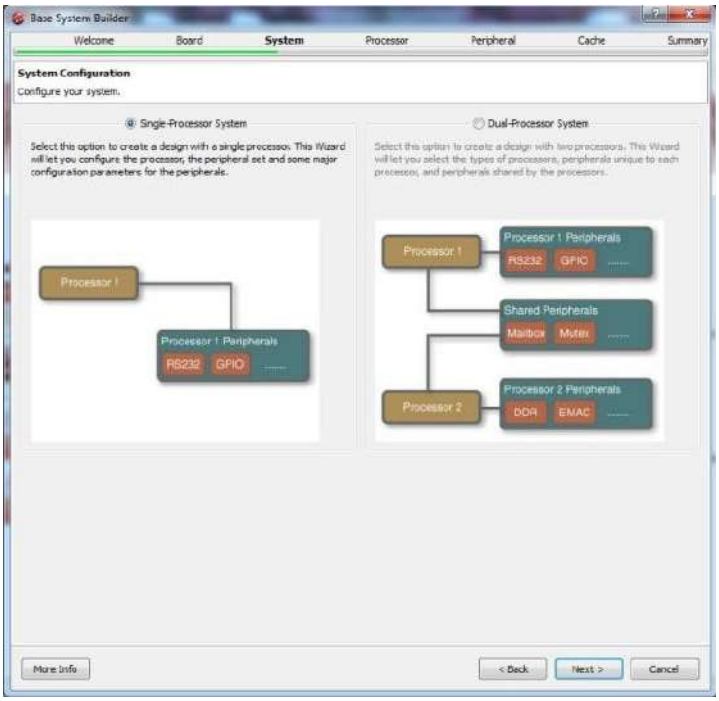



Figure III-7 : Sélection solution mono processeur.

On choisit une architecture mono processeur. Après validation on passe à la figure III.8.

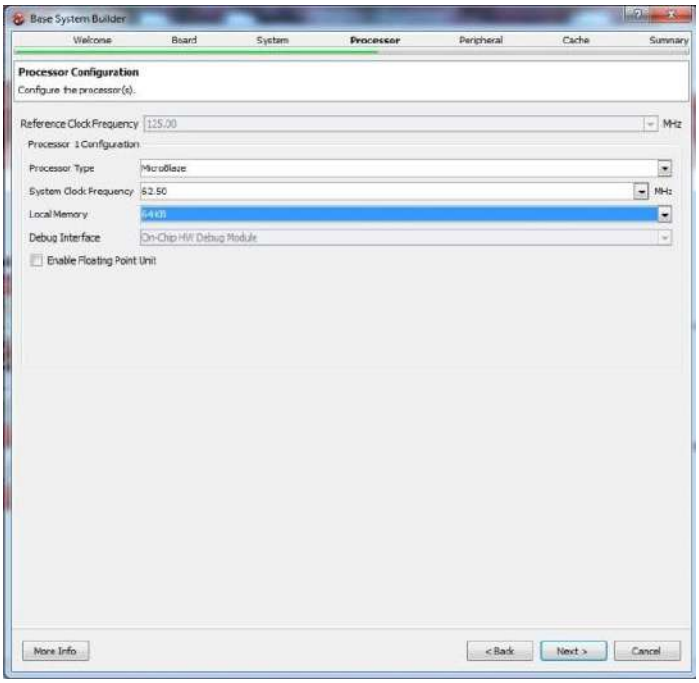



Figure III-8 : sélection taille mémoire et fréquence d'horloge microblaze.

On fixe la taille mémoire interne à 64 koctets avec une fréquence d'horloge du processeur à 62,5 MHz. Ensuite, on supprime les modules non utilisés comme le montre la figure III.9.

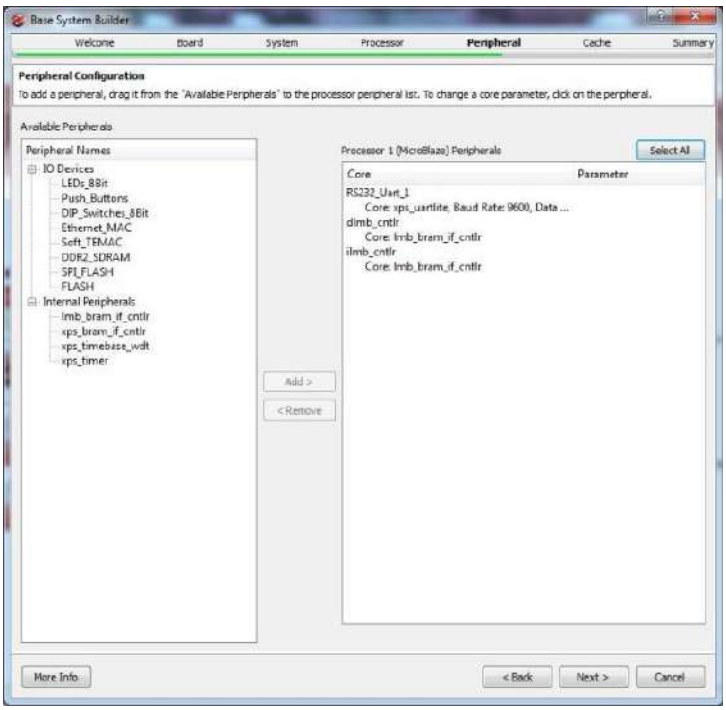



Figure III-9 : sélection modules utiles.

Après validation, le navigateur des projets de Xilinx Platform Studio charge notre projet comme il représenté sur la figure III.10. À ce moment, XPS a généré un fichier de contraintes UCF (figure III.14), un fichier projet (system.xmp), un fichier de spécification de matériel (system.mhs : figure III.13) et un fichier de configuration logiciel (system.mss) dans votre répertoire.

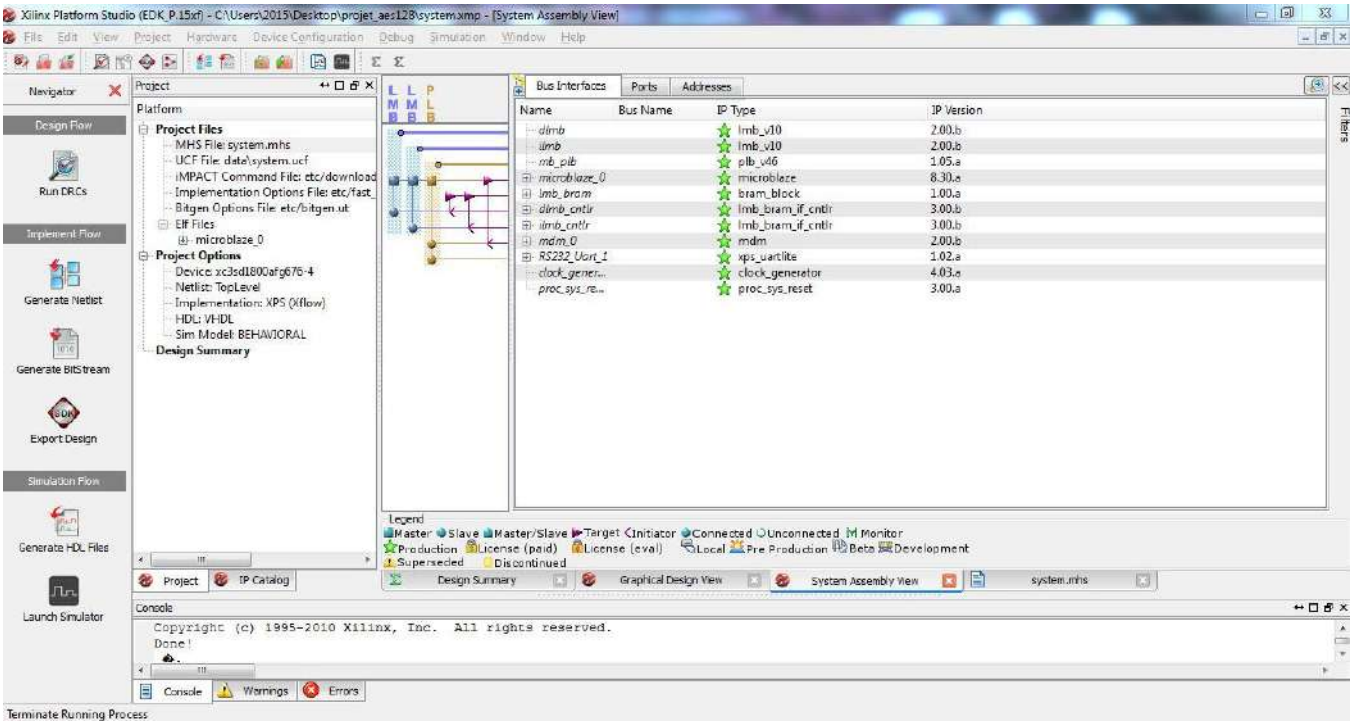



Figure III-10 **:** Le navigateur des projets de Xilinx Platform Studio.

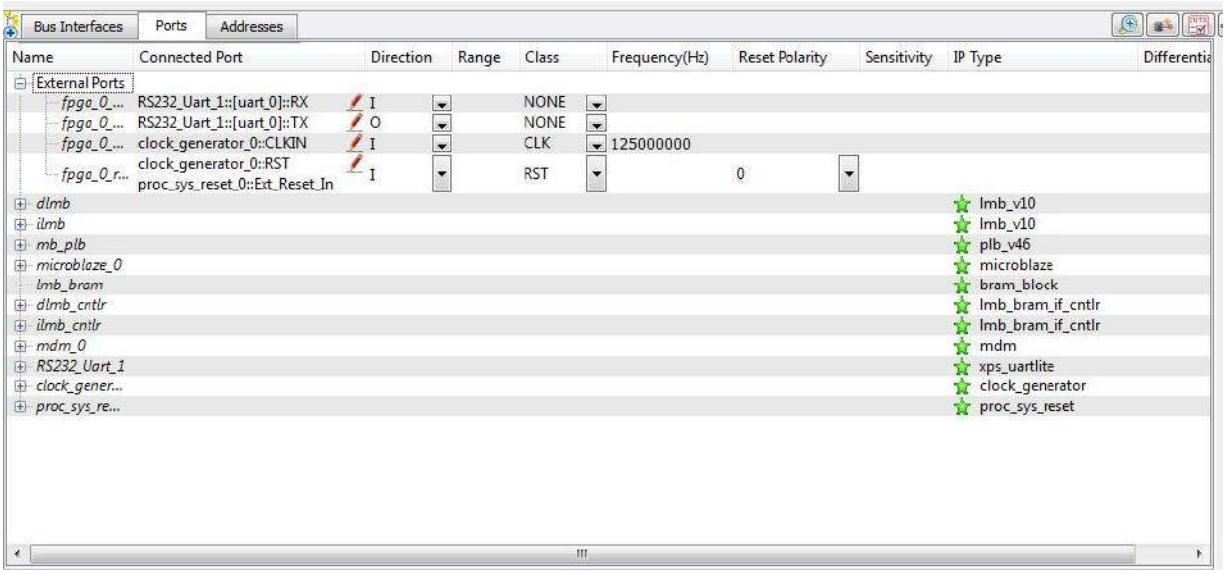
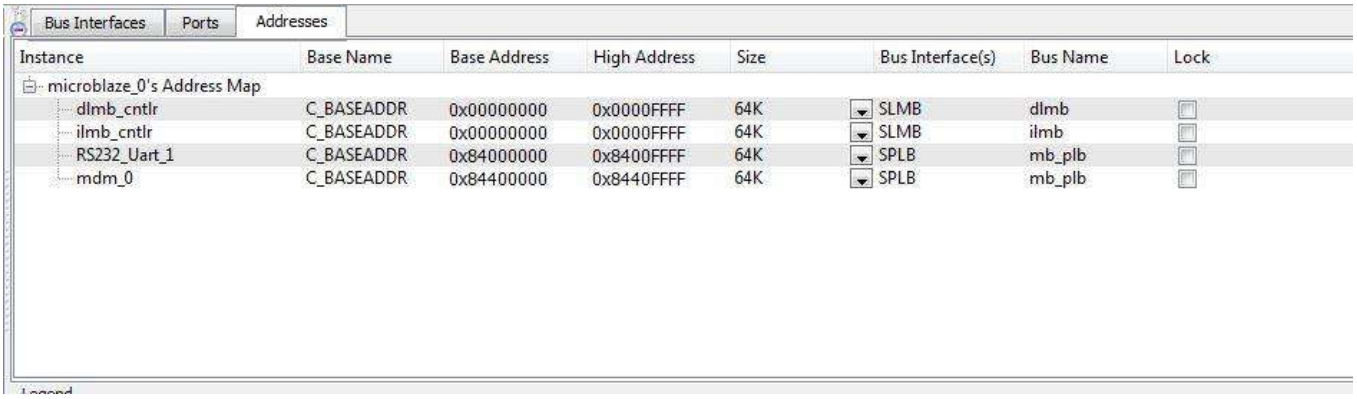



Figure III-11 : Capture d'écran des ports utilisés.



Bus Interfaces | Ports | Addresses

| Instance | Base Name | Base Address | High Address | Size | Bus Interface(s) | Bus Name | Lock |
|---|---|---|---|---|---|---|---|
| microblaze_0's Address Map | | | | | | | |
| dlmb_cntlr | C_BASEADDR | 0x00000000 | 0x0000FFFF | 64K | SLMB | dlmb | |
| ilmb_cntlr | C_BASEADDR | 0x00000000 | 0x0000FFFF | 64K | SLMB | ilmb | |
| RS232_Uart_1 | C_BASEADDR | 0x84000000 | 0x8400FFFF | 64K | SPLB | mb_plb | |
| mdm_0 | C_BASEADDR | 0x84400000 | 0x8440FFFF | 64K | SPLB | mb_plb | |

Figure III-12 : Capture d'écran des adresses réservées aux différents modules.

```
1
2  # ##############################################################################
3  # Created by Base System Builder Wizard for Xilinx EDK 14.1 Build EDK_P.15xf
4  # Thu Jun 04 12:08:11 2015
5  # Target Board:  Xilinx Spartan-3A DSP 1800A Starter Board Rev 1
6  # Family:    spartan3adsp
7  # Device:    xc3sd1800a
8  # Package:   fg676
9  # Speed Grade:  -4
10 # Processor number: 1
11 # Processor 1: microblaze_0
12 # System clock frequency: 62.5
13 # Debug Interface: On-Chip HW Debug Module
14 # ##############################################################################
15  PARAMETER VERSION = 2.1.0
16
17
18  PORT fpga_0_RS232_Uart_1_RX_pin = fpga_0_RS232_Uart_1_RX_pin, DIR = I
19  PORT fpga_0_RS232_Uart_1_TX_pin = fpga_0_RS232_Uart_1_TX_pin, DIR = O
20  PORT fpga_0_clk_1_sys_clk_pin = CLK_S, DIR = I, SIGIS = CLK, CLK_FREQ = 125000000
21  PORT fpga_0_rst_1_sys_rst_pin = sys_rst_s, DIR = I, SIGIS = RST, RST_POLARITY = 0
22
23
24 BEGIN microblaze
25  PARAMETER INSTANCE = microblaze_0
26  PARAMETER C_AREA_OPTIMIZED = 1
27  PARAMETER C_USE_BARREL = 1
28  PARAMETER C_DEBUG_ENABLED = 1
29  PARAMETER HW_VER = 8.30.a
30  BUS_INTERFACE DLMB = dlmb
```

Figure III-13 **:** fichier de spécification de matériel MHS.

```
1  #  Spartan-3A DSP 1800A Starter Board
2  Net fpga_0_RS232_Uart_1_RX_pin LOC = N21  |  IOSTANDARD=LVTTL;
3  Net fpga_0_RS232_Uart_1_TX_pin LOC = P22  |  IOSTANDARD=LVTTL;
4  Net fpga_0_clk_1_sys_clk_pin TNM_NET = sys_clk_pin;
5  TIMESPEC TS_sys_clk_pin = PERIOD sys_clk_pin 125000 kHz;
6  Net fpga_0_clk_1_sys_clk_pin LOC = F13  |  IOSTANDARD=LVCMOS33;
7  Net fpga_0_rst_1_sys_rst_pin TIG;
8  Net fpga_0_rst_1_sys_rst_pin LOC = Y16  |  IOSTANDARD=LVTTL  |  PULLUP;
9
```

Figure III14 **:** fichier de contraintes UCF.

### III.4.2. Implémentation du matériel

On sélectionne l'icône Generate Bitstream pour générer le fichier system.bit contenant l'architecture matérielle de notre système. Cette opération peut prendre une dizaine de minutes.

**Résultats de l'implémentation**

La figure III.15 représente le schéma RTL du système à base du microblaze implémenté. Le rapport des résultats concernant l'occupation de surface est comme le montre la table III.1.

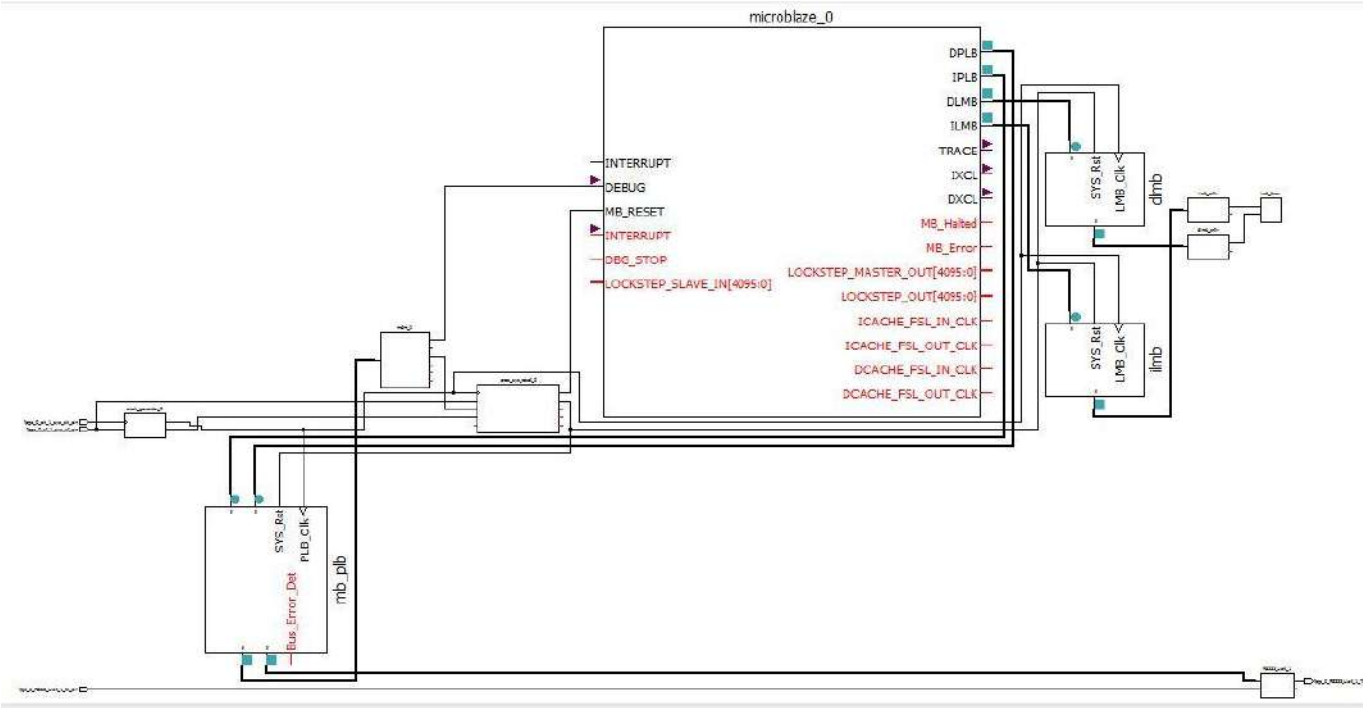



Figure III-15 **:** Schéma RTL du système à base du MicroBlaze implémenté.

Table III-1 **:** Rapport des résultats concernant l'occupation de surface

| **Device Utilization Summary (actual values)** | | | | **[-]** |
|---|---|---|---|---|
| **Logic Utilization** | **Used** | **Available** | **Utilization** | **Note(s)** |
| Number of Slice Flip Flops | 1,011 | 33,280 | 3% | |
| Number of 4 input LUTs | 2,136 | 33,280 | 6% | |
| Number of occupied Slices | 1,555 | 16,640 | 9% | |
| Number of Slices containing only related logic | 1,555 | 1,555 | 100% | |
| Number of Slices containing unrelated logic | 0 | 1,555 | 0% | |
| Total Number of 4 input LUTs | 2,191 | 33,280 | 6% | |
| Number used as logic | 1,752 | | | |
| Number used as a route-thru | 55 | | | |
| Number used for Dual Port RAMs | 256 | | | |
| Number used as Shift registers | 128 | | | |
| Number of bonded IOBs | 4 | 519 | 1% | |
| IOB Flip Flops | 2 | | | |
| Number of BUFGMUXs | 3 | 24 | 12% | |
| Number of DCMs | 1 | 8 | 12% | |
| Number of BSCANs | 1 | 1 | 100% | |
| Number of DSP48As | 3 | 84 | 3% | |
| Number of RAMB16BWERs | 32 | 84 | 38% | |
| Number of BSCAN_SPARTAN3As | 1 | 1 | 100% | |
| Average Fanout of Non-Clock Nets | 4.01 | | | |

### III.4.3. Utilisation de l'outil SDK

Après génération du fichier bitstream de l'architecture matérielle, c'est-à-dire, que notre système à base du microblaze est déjà implémenté. On passe à la conception logicielle de l'algorithme de cryptage / décryptage AES128 en langage C.

On clique sur l'icône Export Design, l'environnement SDK démarre. On crée un nouveau projet (test_aes128), avec les fichiers développés : aes.c, aes.h et test-aes128.c. Les autres fichiers : platform.c, platform.h et platform_config.h sont des fichiers chargés automatiquement par l'outil SDK.

La figure III.16 montre l'environnement SDK avec notre projet développé.

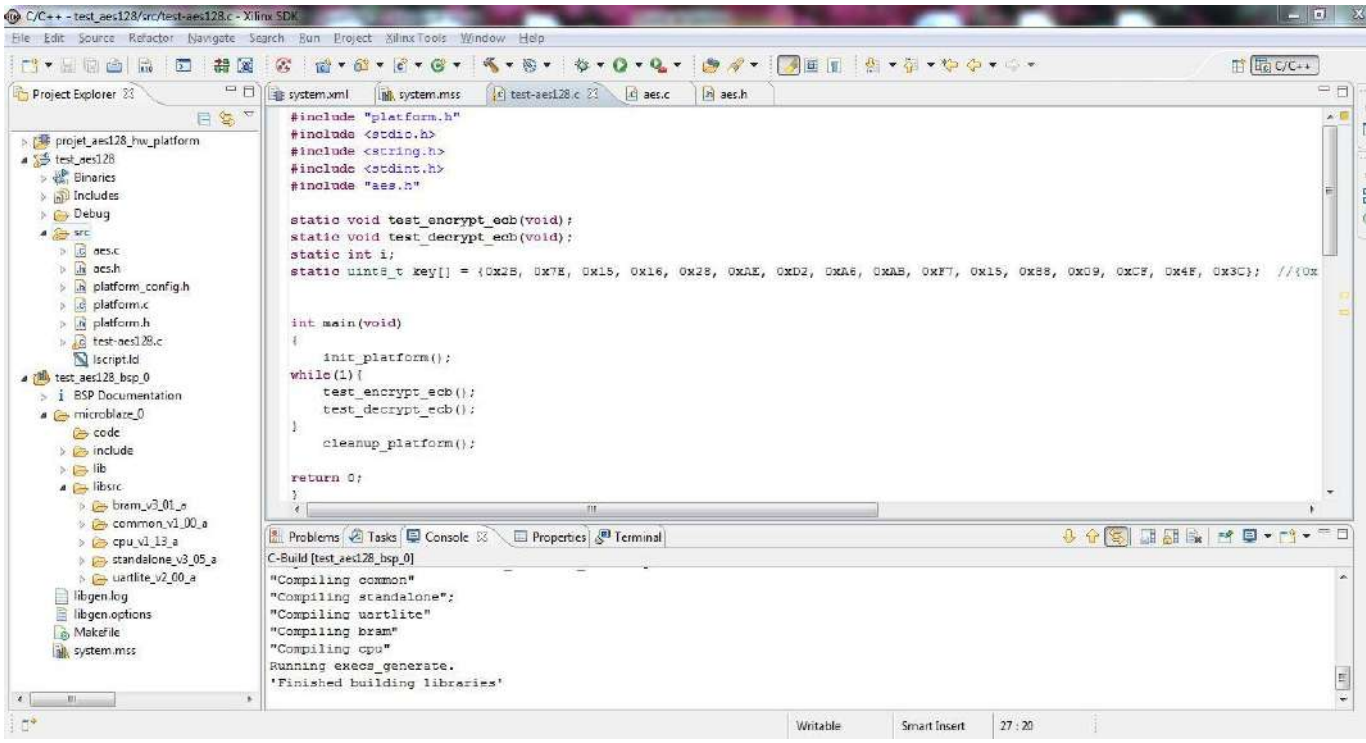



Figure III-16 **:** Le navigateur des projets de Software Development Kit.

### III.5. Organigrammes des fonctions de test de cryptage/décryptage implémentée

Les figures III.17 et III.18 montrent respectivement les organigrammes des fonctions de test des algorithmes de cryptage / décryptage AES 128 bits implémentées sur l'architecture matérielle à base du microblaze.

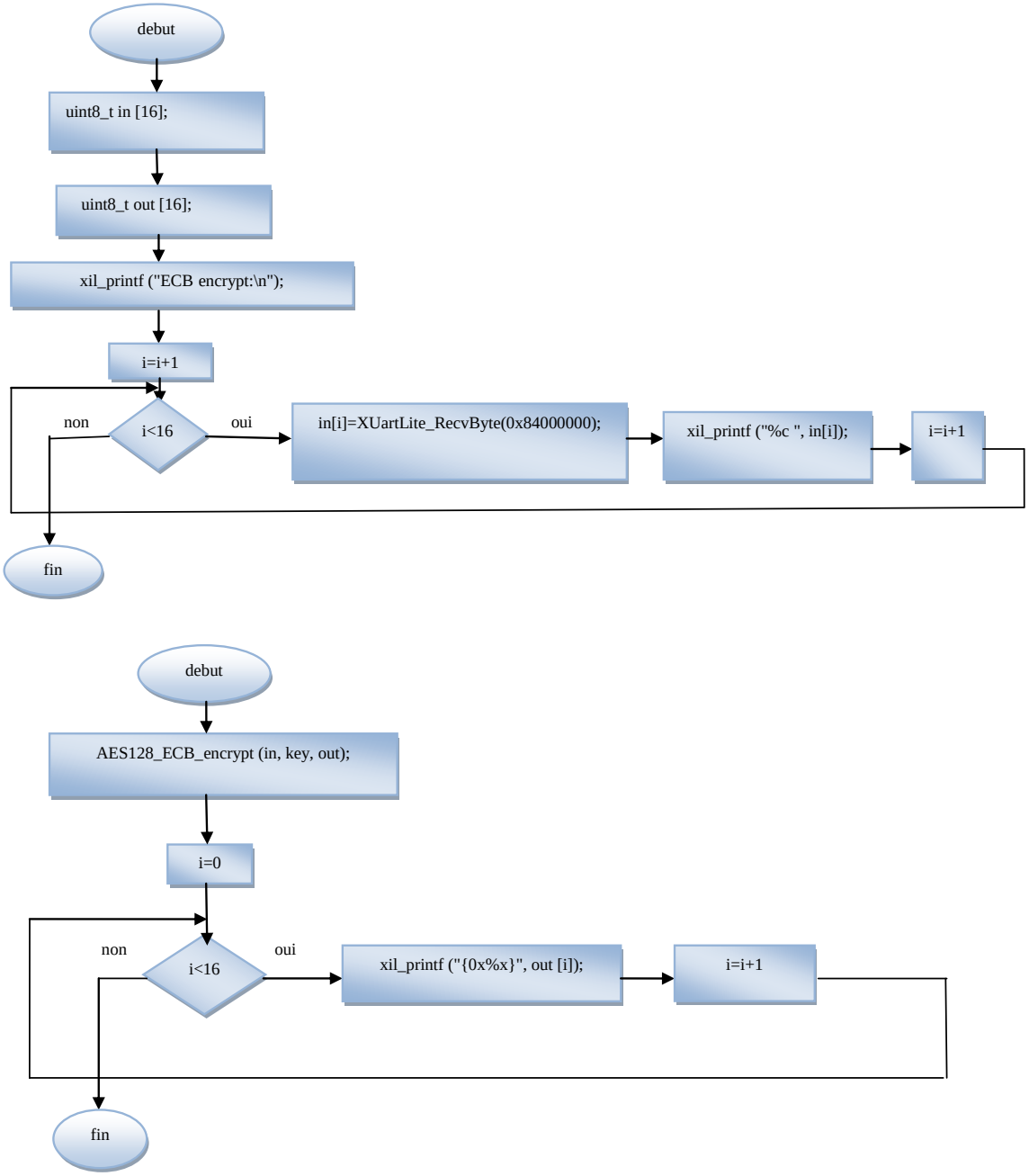



Figure III-17 : Organigramme de la fonction de test de l'algorithme de cryptage AES128.

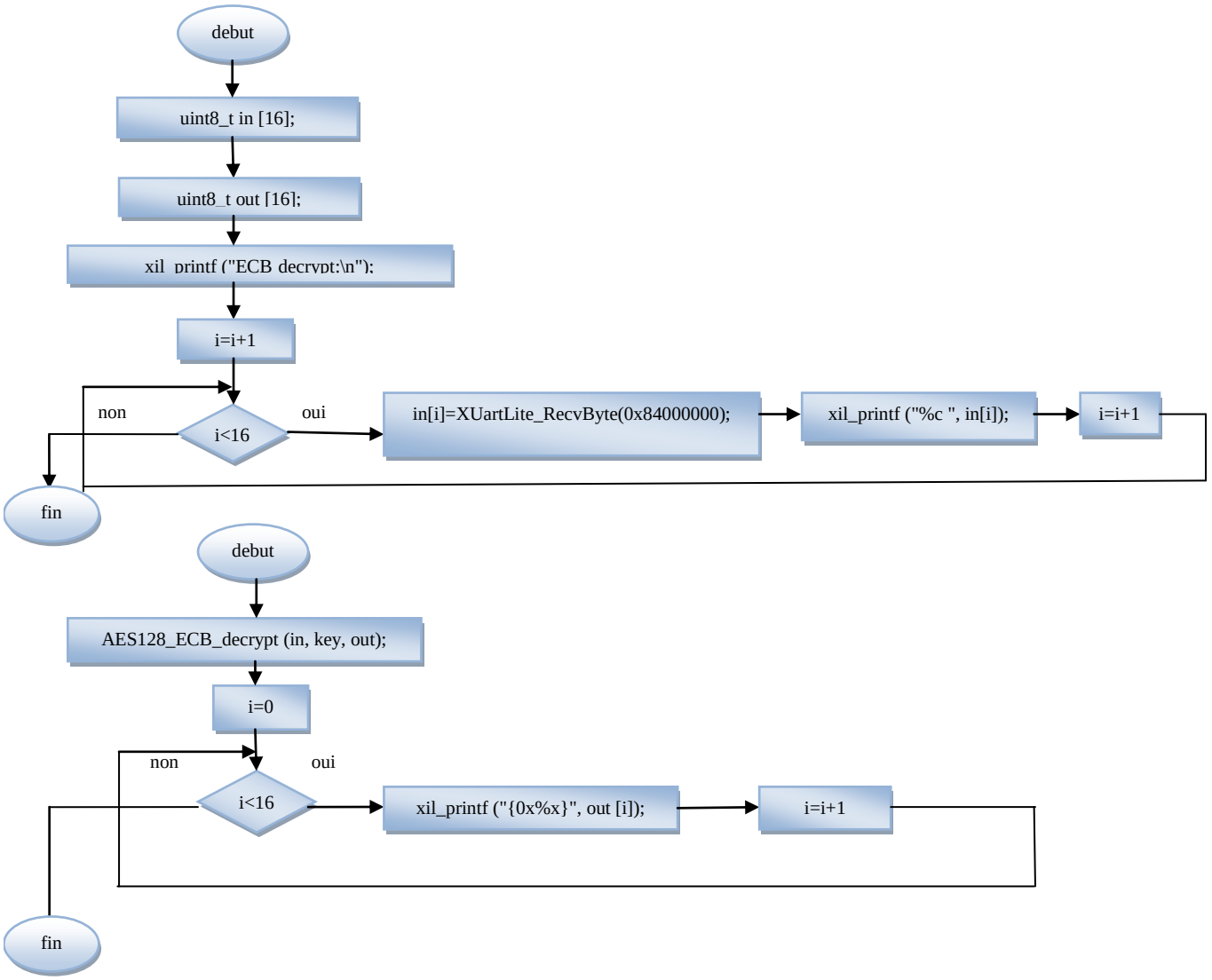



Figure III-18 : Organigramme de la fonction de test de l'algorithme de décryptage AES128.

### III.6. Résultats du test pratique sur la carte FPGA :

Pour faire le test pratique du système développé sur la carte FPGA, on a utilisé les outils suivants :

- La carte FPGA.
- Un ordinateur portable.
- Un câble adaptateur liaison série RS232 / USB.
- Un hyper Terminal HERCULES 3.2.5 pour l'envoi et la réception des données en hexadécimal via le port USB émulé port série RS232.

Afin d'assurer la synchronisation entre l'ordinateur et la carte, on a utilisé la configuration suivante : le débit de transfert = 9600 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 (même exemple du document officiel Fips-197 **[8]**) :

0x 3243F6A8885A308D313198A2E0370734

Le message crypté reçu : 0x 3925841D02DC09FBDC118597196A0B32

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 : 0x 2B7E151628AED2A6ABF7158809CF4F3C

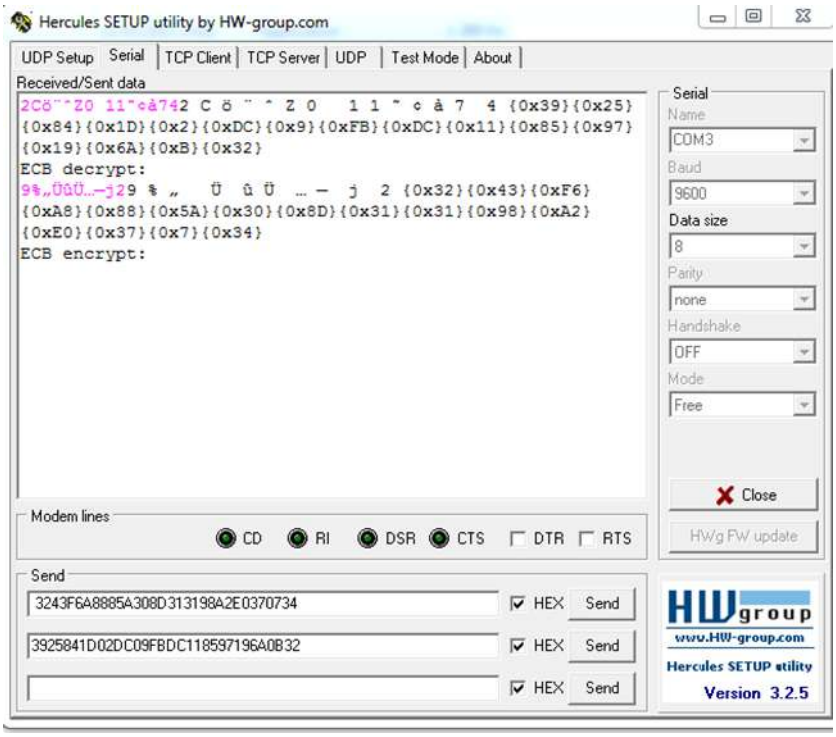



Figure III-19 : Résultat du test pratique sur la carte.

## III.7. Conclusion :

Notre conception pour le chiffrage/déchiffrage avec l'algorithme AES 128 bits, a été 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.

Nous suggérons au futur, de faire une comparaison entre notre implémentation logicielle sur un softcore, avec une implémentation matérielle (développée en langage VHDL) d'un PFE antérieur **[10]**.

# Conclusion générale

# Conclusion générale

Dans ce projet, nous avons développé une conception d'architecture de l'algorithme AES de cryptage et de décryptage dans le circuit FPGA, l'implémentation harde de l'algorithme AES est une solution qui permet la validation de façon matérielle de nouveaux principes de sécurité.

Le circuit développé est pris par l'outil de Xilinx et enfin fait l'exécution en utilisant le circuit FPGA SPARTAN 3A DSP XC3SD1800A.

Notre conception peut chiffrer et déchiffrer des données au format hexadécimal , 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 sans fil actuels, ou d'autres processus de transfert de données électroniques dont la sécurité est nécessaire.

Finalement, l'approche traitée au cours de ce travail peut-être développé et utiliser au futur comme partie essentielle dans l'implémentation Hardware Software sur les circuits programmables.

# Bibliographie

# Bibliographie

**[1]** RAMDANI mohamed, problèmes de sécurités dans les réseaux de capteurs avec prise en charge de l'énergie, mémoire de magister.

**[2]** Renaud Dumont. Cryptographie et Sécurité informatique Notes de cours provisoires 2009 – 2010.

**[3]** Federal Information, Processing Standards Publication 197, November 26, 2001

Announcing the ADVANCED ENCRYPTION STANDARD (AES)

**[4]** Kamel MESSAOUDI, Traitement des signaux et images en temps réel: implantation de H.264 sur MPSoC, Thèse de Doctorat, 2012.

**[5]** « Rod Jesman Fernando Martinez Vallina Jafar Saniie ». MicroBlaze Tutorial Creating a Simple Embedded System and Adding Custom Peripherals Using Xilinx EDK Software Tools.

**[6]** Site Internet, http://www.opencores.com.

**[7]** « Mr. AWAL DADE Sahabi ». Implémentation de l'interface SPI sur une carte FPGA « Juin 2014 ».

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

**[9]** Getting Started with the Spartan-3A DSP S3D1800A Starter Platform User Guide", UG485, version 1.1, June 12, 2008,p 7-8. www.xilinx.com.

**[10]** KHIDER Ali et YAALA Mohamed, Implémentation de l'Algorithme de Cryptage AES sur un circuit FPGA, 2012-2013.