N° d'ordre :

REPUBLIQUE ALGERIENNE DEMOCRATIQUE & POPULAIRE

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR & DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE DJILLALI LIABES
FACULTE DES SCIENCES EXACTES
SIDI BEL ABBES

THESE
DE DOCTORAT EN SCIENCES

Présentée par : FILALI Mohamed Amine

Spécialité : Informatique
Option : Informatique

Etude et implémentation et intégration De L'Algorithme AES-IP CORE Dans Les Architectures Applicatives

Soutenue le 26/06/2022 .
Devant le jury composé de :

Président :	FARAOUN Kamel Mohamed	Professeur UDL SBA
Examinateurs :	RAHMOUN Abdellatif	Professeur ESB SBA .
	KESKAS Nabil	Professeur ESB SBA
Directeur de thèse :	GAFOUR Abdelkader	Professeur UDL SBA.

Année universitaire : 2021-2022

Je dédie ce modeste travail à :

Mes chers parents, pour tous leurs sacrifices, leur soutien tout au long de mes études, grâce à eux je n'ai manqué de rien,

Ma petite famille ma femme et mon fils « Wassim » pour leur encouragement permanent, et leur soutien moral,

Dr. Belkhaira qui m'a soutenu tout au long de ma thèse de doctorat,

Que ce travail soit l'accomplissement de vos vœux, et le fruit

de votre soutien infaillible,

« MERCI » d'être toujours là pour moi.

Remerciement :

Au nom de dieu saint clément et miséricordieux

Nous tenons avant tout à remercier notre dieu pour sa bienveillance et sa bénédiction.

La première personne que je souhaite remercier est mon directeur de thèse Monsieur Pr GAFOUR Abdelkader. Il m'a constamment soutenu, encouragé et stimulé durant des années de thèse. Ses nombreuses remarques ont montré une très vaste connaissance du sujet. En dehors du plan technique, nous avons eu un bon rapport d'amitié et de respect mutuel.

Ensuite j'adresse mes remerciements à tous les membres de mon jury d'avoir accepté d'évaluer mon travail. J'apprécie l'intérêt qu'ils ont porté à mes travaux en cryptographie.

Je remercie également l'ensemble des enseignants et les membres du personnel non enseignant du département d'informatique de la faculté des sciences exactes de l'Université de Sidi Bel-Abbès

Mes remerciements ne peuvent s'achever, sans une pensée pour mon ami Dr. Belkhaira qui m'a soutenu tout au long de ma thèse de doctorat,

«Few persons can be made to believe that it is not quite an easy thing to invent a method of secret writing which shall baffle investigation. Yet it may be roundly asserted that human ingenuity cannot concoct a cipher which human ingenuity cannot resolve...»

Edgar Allan Poe *A Few Words on Secret Writing, 1841.*

Table des matières

Chapitre V ***Implémentation et intégration AES IP***

Liste des figures

Chapitre II

Chapitre III

Chapitre IV

Liste des figures

Chapitre V

Liste des figures

LISTE DES ACRONYMES ET ABREVIATIONS

AES	Advanced Encryption Standard
ASIC	Application Specie Integrated Circuit
BRAM	Block Random Access Memory
CBC	Cipher Block Chaining
CFB	Cipher Feed Back
CLB	Configurable logic block
CPLD	Complex PLD
CTR	Compteur
DES	Data Encryption Standard
DLL	Delay Locked Loop
DSA	Digital Signature Algorithm
ECB	Electronic Code Book
FDC	D-type flip-flop active-high Clear
FF	Flip Flop
FIPS	Federal Information Processing Standard
FPGA	Field Programmable Gate Array
IDEA	International Data Encryption Algorithm
IP	intellectual property
LUT	Look Up Table
RTL	Register Transfer Level
NIST	National Institute of Standards and Technology
OFB	Output Feed Back
PLD	Programmable Array Device
RSA	Rivest Shamir Adelman cryptosystem
SBox	Substitution Box
VHDL	Very heights peed integrated circuit Hardware Description Language
XOR	Exclusive
ORGF(2^p)	Champ de Galois

Introduction générale

L'information est un élément constitutif et déterminant dans tous les domaines. Tout au long de l'histoire, l'humanité a essayé d'envoyer des informations d'une façon sécurisée. La dissimulation d'information a été utilisée comme instrument de sécurisation pour les stratégies militaires et échange de données secrètes [1].

La nécessité de transfert sécurisé d'information a traversé le temps et est encore énormément utilisée dans le monde numérique. Il y a une variété importante de domaines d'applications des méthodes de dissimulation d'information. Ces méthodes disposent de différentes caractéristiques et peuvent être classifiées selon leurs objectifs et leurs contraintes.

D'une façon générale, les méthodes de dissimulation d'information peuvent être groupées en deux grandes familles:

- La première famille utilise des techniques pour rendre incompréhensible le message :la cryptographie qui se sert des mathématiques pour rendre le message indéchiffrable, et le brouillage où mélange (scrambling) qui utilise des techniques pour désordonner le message et le rendre illisible.
- La deuxième famille utilise une porteuse comme enveloppe pour cacher le message.

La cryptographie, tout comme le savoir, a commencé avec l'écriture. Elle concerne l'humanité depuis longtemps, elle reste une technique et l'on pourrait la restreindre à une certaine caste. En effet, si l'on s'attarde à faire un parallèle, par exemple, avec l'automobile, beaucoup de personnes emploient aujourd'hui une voiture, sans avoir la moindre idée du fonctionnement interne du moteur, de ce qu'il y a sous le capot. Il en va de même pour le chiffrement et les algorithmes associés. Dans ce mémoire nous sommes donc convié à une introduction au monde de la cryptographie à travers une frise historique relatant les grands moments de sa vie.

Introduction générale

Problématique :

Les réseaux numériques ont tellement évolué qu'ils sont devenus un mécanisme essentiel de communication. Ils permettent de transmettre toute sorte d'informations. Ainsi La croissance exponentielle de matériel (processeur, Ram… etc.).

La transmission des données soulève donc un nombre conséquent de problèmes qui ne sont pas tous encore résolus. De plus, les réseaux informatiques sont complexes et les écoutes illégales sont nombreuses.

Afin de comprendre l'approche et le but de notre travail de recherche, nous avons réuni les problèmes concernant la sécurité et l'authenticité de la donnée transmise sur des réseaux numériques dont notamment :

- Un premier problème est relatif à l'aspect sécurité et à l'authenticité des données pendant la transmission, mais également après réception de celles-ci.

Toute information circulant peut être capturée, lue et ou modifiée.

- Un deuxième problème concerne le temps de transfert. En effet, du fait de la quantité importante de données, comme les tailles des images doivent être réglables avant le transfert.
- Pour des raisons de confidentialité, certaines données doivent être rendues complètement ou partiellement illisibles et non déchiffrables pendant le transfert.
- Evaluation du matériel hardware comme la vitesse des processeurs qui est devenue un paramètre très important pour les attaques sur les différents systèmes Cryptographiques.

Contribution

Plusieurs méthodes ont été développées pour résoudre les problèmes de sécurité et d'authenticité dans le transfert des données. Dans le domaine de cryptographie on trouve plusieurs algorithmes comme DES, RSA, AES ….

Des études ont démontré que l'algorithme AES est plus sécurisé que les autres algorithmes de chiffrements.

Dans ce travail nous allons présenter une architecture d'algorithme AES IP-CORE qui permet d'avoir des temps de traitement relativement court.

Plan de la thèse

Cette thèse est organisée de la façon suivante:

Le premier chapitre présente une introduction sur la cryptologie où nous définissons ses mots clés les plus utilisés et nous exposons ses concepts. Le chapitre 2 décrit les déférents systèmes cryptographiques,

Dans le chapitre 3 nous présentons une description détaillée de l'algorithme d'AES. En chapitre 4 nous présentons la technologie FPGA

Nous décrivons dans le chapitre 5 l'architecture d'Algorithme AES core et son implantation sur circuit XILINX. Nous explicitons également dans ce chapitre les résultats et discussions. Nous terminons par une conclusion générale.

Chapitre I

Sommaire:

1. Introduction:

La cryptographie constitue un ensemble de techniques permettant de garantir la confidentialité et l'intégrité des échanges d'information. Il s'agit d'une science ancienne, puisqu'il y a près de deux mille ans que Jules César l'utilisait déjà. [1][2] Jusqu'à récemment, elle est restée confinée aux domaines militaires et diplomatiques. Toutefois le développement des télécommunications et des échanges d'information sous forme électronique au cours des vingt-cinq dernières années a permis d'émanciper(libérer)la cryptographie de son statut de science «top secrète».

1.1. Introduction à la cryptographie

La cryptographie est l'art de crypter l'information pour la rendre inaccessible aux personnes non concernées. Il ne faut pas la confondre avec la cryptanalyse, qui est juste le contraire : c'est l'art de décrypter, ou de casser les protections. Ces deux branches forment la cryptologie.
En gros, on a :

Cryptologie=Cryptographie+Cryptanalyse.

Rappelons que le chiffrement est une branche de la cryptographie.

Texte en clair **Texte en clair original**

M **chiffre** son message pour le rendre illisible sauf pour *C* qui doit d'abord le **déchiffrer** pour pouvoir le lire.
Si le message de départ est noté M,

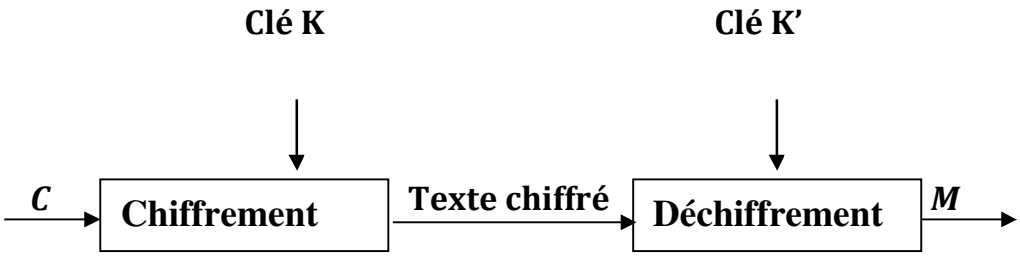
Le procédé de chiffrement est noté E,

Le procédé de déchiffrement est noté D,

Le message chiffré est noté C, alors on peut écrire:

E (M)=C ; D(C)=M ; ou encore **D(E(M))=M ;**

1.2. La notion de clé:



Pour une vraie sécurité, tous les algorithmes modernes de chiffrement utilisent une **clef**, notée k. Cette clé peut prendre une des valeurs parmi un grand nombre de valeurs possibles.

La valeur de la clé affecte les algorithmes de chiffrement et de déchiffrement et donc les fonctions de chiffrement et de déchiffrement en dépendent, ce que l'on note:

$$En(M)=C ;$$

$$Dn'(C)=M ;$$

$$Dn'(En(M))=M;$$

1.3. État de l'art sur la cryptographie

Voici donc un rapide descriptif des faits marquants et des précurseurs qui ont permis l'apparition et l'évolution de la cryptographie

* Vers 1900 av. J.-C., un scribe égyptien a employé des hiéroglyphes non conformes à la langue correcte dans une inscription.

* Quatre siècles plus tard, vers 1500 av. J.-C., une tablette mésopotamienne contient une formule chiffrée pour la fabrication de vernis pour les poteries.

* Cinq siècles avant notre ère, des scribes hébreux mettant par écrit le livre de Jérémie ont employé un simple chiffre de substitution connu sous le nom d'Atbash.

* En 487 av. J.-C., les grecs emploient un dispositif appelé la scytale, un bâton autour duquel une bande longue et mince de cuir était enveloppée et sur laquelle on écrivait le message. Le cuir était ensuite porté comme une ceinture par le messager [17]. Le destinataire avait un bâton identique permettant d'enrouler le cuir afin de déchiffrer le message.

* Au 9 ème siècle, Abu Yusuf Ya'qub ibn Is-haq ibn as-Sabbah Oomran ibn Ismail al-Kindi rédige le plus ancien texte connu décrivant la technique de décryptement appelée analyse des fréquences.

* A partir de 1226 une timide cryptographie politique apparaît dans les archives de Venise, où des points ou des croix remplacent les voyelles dans quelques mots éparses (répandus) [16].

Voici quelque repère de la cryptographie moderne :

- 1975 conception du standard de chiffrement de données adopté en1977
- 1976 Diffie et Hellman introduisent l'idée de système à clé publique
- 1978 invention de RSA le premier système concret de cryptographie à clé publique.
- 1985 invention du système cryptographie El Gamal
- 1991 adoption du premier standard de signature basée sur l'algorithme discret.
- 2000 adoption du Rijndeal comme AES(successeur du DES).[6]

2. Fonctionnement:

2.1. **Services de Sécurité (la fonction de cryptographie):**

• **La confidentialité** : est une protection de l'information d'une divulgation (révélation) non autorisée.

• **L'intégrité**: correspond à une protection contre la modification non autorisée de l'information.

• **La disponibilité**: c'est de s'assurer que les ressources sont accessibles aux utilisateurs légitimes

• **L'authentification**:

• **L'authentification d'entités** : (*entity authentication*) c'est un procédé permettant à une entité d'être sûre de l'identité d'une seconde entité à l'appui d'une évidence corroborante (ex: présence physique, cryptographique, biométrique, etc.).

Le terme *identification* est parfois utilisé pour désigner également ce service.

- **L'authentification de l'origine de données**: (*data origin authentication*) c'est un procédé permettant à une entité d'être sûre qu'une deuxième entité est la source originale d'un ensemble de données. Par définition, ce service assure également l'intégrité de ces données.

• **La Non-répudiation** : elle Offre la garantie qu'une entité ne pourra pas nier être impliquée dans une transaction (accord).

• **La Non-Duplication** :est la protection contre les copies illicites(illégales).

• **L'anonymat** (d'entité ou d'origine de données): elle permet de préserver l'identité d'une entité, de la source d'une information ou d'une transaction.[7], [8].

2.2. Définitions:

La cryptologie, étymologiquement la *science du secret, vient* du grec «Cryptos»=«caché»:

- Cacher les données, les rendre illisibles à celui qui n'a pas la «clef» pour les lire.
- Cela va plus loin que simplement caché les données:

-signatures électroniques=authentification.

L'algorithme cryptographique est une fonction mathématique utilisée pour le chiffrement et le déchiffrement.

La fonction (ou transformation) est définie par deux ensembles X et Y et d'une règle f qui assigne à chaque élément de X précisément un élément de Y.

Nous notons cela: $f:X \rightarrow Y$

Le chiffrement est le processus consistant à appliquer une transformation E paramétrée par k et notée Ek à un message m appartenant à M est appelé chiffrement de m.

Le déchiffrement est le processus consistant à appliquer une transformation D paramétrées par k0 et notée Dk0 à un message chiffré C appartenant à C est appelé déchiffrement de C.

La cryptanalyse étudie la sécurité des procédés de chiffrement utilisés en cryptographie. Elle consiste alors à casser des fonctions cryptographiques existantes, c'est-à-dire démontrer leur insécurité. La cryptanalyse utilise une intéressante combinaison de raisonnement analytique, d'application d'outils mathématiques, de découverte de redondances, de patience de détermination et de chance. La personne qui exerce la cryptanalyse est appelée **cryptanalyse**.

Les mots chiffrés respectivement déchiffrés ont la même signification que cryptés et décryptés**.**

2.3. Les lois de Shannon[24][22]:

Claude Elwood Shannon (1916-2001) est considéré comme le père de la théorie de l'information. Il a établi le lien entre l'algèbre de Boole et la commutation électronique, posant ainsi les bases des systèmes numériques actuels. C'est à lui que l'on doit la notion de « bit ». Il s'est illustré dans de nombreux domaines, notamment la cryptographie.

Les principes fondamentaux d'un algorithme de cryptographie sont basés sur deux notions essentielles énoncées par Shannon:

● **La confusion** vise à rendre le texte aussi peu lisible que possible. Ceci peut se faire par une substitution systématique de symboles ou par un algorithme de codage.

● **La diffusion** vise à rendre chaque élément d'information du texte chiffré dépendant d'un nombre aussi grand que possible d'éléments d'information du texte clair.[9]

2.4. Types de chiffrement :

2.4.1. Cryptographie symétrique et asymétrique:

En cryptographie, on distingue deux approches:

- La cryptographie symétrique,
- La cryptographie asymétrique.

Si les origines de la première sont très anciennes(2000av.J.-C).

Le principe de la cryptographie asymétrique n'apparaît qu'en 1976 [18][20].

1- Principe de la cryptographie symétrique:

Aussi appelée cryptographie conventionnelle ou à clés secrètes. On parle de cryptographie symétrique lorsque les clés de chiffrage et de déchiffrage sont les mêmes K=K' (clé secrète). Ceci implique que les deux parties se sont au préalable mises d'accord sur la clé à utiliser dans leur échange d'information[21].

Prenons l'exemple suivant:

- L'émetteur(A) et le destinataire(M) doivent se mettre d'accord préalablement sur la clé(k) à utiliser. Pour ceci, ils ne doivent pas utiliser le réseau de communication standard qui est susceptible d'être espionné (par D).
- Chaque fois que A veut transmettre un message (m) à M, il utilise sa clé secrète pour chiffrer (c=Ek (m)), et il envoie le résultat de ce chiffrage par l'intermédiaire du même canal.
- M utilise à son tour la même clé secrète et le même algorithme public pour déchiffrer le message codé qu'il a reçu.

Les principaux problèmes de cette méthode sont les suivants:

- Si D parvient à découvrir la clé, alors elle pourra déchiffrer tous les messages encodés avec cette clé.

Elle peut même se faire passer pour A ou M.

- Il est impératif que la clé ne soit connue que par A et M. Orl'acheminement de la clé entre A et M est un problème non trivial

(pas banal). On parle de « canal sûr » pour désigner le média qui servira à cet échange. Dans la pratique, il est difficile de trouver un canal sûr.

Pour garantir la sécurité du chiffrage symétrique, il faut donc éviter de transmettre la clé puisque le canal de transmission n'est pas sûr.

Résoudre ce problème revient à trouver une solution au paradoxe suivant:

- Comment transmettre la clé au destinataire sans la transmettre?

C'est en 1976 que ce problème apparemment insoluble fut résolu par deux chercheurs américains : Whitfield Diffie et Martin Hellman (assistés par Ralph Merkle). Ainsi naquit le principe de la cryptographie asymétrique, considéré comme la plus grande réussite de la cryptographie moderne.

2- Principe de la cryptographie asymétrique [38]**:**

Aussi appelée cryptographie publique ou à clés publiques. Ce type de cryptographie est basée sur deux clés séparées et distinctes, l'unepublique, l'autre maintenue secrète (privée). Et il est impossible de retrouver l'une à partir de l'autre. La source utilise la clé publique pour chiffrer le message alors que le destinataire utilise la clé privée pour déchiffrer le message **[Seulement les clés (publique et privée) du destinataire sont impliquées dans l'échange confidentiel]**.

Une des difficultés principales de la méthode symétrique est que chaque couple potentiel d'utilisateurs doit posséder sa propre clé secrète, et l'échanger par un moyen sécurisé avant leur premier échange d'information, ce qui peut s'avérer difficile à réaliser dans la pratique.

Le but d'un système à clé publique est de résoudre ce problème. La clé publique est généralement publiée dans un répertoire. L'avantage est

Donc que l'émetteur peut envoyer un message au récepteur sans communication privée préalable (il choisit sa clé privée, et la clé publique de l'émetteur). Le récepteur est la seule personne à pouvoir déchiffrer le message en appliquant sa clé secrète et personnelle, et la clé publique d'émetteur. On dit généralement que chaque clé déverrouille le code produit par l'autre. Il est intéressant de remarquer qu'avec ce système,

même l'émetteur qui a chiffré un message pour le récepteur, ne pourra déchiffrer le message ainsi codé.

Diffie et Hellman ne parvinrent pas à mettre en pratique leur admirable trouvaille[36][35].

C'est au MIT de Boston que trois chercheurs, sans le savoir, allaient résoudre le problème de Diffie et Hellman. C'est, en effet, en tentant de prouver que tout système à clé publique possède une faille que Ronald Rivest, Adi Shamir et Leonard Adelman aboutirent en1977 à l'algorithme à clé publique qui porte désormais leurs noms : RSA. [17][18] La sécurité de RSA repose sur la difficulté de factoriser des grands nombres entiers premiers.

3. La performance des algorithmes:

3.1. Algorithmes de bloc et de flot:

En cryptographie, on peut voir les données à chiffrer de deux manières différentes [31]:

- Le texte clair est divisé en blocs dont la taille est fixe et dépendante de la longueur de la clé. Puis, l'algorithme est appliqué successivement à chacun de ces blocs. Les algorithmes recourant à cette méthode sont dits« de blocs ».
- Le texte clair est chiffré bit après bit en temps réel. Cette méthode est adaptée au chiffrage d'information de nature synchrone comme, par exemple, la parole avec GSM. Les algorithmes recourant à cette méthode sont dits« de flot».

4. Outils de la cryptographie :

4.1. Les générateurs (pseudo)aléatoires:

La génération de nombre aléatoire est un processus très important pouvant compromettre la sécurité d'un bon nombre de systèmes de cryptage.

Entre autres applications on trouve la génération de clés de session (des clés utilisées qu'une seule fois pour envoyer d'autres clés), les vecteurs d'initialisation (DES - CBC mode), des secrets nécessaires à la génération de signature (El Gamal), etc.…

Un générateur aléatoire (*random generator*) est un dispositif capable de générer des nombres de façon *aléatoire*, *imprévisible (inattendu)* et *non reproductible*. Un tel générateur est normalement doté d'un dispositif externe mesurant des phénomènes physiques connus par leur non-déterminisme (une source radioactive ou quantique). Les générateurs pseudo-aléatoires sont des procédés déterministes développés à partir d'une séquence aléatoire initiale(*seed*) pouvant être obtenue par des méthodes diverses (la fréquence de frappe d'un utilisateur, le nombre d'accès disque, le nombre de paquets reçus par une interface réseau, etc.)

4.2. Les fonctions à sens unique :

La notion de *fonction à sens unique* est fondamentale pour la cryptographie à clé publique. Une telle fonction est une fonction aisée à calculer mais l'inverse est considérablement plus difficile. En d'autre terme, étant donné un x, il est facile de calculer f(x), mais étant donné f(x) il est difficile de calculer x. *«difficile»* veut dire qu'il faudrait des millions d'années pour calculer la fonction inverse. De nombreuses fonctions ont l'air d'être des fonctions à sens unique (d'un point de vue strictement mathématique les fonctions à sens unique n'existent pas)par exemple x^2dans un corps fini est

facile à calculer mais le calcul de $\sqrt{x}$ est nettement plus difficile. MD2, MD4, MD5, SHA, etc. sont des algorithmes basés sur des fonctions à sens unique.[31][32][14][15].

5. Cryptographie Contrainte ou nécessité:

La cryptographie est à la fois une contrainte et une nécessité.

5.1. Nécessité:

Dans bon nombre de domaines et du fait du manque de sûreté de certains protocoles de communication (ceux de courrier électronique, POP3), il est désormais nécessaire d'avoir recours à la cryptographie. Que tout le monde puisse lire le message où vous demandez à votre femme ce qu'elle fait à manger le soir même n'est en soit pas très gênant, mais que ces mêmes personnes puissent lire le rapport confidentiel sur l'étude de marché que vous venez de faire pour un client pose plus de problèmes [36].

5.2. Contrainte

Etre obligé de chiffrer chaque message avant de l'envoyer et de déchiffrer chaque message reçu est une très lourde contrainte. De plus pour chiffrer un fichier ou un message, il faut retaper à chaque fois la clef de chiffrement. Pour les personnes des relations publiques qui traitent plusieurs centaines de messages par jour, cela pourrait devenir très vite ingérable.

Il est donc nécessaire que soit établie une hiérarchie dans les données : publiques, confidentielles, secret industriel par exemple. Et ensuite

d'appliquer différents niveaux de cryptage à chaque donnée en fonction du niveau de sécurité requis.

6. Conclusion

La cryptographie a plus de 3 000 ans mais n'en est qu'à ses balbutiements. Hier réservée aux militaires, aujourd'hui utilisée uniquement par une élite maîtrisant les nouvelles technologies.

Demain la cryptographie devra à nouveau changer, d'une part pour être plus sûre en utilisant des techniques comme la cryptographie quantique et d'autre part arriver dans chaque ordinateur.

La cryptographie moderne a pris son essor dans les années 1970 avec DES et RSA que l'on peut considérer comme la première génération de « vrai » algorithmes(Enigma et stéganographie n'étant que des mécanismes). Aujourd'hui, la cryptographie s'est déjà imposée dans la vie courante en particulier pour la protection des transactions.

Chapitre II

Sommaire:

1. Introduction

La cryptographie est une science très ancienne qui date de 1900 ans avant J.-C... Des recherches indiquent qu'un scribe égyptien a employé des hiéroglyphes non conformes à la langue pour écrire un message. De ce temps-là et au long de l'histoire, la cryptographie a été utilisée exclusivement à des fins militaires. Aujourd'hui, les réseaux informatiques exigent une phase de cryptographie comme mécanisme fondamental afin d'assurer la confidentialité des informations numériques. En se rapportant à des événements historiques de notre vie informatisée [35], nous pouvons retenir : la machine Énigma créée par Dr Arthur Scherbius en 1923 et qui a été utilisée largement dans la seconde guerre mondiale; la théorie de l'information de Claude Shannon en 1949, l'algorithme lucifer développé par IBM en 1970, la théorie du système à clef publique par Whiteld Die et Martin Hellman en 1976 qui a marqué le début de la cryptographiemoderne, le premier standard pour le cryptage, l'algorithme DES en 1976, les résultats de la cryptographie quantique par Charles H. Bennett et Gilles Brassard en1990, et le standard AES en 2000, qui a remplacé le DES.

1.2. Caractéristiques d'un algorithme cryptographique

• Réversibilité

Lorsqu'on applique un algorithme *réversible* à un message, il est possible de retrouver le message original à partir du message chiffré en appliquant la transformation inverse. En accordant cette possibilité au destinataire du message, on a réalisé un système de chiffrement.

Par contre, lorsqu'on applique un algorithme *irréversible* à une donnée, il n'existe aucun moyen de retrouver le message clair à partir du cryptogramme. Il sera seulement possible de vérifier le message original par une nouvelle application de l'algorithme, et faire une comparaison avec le message chiffré. Ce type d'algorithme sert à effectuer des *contrôles d'intégrité* et/ou *d'origine*.

• **Symétrie**

Si la clé utilisée pour le déchiffrement est la même que celle utilisée pour le chiffrement ou si elle s'en déduit, on parle d'algorithme *symétrique*.

Par opposition, les algorithmes *asymétriques* utilisent pour le chiffrement et le déchiffrement des clés qui ne peuvent se déduire facilement l'une de l'autre.

• **Mode de chiffrement**

Le chiffrement d'un message peut être réalisé selon 2 procédés:

- Le chiffrement par blocs: le message est décomposé en bloc de longueur fixe avant de leur appliquer l'algorithme bloc par bloc.
- chiffrement par flux ou en continu : l'algorithme de chiffrement est appliqué à chaque élément binaire du message à chiffrer.

• **Résistance**

La fiabilité d'un algorithme de chiffrement est liée à sa capacité de résistance aux attaques menées par d'éventuels *décrypteurs* et qui se traduit en termes de temps nécessaire pour décrypter un message. L'algorithme est alors considéré comme cassé ou percé.

En raison du coût des télécommunications, un algorithme ne doit pas être trop expansif et générer un cryptogramme beaucoup plus volumineux que le clair.

La cryptanalyse est le processus permettant à un intrus de retrouver des informations secrètes (clés, message en clair) à partir d'informations publiques(cryptogrammes, algorithmes).

La stratégie adoptée par un cryptanalyse dépend de la nature de l'algorithme et des informations dont il dispose. On distingue trois méthodes de réalisation de cette attaque:

- ***Méthode systématique*** : le cryptanalyse procède par essais successifs sur la liste exhaustive des clés. En théorie, cela est toujours possible mais en mettant des moyens illimités (temps, espace mémoire, capacité de calcul). La protection est de choisir des clés longues et des algorithmes lents.

- ***Méthode analytique*** : à partir des algorithmes, trouver des relations mathématiques entre les différents paramètres (clés, message en clair, cryptogramme) pour déduire des informations sur le secret. La protection est d'utiliser des algorithmes complexes à exprimer sous forme d'équations.
- ***Méthode statistique*** : à partir d'un grand nombre de messages, trouver des relations statistiques entre le message en clair, le cryptogramme et les clés. La protection consiste à brouiller les statistiques et à réduire la redondance du message en clair.

• **Publication**: Toutes les spécifications d'un algorithme *publié* sont librement disponibles et toutes les étapes du chiffrement sont décrites dans la littérature.
Par contre, les spécifications d'un algorithme *non publié* restent le secret du petit cercle de ses concepteurs, élargi éventuellement à son commanditaire.

• **Implémentation:** Un algorithme peut être implémenté sous forme *logicielle* (programme) ou *matérielle*(puce).
La forme logicielle présente les avantages de portabilité. Un algorithme publié peut être mis en œuvre surtout type de machine ou de système d'exploitation.
La forme matérielle présente des avantages de rapidité et de protection, indispensable en l'occurrence pour les algorithmes non publiés.

• Autorisation de mise en œuvre L'usage des algorithmes de cryptographie en France est réglementé et soumis au régime des matériels de guerre de seconde catégorie. Leur mise en œuvre est régie par des décrets précisant les démarches à effectuer pour la fabrication, la commercialisation, l'acquisition, la détention et l'utilisation des moyens cryptographiques.
En fonction des algorithmes mis en œuvre, on distingue trois principales familles de systèmes cryptographiques: symétriques, asymétriques et irréversibles[21].

2. SYSTEMES CRYPTOGRAPHIQUES CLASSIQUES[16][12][11]

Les systèmes cryptographiques classiques sont basés sur deux types de transformations: les *transpositions* et les *substitutions*.

2.1. Algorithmes de transposition

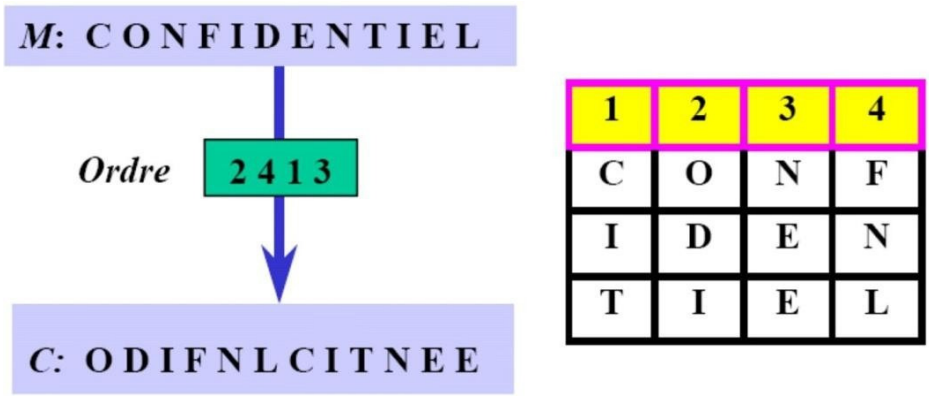
Les algorithmes de transposition réordonnent les caractères sur la base d'une figure géométrique. Le chiffrement a lieu en deux étapes selon le schéma ci-dessous:

Le message en clair est d'abord transcrit selon une figure et un mode d' "écriture" donnés. Le message chiffré est ensuite constitué en prenant les caractères selon un mode de "lecture". La clé est formée de la figure et des modes d'écriture et de lecture.

1-Transpositions par colonnes

Le message en clair est écrit ligne par ligne. Le message chiffré est obtenu en prenant les caractères colonne par colonne dans un certain ordre.

Exemple:



La détermination et le choix de l'ordre peut se faire sur la base d'un mot-clé, d'une phrase-clé ou encore d'un texte-clé pour rendre plus difficile la cryptanalyse.

2.2. Algorithmes de substitution

1- Substitution simple

Chaque caractère du message en clair et provenant d'un alphabet noté *A*, est remplacé par le caractère correspondant appartenant à un alphabet de substitution noté *S*. L'alphabet *S* est un réarrangement de l'ordre lexicographique des caractères de *A* .

Soit *A* un alphabet à n caractères :{a0,*a1,a3,..., an-1}*;

*C'*est représenté par: {*f(a0),f(a1) ,f(a3),..... ,f(an-1)*} où la transformation

f:A----->S

remplace chaque caractère de *A* par le caractère correspondant de *S*.

La clé est déterminée ici par l'alphabet *S* ou la fonction *f*...

Le message *M=m1m2m3 .....* est transformé en *Ek(M)=f (m1)f(m2)f(m3)…*

***Exemple*:** En prenant les alphabets A et S suivants:

A :	A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
S :	C E S A R B D F G H I J K L M N O P Q T U V W X Y Z

Le chiffrement du message *M* nous donne :

M:	C O N F I D E N T I E L
C:	S M L B G A R L T G R J

L'alphabet de substitution peut être construit à partir d'un mot-clé. Dans cet exemple, il s'agit du mot-clé CESAR.

Il peut être également obtenu par un simple décalage de l'alphabet *A*, de *k* positions, modulo la taille de *A*. Dans ce cas, *f (a) = (a + k) mod n, n* étant la taille de l'alphabet *A*. Ce type d'alphabet est désigné par le "Chiffre de César" car Jules César l'utilisait Avec *k =3*[20].

En comparant les fréquences moyennes des caractères du message chiffré avec ceux du langage utilisé, le cryptanalyse peut facilement retrouver la fonction *f*. Cette analyse peut être étendue à la distribution du bigramme et trigramme.

3. SYSTEMES SYMETRIQUES OU A CLES ECRETE:

La cryptographie à algorithmes symétriques utilise la même clé pour les processus de codage et de décodage ; cette clé est le plus souvent appelée "secrète" (en opposition à "privée") car toute la sécurité de l'ensemble est directement liée au fait que cette clé n'est connue que par l'expéditeur et le destinataire. La cryptographie symétrique est très utilisée et se caractérise par une grande rapidité (cryptage à la volée, "on-the-fly"), des implémentations aussi bien software (Kryp to Zone, firewalls logiciels type Firewall-1 et VPN-1 de Check point) que hardware (cartes dédiées, processeurs cryptos 8 à 32 bits, algorithmes câblés...) ce qui accélère nettement les débits et autorise son utilisation massive[24].

Ce type de cryptographie fonctionne habituellement suivant deux procédés différents, le cryptage par bloc set le cryptage de "Stream"(encontinu).

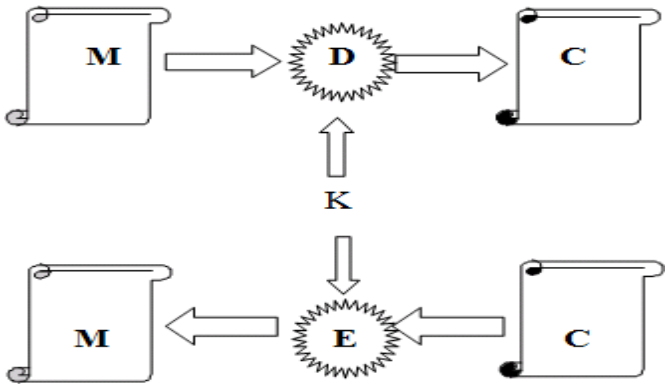


Fig 2.1.Système Symétrique

3.1 Le chiffrement par flot

Pour comprendre le cryptage en continu, il suffit de connaître par exemple les vidéos au format Real Vidéo très répandues sur internet : on visualise l'image au fur et à mesure que les données sont reçues. Le principe est le même dans le cas de «Stream-ciphers" : le cryptage est effectué bit-à-bit sans attendre la réception complète des données à crypter[31].

Une technique de chiffrement, du nom de "One-Time Pad" est utilisée pour chiffrer les flux. C'est le chiffrement inconditionnel le plus sûr. Pour cela, on a besoin d'une chaîne aléatoire de la même longueur que le message d'origine, ce qui n'est pas pratique. Le but d'un Stream cipher est de générer une chaîne aléatoire à partir d'une clé de longueur courte.
Une autre technique consiste à "xorer", c'est-à-dire à appliquer un OU exclusif(XOR) au message avec un autre message prédéfini. Bien entendu, cela nécessite que le destinataire (la personne qui décrypte) connaisse le message prédéfini et donc cela rajoute de la complexité au schéma général.
Les Stream-ciphers sont utilisés aujourd'hui par différentes applications. Pour chiffrer les flux, l'algorithme RC4 est très utilisé.

3.2 Le chiffrement par bloc[35]

Quatre modes de chiffrement par bloc sont utilisés : Electronic Code Book (ECB), Cipher Block Chaining (CBC), Cipher Feed Back (CFB) ou Output Feed Back (OFB). Le cryptage en blocs (block-cipher) est au contraire beaucoup plus utilisé et permet une meilleure sécurité. Les algorithmes concernés sont également plus connus (DES, AES, Skipjack...) ; leur nom vient du fait qu'ils s'appliquent à des blocs de données et non à des flux de bits (cf. Stream-ciphers). Ces blocs sont habituellement de 64 bits mais cela dépend entièrement de l'algorithme utilisé et de son implémentation. De même, la taille de la clé varie suivant l'algorithme et le niveau de sécurité requis ; ainsi, un cryptage de 40 bits (c'est-à-dire utilisant une clé longue de 40 bits) pourra être déclaré faible puisque aisément cassable. Un cryptage de 56 bits (qui est le standard dans le cas du DES) sera qualifié de moyen puisque cassable mais nécessitant pas mal de moyens pour être exploitable (vis-à-vis du temps requis et de la valeur des données). Enfin, un cryptage de 128 bits (valeur standard utilisée par Rijndael alias AES) est plutôt fort à l'heure actuelle.
Rappelons à cette occasion que la Loi de Moore prévoit le doublement de la puissance de calcul des processeurs tous les18 mois (Loi toujours vérifiée de la

fin des années 70 à nos jours). Sans entrer dans les détails, il faut savoir que le cassage de cryptés nécessite essentiellement des ressources processeur, RAM et éventuellement ROM ou disque dur si le cassage se fait par pré-calcul. L'évolution générale est donc extrêmement rapide, sans parler des ordinateurs plus perfectionnés (scientifiques ou autres), à architectures parallèles, ou distribuées...Il reste donc relatif de parler de sécurité absolue, en tout cas en ce qui concerne la cryptographie symétrique.

Les quatre modes cités précédemment sont plus ou moins indépendants de l'algorithme choisi. Toutefois, tous les algorithmes ne permettent pas d'utiliser tous les modes possibles. Pour mieux comprendre, voyons ces modes plus en détails. Pour désigner le processus de cryptage simple (tel que décrit précédemment). On utilisera la notation suivante :

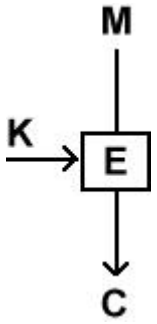


Fig 2.2.Mode Dictionnaire(EBC).

Où K désigne la clé utilisée par l'algorithme, E désigne le cryptage en lui-même ,M (ou m, mi) désigne le message en clair (c'est-à-dire un bloc) et C (ou c, ci) le chiffré résultant

3.2.1 Le mode Electronic Code Book (ECB) est le plus simple des modes et s'applique aux blocks ciphers. Il revient à crypter un bloc indépendamment des autres ; cela permet entre autre de crypter suivant un ordre aléatoire (bases de données, etc...)mais en contre partie, ce mode est très vulnérable aux attaques.

Il est par exemple possible de recenser tous les cryptés possibles (code books) puis par recoupements et analyses statistiques recomposer une partie du

Message original sans avoir tenté de casser la clé de chiffrement. Il demeure que si la clé fait 128 bits ou plus, cette attaque n'est pas exploitable en pratique de nos jours. Cette technique est sensible à l'inversion ou la duplication de blocs sans que le destinataire s'en aperçoive. On peut l'utiliser pour pipeline du hardware.

3.2.2 Le mode Cipher Block Chaining (CBC) peut être utilisé par les algorithmes en bloc. C'est d'ailleurs le mode le plus courant. Il permet d'introduire une complexité supplémentaire dans le processus de cryptage en créant une dépendance entre les blocs successifs ; autrement dit, le cryptage d'un bloc va être, d'une manière ou d'une autre, lié à/ou aux blocs/chiffrés précédents.Le schémas de base sera le suivant:

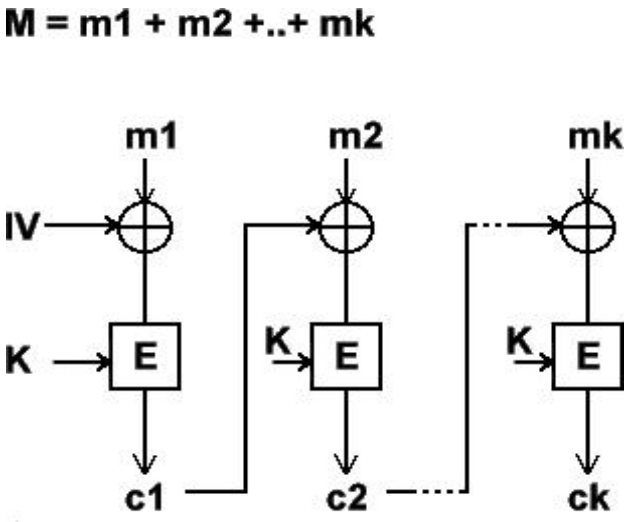


Fig 2.3. Cipher Block Chaining Mode (CBC)

Le message initial M est divisé en n blocs mi conformément aux spécifications de l'algorithme (par exemple en blocs de 64 bits). Chaque bloc donne un chiffré correspondant (ci) après cryptage suivant le même algorithme E utilisant la même clé K. Comme expliqué ci-dessus, le mode CBC introduit une dépendance entre deux cycles de cryptage: le chiffré obtenu aura n gi-1 est utilisé pour obtenirle chiffré du rang i. Concrètement, ce chiffré ci-1 subit un XOR avec le bloc mi. On peut se demander ce qu'il se passe lors du premier cycle d'encodage, lorsqu'il n'ya pas encore déchiffré à xorer avec notre premier bloc.

La réponse est que l'on utilise une valeur par défaut prédéfinie appelée Vecteur d'Initialisation (Initialisation Vector, IV). Ce vecteur d'initialisation change à chaque session, et doit être transmis au destinataire. Par contre, il n'est pas nécessaire de le chiffrer avant de l'envoyer : il peut être connu de l'adversaire. Il évite l'attaque sur le mode ECB en multipliant la taille de la base de données pré-calculée. Il ne faut néanmoins pas négliger l'importance de ce vecteur qui peut constituer une faille sérieuse s'il est mal choisi et compromettre ainsi l'intégrité de l'ensemble malgré l'utilisation de composantes fortes (algos, clés, etc.). Le déchiffrement est auto-synchronisé comme le mode EBC. Si on perd un bloc dechiffré, on pourra se resynchroniser en ne perdant que deux blocs.

3.2.3 Le mode Cipher FeedBack (CFB) est un mode destiné aux blocks ciphers dans le but d'en autoriser une utilisation plus souple, qui s'apparente plus à celle des algorithmes en continu. On peut le considérer comme un intermédiaire entre les deux. En effet, en partant d'un algorithme en bloc utilisant une longueur standard de n bits/blocs, le mode CFB va permettre de crypter des blocs dont la longueur pourra varier de n à 1 bits/blocs. Sachant que dans ce dernier cas, il serait plus économique en calculs d'utiliser directement un algorithme en continu. Quant au cas où la longueur est celle de l'algorithme (à savoir n), le schéma de CFB se simplifie et ressemble à celui deCBC(à quelques nuances près)

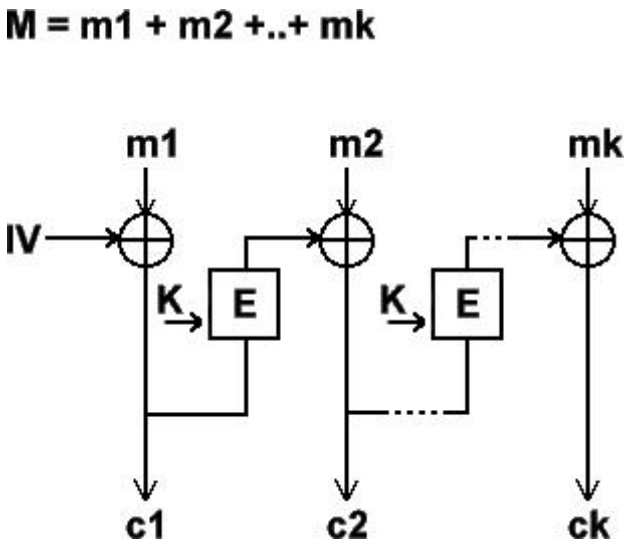


Fig 2.4.Mode Rebouclage sur la sortie(CFB).

3.2.4 Le mode Output FeedBack (OFB) est une variante de mode CFB précédemment abordé. Il est d'ailleurs parfois appelé ' internal feedback'. Il présente beaucoup de problèmes de sécurité et il est peu conseillé sauf dans le cas où sa longueur est égale à celle de l'algorithme utilisé.

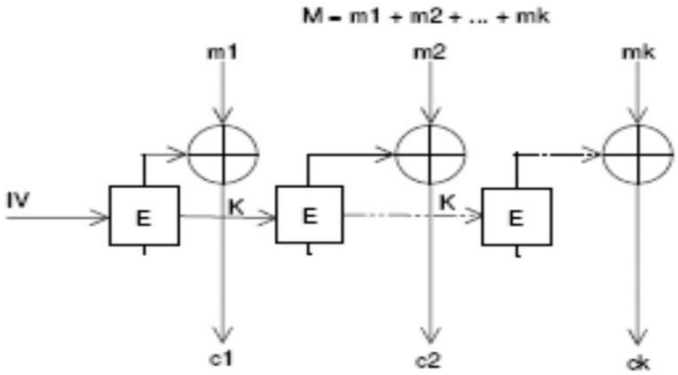


Fig 2.5. Mode OFB

3.3.DES(Data Encryptions Standard)[26]

La structure générale de DES est un schéma de Feistel, du nom du concepteur de l'algorithme Lucifer. Ce schéma utilise un certain nombre de tours (16, Dans le cas de DES) et permet de réaliser une permutation de l'espace des messages clairs dans l'espace des messages chiffrés, telle que la permutation inverse soit facilement implantable, et même, dans une large mesure, partage une partie de l'implantation de la première permutation.

On travaille sur des blocs de données binaires de largeur paire 2n(dans le cas de DES, 2n = 64) ; l'entrée du tour i est découpée en deux moitiés de taille n, la partie gauche Li et la partie droite Ri. Une fonction fi, dite (de Confusion), prenant une entrée de taille n et une sortie de même taille, est utilisée à chaque tour. Cette fonction n'a pas besoin d'être inversible, et elle est en général indexée par une clé. La sortie du tour i, qui est aussi l'entrée du tour i+1, est définie ainsi: $L_{i+1} = R_i$

$$R_{i+1} = L_i \text{ XOR } f_i(R_i)$$

Autrement dit, la moitié gauche est combinée par un ou exclusif avec le résultat de la fonction fi appliquée à la moitié droite; puis les deux moitiés Sont échangées. Il est facile de voir que ce tour est inversible, l'inverse se calcule ainsi: Ri=Li+1

Li=Ri+1 XOR fi(Li+1)

3.4 Les réseaux de Feistel[27]

Un réseau de Feistel est une méthode générale de transformation de n'importe quelle fonction en une permutation. Il a été inventé par Horst Feistel pour le design de Lucifer et popularise par le DES. On le retrouve dans bon nombre d'algorithmes de chiffrement par bloc dont CAST-128, Blowfish ou encore RC5. Concrètement, dans ce schéma, un bloc de texte en clair est découpé en deux ; la transformation est appliquée lors de ce tour à une des deux moities, et le résultat est combiné avec l'autre moitié par un ou exclusif. Les deux moitiés sont alors inversées pour l'application du tour suivant. Deux tours complémentaires forment un cycle. Lorsqu'un cycle est complet, chaque bit du bloc de texte à traiter a été modifié une fois.

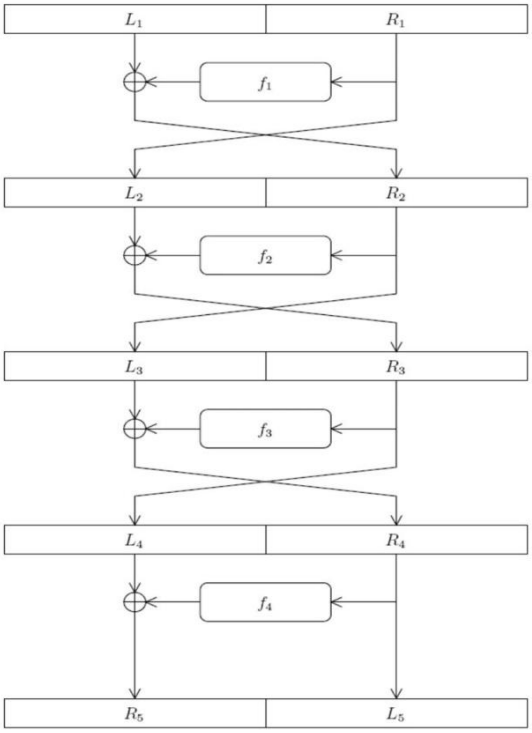


Fig 2.6 Un schéma de Feistel à quatre tours

3.5. IDEA(International Data Encryption Algorithm)[28]

Massey en 1990 puis renforcé par Bihamet Shamir en 1992. Dans IDEA, les processus de chiffrement et de génération de clés diffèrent complètement de ceux du DES: la taille de la clé est de 128 bits. Il n'utilise pas de tables de substitution *S*. La philosophie de conception de cet algorithme est basée sur le mélange d'opérations de différents groupes algébriques : le OU exclusif puis l'addition et la multiplication modulo 216. Ces opérations sont combinées de manière à obtenir une transformation complexe très difficile à cryptanalyse.

Le processus de génération des sous-clés est basé sur un mélange complexe d'opérations de décalage circulaire afin de fournir six clés secondaires pour les huit round de l'algorithme IDEA.

3.6. Blowfish

Blowfish est un algorithme développé en 1993 par Bruce Schneier, et devint rapidement l'une des alternatives la plus populaire du DES. Il a été conçu pour être facilement implémentable et avoir une vitesse d'exécution élevée. C'est un algorithme très condensé qui peut être exécuté dans une mémoire inférieure à 5 Kilo octets. L'une des principales caractéristiques de Blowfish est la longueur variable de sa clé qui peut atteindre 448bits. En pratique,les clés utilisées sont de 128bits. Le processus de chiffrement est composé de seize rounds.

En plus des tables S et de la fonction OU exclusif, Blowfish utilise également l'addition binaire. Contrairement au DES, Blowfish utilise des tables S dynamiques qui sont générées en fonction de la clé. Dans Blowfish, les clés secondaires et les tables S sont générées par l'application répétée de l'algorithme lui-même sur la clé.

Un total de 521 exécutions de l'algorithme de chiffrement Blowfish est nécessaire pour fournir les tables S et les clés secondaires. Par conséquent, Blowfish ne convient pas aux applications dans lesquelles les clés secrètes changent fréquemment[22].

3.7. RC5

RC5 est un algorithme développé en 1994 par Ron Rivest, un des inventeurs de l'algorithme à clé publique RSA. RC5 a été conçu pour avoir les caractéristiques suivantes:

- Il est convenable pour des implémentations logicielle ou matérielle : RC5n'utilise que des opérations de base communément existantes dans les microprocesseurs.
- Il est rapide et c'est un algorithme orienté mot. Les opérations élémentaires sont exécutées sur des mots entiers de données.
- Il est adaptable à tous les processeurs où le nombre de bits dans un mot est un paramètre variable dans RC5.
- Le nombre variable de rounds est un second paramètre dans RC5
- La longueur de clé variable est le troisième paramètre dans RC5.
- Sa structure simple est facile à implémenter et facilite la tâche de détermination de la force de cet algorithme.
- Il nécessite un faible besoin en mémoire ce qui rend l'algorithme très adapté aux cartes à puce et à d'autres équipements à mémoire réduite.
- Il est de haute sécurité : RC5 est destiné à procurer une sécurité élevée avec des paramètres adaptés
- Les rotations sont dépendantes des données : RC5 intègres des rotations(décalages circulaires) dont la quantité dépend des données. Ceci semble renforcer l'algorithme contre la cryptanalyse.

RC5 est utilisé dans bon nombre de produits de RSA Data Security.
Le récapitulatif des algorithmes symétriques est donné par le tableau suivant:

4. SYSTEMES ASYMETRIQUES OU A CLE PUBLIQUE[31][33]

Une des avancées les plus importantes de la cryptographie du XXème siècle concerne le développement du cryptage à clé publique. Les algorithmes à clé publique ou asymétrique, se fondent sur l'utilisation de clé de cryptage(publique) et de décryptage(privée). Les algorithmes à clé publique exigent que

le calcul de la clé privée à partir de la clé publique soit quasi impossible. Cette exigence permet de rendre publique la clé de cryptage sans affecter la sécurité de l'algorithme Whitfield Diffie et Martin Hellman furent les premiers à publier l'idée d'une cryptographie à clé publique en1976. Mais en1997, le CESG (Communication Electronics Security Group) du Royaume-Uni rendit publiques des informations auparavant secrètes indiquant que la cryptographie à clé publique fut inventée par James Ellis du CESG en 1970.

4.1. Algorithme RSA (Rivest, Shamir, Adleman)

Cet algorithme est très largement utilisé, par exemple dans les navigateurs pour les sites sécurisés et pour chiffrer les emails. Il est dans le domaine public. L'algorithme est remarquable par sa simplicité. Il est basé sur les nombres premiers.

Pour encrypter un message:

c=m^e mod n

Pour décrypter : **m =c^d mod n**

m=message en clair

c=message encrypté

(e,n) : constitue la clé publique

(d,n): constitue la clé privée

n: est le produit de 2 nombres premiers

^: est l'opération de l'élévation à la puissance(a^b:a puissance b)

mod : est l'opération de modulo (reste de la division entière). Créer une paire de clés est très simple, mais il ne faut pas choisir n'importe comment **e**, **d** et **n**. Et le calcul de ces trois nombres est tout de même délicat.

Voici comment procéder :

Prendre deux nombres premiers **p** et **q** (de taille à peu près égale).

Calculer **n=pq**.

Prendre un nombre **e** qui n'a aucun facteur en commun avec **(p-1)(q-1)**.

Calculer **d** tel que **ed mod (p-1)(q-1) = 1**

Le couple**(e,n)** constitue la clé publique. **(d,n)** est la clé privée.

Exemple:

Commençons par créer notre paire des clés:

Prenons 2 nombres premiers au hasard: **p=29**,**q=37**

On calcule **n=pq=29*37=1073**

On doit choisir **e** au hasard tel que **e** n'ai aucun facteur en commun avec **(p-1)(q-1)**: **(p-1)(q-1)=(29-1)(37-1)=1008**

On prend **e=71**

On choisit **d** tel que **71*d mod 1008 =1**

On trouve **d=1079**

On a maintenant nos clés:

La clé publique est **(e,n) = (71,1073)**(=clé d'encryptage)

La clé privée est **(d,n)=(1079,1073)**(=clé de décryptage)

Pour encrypter le message '**HELLO**'. On va prendre le code ASCII de chaque caractère et on les met bout à bout :

m=7269767679

Ensuite, il faut découper le message en blocs qui comportent moins de chiffres que **n**. **n** comporte 4 chiffres, on va donc découper notre message en blocs de 3 chiffres :

726 976 767 900

On complète avec des zéros.

Ensuite on encrypte chacun de ces blocs:

726^71	mod	1073	=	436
976^71	mod	1073	=	822
767^71	mod	1073	=	825

900^71mod1073=552

Le message encrypté est **436822825552**. On peut le décrypter avec d:

436^1079	Mod	1073	=	726
822^1079	mod	1073	=	976
825^1079	mod	1073	=	767
552^1079	mod	1073	=	900

C'est-à-dire la suite de chiffre **726976767900**.

On retrouve notre message en clair **7269767679** :'**HELLO**'

4.2. Algorithme d'El Gamal

Si RSA est un des algorithmes à clé publique les plus populaires, il en existe quelques autres. L'un d'entre eux, l'algorithme ElGamal, a été développé par Taher ElGamal en1984. Il n'est pas breveté et peut être utilisé librement.

La sécurité de cet algorithme se fonde sur la difficulté à calculer un algorithme discret dans un champ fini.

Fonctionnement de l'algorithme:

Sélectionner un nombre premier p et deux nombres aléatoires g et x, inférieurs

à p.

1. La clé publique correspond à la formule : $y = g^x \bmod p$. La clé privée est x. Si g, x et p sont des grands nombres, il est difficile de calculer x en connaissant y, g et p.

2. Pour crypter un bloc de message m, choisir un nombre aléatoire k, premier avec (p-1) et calculer $a=g^k \bmod p$ et $b=my^k \bmod p$. Le texte crypter de m

Est a et b.

3. Pour décrypter a et b, calculer m = (b/a^x) mod p.

Algorithme d'El Gamal

- p est premier
- g et x sont aléatoires et inférieur à p
- y= gx mod p
- k est un nombre aléatoire généré à chaque cryptage et qui est premier avec (p-1)
- m est le texte en clair
- a et b forment le texte crypté

5. Systèmes Irréversibles [22]

Dans un système irréversible, le message chiffré produit est un condensé du message original. Le message chiffré ***C*** est générée en utilisant un algorithme basé sur une *fonction de hachage* ***H*** ou à sens unique. On trouve plusieurs algorithmes basés sur ces fonctions: SH1,SH2, MD2,MD5, signature numérique

5.1. Algorithme de signature numérique:

Dans les systèmes manuels classiques, l'authentification de nombreux documents légaux, fiscaux et administratifs est déterminée par la présence de signatures manuscrites dûment mandatées.

Pour certaines applications spécifiques telles que les systèmes de virement bancaire et de transfert de fonds électronique, il est nécessaire d'avoir recours à des techniques prouvant l'authenticité et la véracité des messages reçus à travers un réseau de communication. En somme il faut concevoir un mécanisme remplaçant la signature manuscrite : *la signature numérique ou digitale*.

Pour la signature numérique, on compresse le message par un algorithme de compression pour en obtenir un sceau et on chiffre ce sceau par la clé secrète

du signataire. Ainsi, la signature est indissociablement liée au message signé et à l'identité du signataire.

Soit B le récepteur d'un message M signé par A. La signature émise par A doit satisfaire les conditions :

• B doit être en mesure de valider la signature de A sur le message M,

• Aucun autre correspondant, y compris B, ne doit être capable de produire la signature de A.

• Pour se protéger contre toute répudiation éventuelle par A, une tierce partie doit être en mesure de prouver l'émission de cette signature par A.

Les signatures numériques peuvent être implantées au moyen d'un système irréversible ou asymétrique[25].

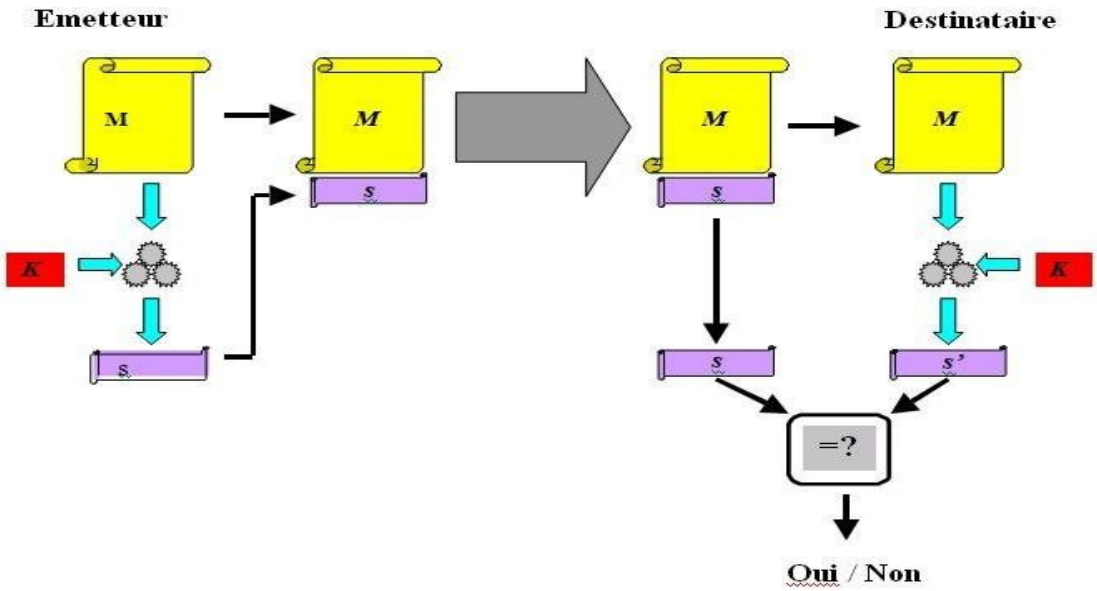


Fig 2.7 **principe de signature numérique**

5.2. Cryptographie Quantique

1. Principe

La cryptographie quantique est née au début des années 70. Elle repose sur le principe d'incertitude d'Heisenberg, selon lequel la mesure d'un système quantique perturbe ce système.

Dans tous les cas étudiés auparavant, il a été considéré que le canal de communication peut être espionné. Un système de cryptographie parfait devrait donc utiliser un canal de communication sur. Pour cela la cryptographie quantique propose une solution:

Si une personne tente d'intercepter les communications alors le message est modifié ou détruit. Pour cela, la cryptographie quantique utilise des photons polarisés à des angles de 0°, 45°,90° et 135°. [11]

L'espion ne peut obtenir des informations, même partielles, sur le message sans altérer celui-ci de manière imprévisible et incontrôlable. Le principe de base est que chaque bit est codé avec un seul photon, si celui-ci est capté alors le récepteur ne le reçoit pas ou alors de manière modifié et est ainsi mis au courant de l'écoute. De plus, ce système résout le problème de la distribution des clefs. Deux utilisateurs peuvent donc s'échanger des clefs et ce de manière complètement sûre. [7]

6.3 La stéganographie

1. Principe

La stéganographie consiste à cacher des informations dans un flot de données redondantes comme une image ou un texte. Ce mot provient du grec graphie, écriture-dessin et de stégano, couvert. La stéganographie est aussi la science qui consiste à cacher un message dans un autre texte stéganographie.

Conclusion:

Entre Cryptographie Symétrique et Asymétrique

Il existe des centaines d'algorithmes symétriques et asymétriques capables de fournir un niveau de confidentialité suffisant.

Les solutions symétriques offrent les avantages suivants:

• Rapidité (jusqu'à100 fois plus rapide que les solutions asymétriques)

• Facilité d'implantation en hardware

• Longueur de clé réduite : 128 bits (= 16 caractères => mémorisable !) au lieu de 1024 bits pour des équivalents asymétriques.

Les solutions asymétriques ont comme arguments principaux:

• Echange de clés simplifié : les clés doivent être échangées par un canal authentifié mais non-confidentiel.

• Gestion de clés simplifiée : une seule paire de clés publique/privé suffit à un utilisateur pour recevoir des messages confidentiels de n utilisateurs (au lieu de n clés différentes dans le cas symétrique).

Les problèmes propres aux deux techniques

• La gestion de clés par l'utilisateur reste le maillon le plus faible (niveau de connaissance).

• Sécurité (normalement) basée sur des arguments empiriques (expérimentaux) plutôt que théoriques.

• Restrictions légales **d'usage** et **d'exportation**

Chapitre III

Sommaire:

1. Introduction:

L'Advanced Encryption Standard(AES) le nouveau standard FIPS (Federal Information Processing Standard) Publication qui spécifiera un algorithme cryptographique à clé secrète.

Cet algorithme est utilisé par les organisations gouvernementales américaines pour protéger les informations sensibles (non-classifiées). Le NIST (National Institute of Standards and Technology) anticipe que l'AES sera aussi utilisé par d'autres organisations, institutions et individus en dehors du gouvernement américain et en dehors des Etats-Unis dans certains cas.

L'AES a été développé pour remplacer le DES(Data Encryption Standard).

Le DES est en train d'être abandonné et est aujourd'hui uniquement utilisé dans certains systèmes légaux. Cependant, le NIST prévoit que TripleDES(FIPS PUB 46-3) restera encore un algorithme approuvé pour les utilisations gouvernementales américaines[30].

Le 2 octobre 2000, le NIST a annoncé la sélection l'algorithme de chiffrement symétrique **Rijndael**. Cet algorithme AEA (Advanced Encryption Algorithm) a été développé par deux chercheurs belges :*Joan Daemen* travaille pour la compagnie Proton World International et *Vincent Rijmen* est chercheur au département d'électronique ESAT de l'Université Catholique de Louvain.

2. *Origine d'AES [*31]

Le 2 janvier 1997, l'Institut National des Standards et Technologies du Département du Commerce des Etats-Unis d'Amérique annonce le développement d'une norme nationale pour le traitement de l'information nommée Advanced Encryption Standard, ou Norme de Chiffrement Avancé. Apparu fin 1976, le DES (Data Encryption Standard) a alors près de vingt ans. Le NIST, lors de la dernière révision de sécurité de cet algorithme, en 1993, insère la note suivante dans le dossier:

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

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

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

– Etre public, utilisant un chiffrement symétrique par bloc;

– Avoir une clef dont la longueur pourra être incrémentée autant que nécessaire;

– Garder la capacité d'être implémenté de manière matérielle ainsi que logicielle;

– être nécessairement disponible librement, selon les termes inscrits dans les licences de l'American National Standards Institute(ANSI).

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

– La sécurité,
– L'efficacité informatique,
– L'utilisation de la mémoire,
– l'adéquation matérielle et logicielle,
– La simplicité,
– La flexibilité,
– Le respect des licences.

Un ensemble de données nécessaire à la réalisation d'un dossier de postulation est également présenté, afin que les candidats intéressés puissent d'ores et déjà rassembler les pièces requises. Les intéressés doivent fournir:

– Les spécifications complètes de l'algorithme, incluant les équations mathématiques, tables et paramètres adéquats
– Une implémentation logicielle ainsi que son code source programmé en langage C, pouvant être compilé sur un ordinateur personnel
– Ce programme permet la comparaison entre les diverses propositions, autant en termes de performances que d'utilisation mémoire
– Un rapport fournissant L'efficacité tant au niveau matériel que logique
– Un exemple de texte fournit en clair et sa forme chiffrée
– Si l'algorithme présenté est lié `à une quelconque licence ou `à un brevet dans tout ou partie, tous ces éléments devront être joints
– Une analyse des attaques connues permet de s'assurer ou non de la solidité,
– Pour finir, chaque concurrent devra faire une critique des atouts et limites de sa proposition.

Le 15 avril 1997, après avoir reçu trente-trois commentaires sur sa proposition, le NIST décide de valider son projet et d'en faire la base de travail de l'AES. C'est la première réunion de travail qui conduira à l'appel à candidatures le 12Septembre1997.

Cette réunion de travail fait ressortir les points essentiels que le NIST désirerait voir apparaitre dans l'appel à candidatures finales

– Tout d'abord, il confirme que toutes les soumissions seront rendues publiques, ce qui permettra l'analyse des algorithmes par tout un chacun

– Les auteurs préféreront un chiffrement par bloc, qui traite les données par bloc d'octets, à un chiffrement par flot, qui s'applique bit à bit sur les données, afin de rester compatible avec DES,

– Ces blocs pourront dépasser 64 bits, 128 bits semblant s'imposer comme un standard commun,

– Les implémentations devront pouvoir être éffectuées à la fois de manière logicielle et matérielle,

– Chaque proposition ne pourra faire l'objet de taxe d'utilisation(royalty free). Bien que les algorithmes sous licence soient acceptables, une préférence ira vers des licences libres de droit,

– Afin de limiter les risques de sécurité, une explication mathématique rationnelle est fortement recommandée,

– En matière de tests, il est demandé aux participants de fournir une implémentation de référence en langage Java et une, optimisée, en C. Les tests d'utilisation de la mémoire s'éffectueront sur une même et unique plateforme, dotée d'un processeur Pentium

– Le nombre de cycles et la taille de la clef devra être fixe,

– Pour finir, le NIST pense que la durée de vie d'AES estimée entre 20 e t30 années.

Le 12 septembre 1997, comme prévu dans le déroulement initial, l'appel à candidatures est lancé. Il fait preuve d'un souci de résultat et de sécurité dès le départ en expliquant aux candidats que l'AES devra être plus éfcace et plus résistant que ne l'est le triple DES.

Les candidats doivent expliquer de manière mathématique leurs algorithmes et les soumettre au test de MCT et KAT [MCT & KAT] afin de s'assurer de L'exactitude de leurs implémentations.

Le premier tour de la compétition devant désigner l'algorithme qui sera finalement choisi pour AES s''etend sur la période d'aout 1998 an avril 1999.

Deux réunions seront organisées:

La première conférence, AES1, se déroule du 20 au 22 aout 1998, à Ventura en Californie. Elle présente les quinze algorithmes retenus par la commission comme éligibles et en rejette six, jugés incomplets

Le tour numéro deux commence en aout 1999 pour se terminer en mai 2000. Il révèle enfin le short lis qui composera l'ultime épreuve. Sans grande surprise, on retrouve les choix précédemment évoqué, à savoir :

– Rijndael
– RC6
– Mars
– Twofish
– Serpent

Durant cette troisième et dernière conférence qui se déroule à New York, le NIST a déclaré que l'algorithme de AES est retenu.

2.2. Le choix du NIST :

Le 2 octobre 2000, le NIST a donné sa réponse. L'algorithme retenu est **Rijndael**, contraction des noms des deux inventeurs belges, Vincent **Rijmen** et Joan **Daemen**. Il est retenu pour son bon compromis entre sécurité, performance, efficacité, facilité d'implémentation et flexibilité.

Rijndael travaille par blocs de **128 bits** et il est symétrique. La taille de la clé est généralement de 128 bits avec les variantes 192 et 256 bits.

3. Les corps finis:

Définition : Un corps fini, appelé aussi champ de Galois noté ***GF(p)*** *(Galois Field),* tel que *p* est un nombre premier, est un corps de *p* éléments dans lequel on défini les deux opérations: l'*addition* et la *multiplication modulo p*.

Tous les corps finis ont les propriétés suivantes:

-L'*addition* et la *multiplication* sont des *lois internes*. Le résultat de l'addition ou de la multiplication de deux éléments est un élément du même corps fini.

-L'*élément neutre* de l'addition est noté par *0* et celui de la multiplication par1.

$\forall a \in GF(q)$**:**

$a + 0 = 0 + a =$

$a.a^* 1=1 *a= a.$

-L'*inverse* de *a* par rapport à l'addition est noté par$(-a)$ et son inverse par rapport à la multiplication est noté par (a^{-1}).

Quel que soit a, b, c $\in GF(q)$ *:*

Associativité : $a + (b + c) = (a +b) +c$

Commutativité: $a+b= b+ a.$

Distributivité: $a*(b+ c)=(a*b) +(a* c).$

$(b+c)* a=(b * a)+(c * a).$

4. La présentation de l'algorithme de 'RIJNDAEL':[32]

L'algorithme se présente en deux temps, tout d'abord une procédure d'expansion De la clef, puis la fonction principale de chiffrement.

La fonction de chiffrement se divise en trois : une transformation initiale Avec la clé, une série de **tours**(round) puis une transformation finale.

Le nombre de tours s'établit en fonction de la taille des blocs et de la clé:

{9 tours si la taille des blocs et de la clé sont de 128 bits},

{11 tours si la taille des blocs ou de la clé est de 192 bits},

{13 tours si la taille des blocs ou de la clé est de 256 bits}.

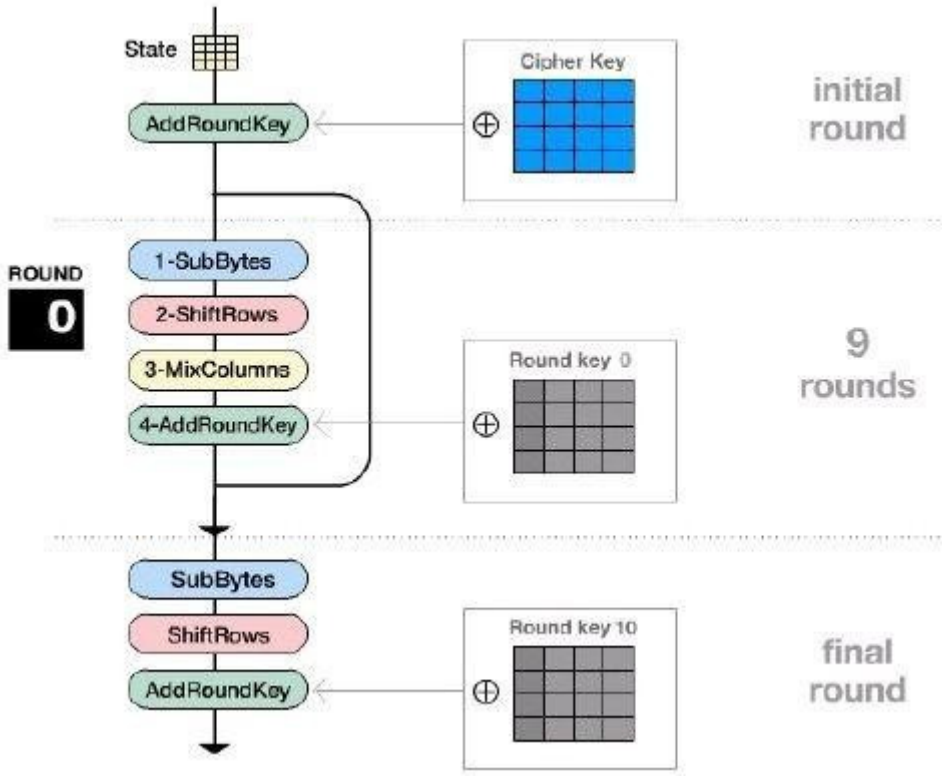


Fig 3.1 **Schémablocdel'algorithme,version128bits**

4.1 Mise en forme des

4.2 Entrées :

Comme tout système cryptographique symétrique, Rijndael dispose de deux entrées, à savoir le texte clair à chiffrer (*plaintext*) ainsi que la clé de cryptage (*key*). Si la longueur de la clé est fixe (on considère la version 128 bits de l'algorithme), la longueur du texte clair est variable d'une application à l'autre. Or Rijndael étant un algorithme de bloc, il doit commencer par découper le texte clair en blocs de 128bits.

Ensuite, le bloc de texte clair en cours de chiffrage ainsi que la clé sont mis sous forme matricielle.

Comme il s'agit de la version 128 bits de l'algorithme, les matrices obtenues sont carrées. Elles comptent 16 éléments et chacun de ces éléments contient 1 byte (16·8 = 128 bits). Pour d'évidentes raisons de taille, les valeurs binaires seront notées sous forme hexadécimale.

On considère les matrices suivantes, contenant des valeurs arbitraires:

32	88	31	E0
43	5A	31	37
F6	30	98	07
A8	8D	A2	34

Text Clair(State)

2B	28	AB	09
7E	AE	F7	CF
15	D2	15	4F
16	A6	88	3C

Clé(Key)

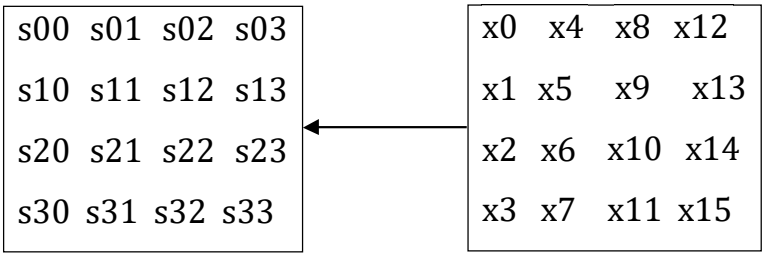
State: State est la structure qui contient les résultats intermédiaires.

C'est un tableau de 4x4 octets:

s00	s01	s02	s03s10
s11	s12	s13s20	s21
s22	s23	s30	s31
s32	s33		

Une telle structure a toujours 4 lignes et le nombre de colonnes égales la longueur de la clé divisée par 32

Assignation: L'assignation state=x est réalisée ainsi:



Où xi indique le ième octet de x.

4.3 Première étape–*Round0*:

La première étape consiste à combiner la matrice *State* (le bloc de texte clair) avec la clé. Cette opération s'appelle *Add Round Key*. Elle consiste à additionner modulo 2 (OU-exclusif ou XOR) chaque byte de la matrice *State* avec son homologue de la matrice *Key* (la clé originale). On obtient ainsi la nouvelle matrice *State* (son nom indique qu'elle désigne l'état actuel du bloc en cours de chiffrage). Elle constitue la matrice d'entrée de l'étape suivante.

4.4 Deuxième étape–*Rounds1-9*:

Cette partie du processus de chiffrage dépend de la taille de la clé utilisée. Comme on envisage la version 128 bits de l'algorithme, cette deuxième étape compte 9 itérations. Chacune de ces 9 itérations effectue successivement les quatre opérations détaillées ci-dessous :

1.Sub Bytes:

Cette opération consiste à remplacer chaque byte de la matrice State par une autre valeur. La substitution se fait à l'aide d'une table appelée S-Box et concrétise le **principe de confusion de Shannon**, ceci obscurcit la relation entre le texte en clair et le texte chiffré. Une S-box prend les bytes que cette table contient sont les bytes de remplacement. Ils sont définis sur la base mathématique des champs finis (ou champs de Galois). Les valeurs de la S-Box sont construites par deux fonctions dans deux champs finis différents ($GF(2^8)$ et $(GF(2))$.

hex	Y															
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
	0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
	1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
	2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
	3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
	4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	B3	29	e3	2f	84
	5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
	6	d0	ef	aa	fb	43	4d	33	85	45	F9	02	7f	50	3c	9f	a8
x	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

La substitution se fait de la manière suivante:

1. Prendre la matrice *State*:

19	A0	9A	E9
3D	F4	C6	F8
E3	E2	8D	48
BE	2B	2A	08

2. Pour chaque élément de cette matrice, procéder comme suit:
   - Le premier caractère hexadécimal de l'élément indique une ligne de la *S-Box* tandis que le deuxième indique une colonne.

- Le byte se trouvant à l'intersection ligne-colonne dans la *S-Box* est celui qui doit être substitué à celui de la matrice *State*.

Ainsi, le premier byte de *State* (premier élément de la matrice, soit 19_h) sera remplacé par le byte se trouvant à l'intersection de la ligne 1 et de la colonne 9 de la *S-Box*, c'est-à-dire $d4_h$.

Le deuxième byte de *State* ($3D_h$) sera remplacé par le byte se trouvant à l'intersection de la ligne3 et de la colonne D de la *S-Box*, c'est-à-dire 27_h.

Et ainsi de suite...

Après traitement des seize bytes, la matrice *State* devient:

D4	E0	B8	1E
27	BF	B4	41
11	98	5D	52
AE	F1	E5	30

2. Shift Rows:

Cette opération consiste à décaler des lignes dans la matrice *State*. En ce sens, elle garantit le **principe de diffusion de Shannon**. De faibles changements dans le texte clair impliquent de grands changements dans le texte chiffré. Les décalages ne modifient pas les valeurs des bytes, mais changent leur ordre.

Les décalages se font comme suit:

- La première ligne n'est pas décalée.
- La deuxième ligne est décalée de 1 byte vers la gauche.
- La troisième ligne est décalée de 2 bytes vers la gauche.
- La quatrième ligne est décalée de 3 bytes vers la gauche.

La matrice *State* issue de l'opération précédente (*Sub Bytes*) est la suivante:

D4	E0	B8	1E
27	BF	B4	41←
11	98	5D	52←←
AE	F1	E5	30←←←

Après les décalages, on obtient la nouvelle matrice *State*:

D4	E0	B8	1E
BF	B4	41	27
5D	52	11	98
30	AE	F1	E5

3. Mix Columns:

Cette opération participe également au **principe de diffusion** de Shannon. Elle consiste à multiplier une matrice constante avec la matrice *State*:

Matrice constante Matrice state Matrice résultat

02	03	01	01
01	02	03	01
01	01	02	03
03	01	01	02

*****

D4	E0	B8	1E
BF	B4	41	27
5D	52	11	98
30	AE	F1	E5

?	?	?	?
?	?	?	?
?	?	?	?
?	?	?	?

Pour obtenir le premier résultat (ligne1,colonne1)on procède comme suit:

02	03	01	01
01	02	03	01
01	01	02	03
03	01	01	02

*

D4	E0	B8	1E
BF	B4	41	27
5D	52	11	98
30	AE	F1	E5

?	?	?	?
?	?	?	?
?	?	?	?
?	?	?	?

Result[1,1]= 02•D4XOR03•BFXOR01•5DXOR01•30

La particularité de ce calcul est que les opérations d'addition et de multiplication se font dans le champ de Galois GF(2^8). Si les additions se réalisent par un simple OU-exclusif(XOR), les multiplications sont plus problématiques...

Multiplications dans le champ de Galois GF (2^8):

Deux méthodes permettent de calculer les produits dans GF(2^8).

1) Application mathématique:

Considérons le premier produit, à savoir 02•D4. Il faut voir les deux bytes à multiplier comme deux polynômes du septième degré a(x), respectivement b(x) dont les coefficients sont donnés par les 8bits composant le byte. Reprenons le calcul ci-dessus:

Result [1, 1] = 02•D4 XOR 03•BF X 01•5D. 01•30

02= 00000010a(x)

D4=11010100 b(x)

$a(x)=0x^7+0x^6+0x^5+0x^4+0x^3+0x^2+1x^1+0x^0=x$

$b(x)=1x^7+1x^6+0x^5+1x^4+0x^3+1x^2+0x^1+0x^0=x^7+x^6+x^4+x^2$

Maintenant que a(x) et b(x) sont connus, il reste à les multiplier modulo le polynôme de degré 8 $m(x) = x^8+x^4+x^3+x+1$. Ce polynôme est irréductible dans $GF(2^8)$, il ne peut être divisé que par lui-même et par 1:

$a(x)b(x) \bmod m(x) = x(x^7+x^6+x^4+x^2) \bmod x^8+x^4+x^3+x+1$

$a(x)b(x) \bmod m(x)=x^8+x^7+x^5+x^3 \bmod x^8+x^4+x^3+x+1$

Le résultat de cette expression est le polynôme représentant le produit 02•D4 dans le champ de Galois $GF(2^8)$.

Ce calcul ne sera mené à terme car l'objectif était juste de montrer le **lien entre Rijndael et les champs de Galois**. Il ne présente pas de réel intérêt dans la mesure où la deuxième méthode est non seulement plus simple, mais aussi plus adaptée à l'implémentation dans des circuits programmables.

2) Propriétés du calcul binaire:

Dans le calcul binaire, certaines opérations sont particulièrement simples.C'est notamment le cas de la multiplication par 2 qu'on réalise par un simple décalage des bits vers la gauche.

Soit N, un nombre binaire arbitraire et 2N, son double:

N=10101110 2N=101011100

Cette propriété simplifie considérablement les multiplications dans le champ de Galois. La matrice constante ne compte que 3 valeurs différentes: 01,02 et 03. Les valeurs de la matrice *State* sont donc toutes multipliées par l'une de ces trois valeurs. Elles sont toutes faciles à réaliser en binaire, mais montrent tout de même quelques particularités liées au champ de Galois:

Multiplication par 01(00000001 en binaire):

Le nombre reste inchangé!

Multiplication par 02(00000010 en binaire):

Les bits du nombre sont décalés vers la gauche:

N=10101110 2N=101011100

Puisque les opérations se font dans un nombre fini des valeurs (champ de Galois GF(2^8) de 256 valeurs), le MSB de 2Ndoit être omis (effacé):

2N=**1**01011100

2N=01011100

.Si le MSB(que nous venons d'omettre) étai tun'0', alors 2N est le résultat final De la multiplication:2N=01011100

. Si par contre le MSB (que nous venons d'omettre) était un '1', ce qui est le cas dans cet exemple, alors il faut encore additionner (XOR) le nombre binaire 00011011(1B) à 2N afin de compenser la perte du MSB occasionnée (provoquée) par le décalage :

2NXOR00011011=01011100XOR00011011=01000111

2N=01000111

Multiplication par 03(00000011 en binaire):

La multiplication par 03 peut être vue comme une multiplication par 02 suivie d'une Addition modulo 2 du nombre multiplié. On peut illustrer ceci par un petit exemple en Notation décimale (N=6) :

3N=2N+N=2•6+6=12+6= 18(=3•6)

Voyons maintenant un exemple binaire:

N=00101110

2N=01011100

3N=2NXORN=01011100XOR00101110=01110010

Grâce à ces règles de multiplication, nous pouvons reprendre l'expression permettant de calculer le premier élément de la matrice *Résultat*:

Result[1,1]=02•D4XOR03•BFXOR01•5DXOR01•30

02•D4:

1•D4=11010100, MSB=1

2•D4=10101000XOR00011011=10110011

03•BF:

1•BF=10111111, MSB=1

2•BF=01111110XOR00011011=01100101

3•BF=2•BFXORBF=01100101XOR10111111=11011010

01•5D:

1•5D=01011101

01•30:

1•30=00110000

Il reste à additionner modulo 2 les quatre résultats des multiplications pour obtenir le Premier élément de la matrice *Résultat*:

Resultat[1,1]=10110011XOR11011010XOR01011101XOR00110000

Résult[1,1]=00000100=04_h

L'opération *Mix Columns* est particulièrement laborieuse (pénible) puisque les 15 autres éléments de la matrice doivent tous être calculés de cette manière !Voici donc le résultat final: la matrice *Résul t*(cette matrice devient naturellement la matrice *State* de la prochaine opération) :

04	E0	48	28
66	CB	F8	06
81	19	D3	26
E5	9A	7A	4C

4. Add Round Key:

Lors du processus de chiffrage, Rijndael transforme la clé (matrice *Key*).

01	02	04	08	10	20	40	80	1B	36
00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00

A chaque itération (*Round*), une matrice *Key* différente est utilisée (*RoundN Key*). Ceci permet d'éliminer les attaques liées à la clé (Biham) en faisant disparaître la symétrie.

Pour obtenir les 10 nouvelles clés nécessaires (puisque la version 128 bits de l'algorithme compte en tout 11 *Rounds*), Rijndael procède à une opération appelée *Key Scheduling* ou *Key Expansion* (cette opération sera expliquée au point suivant).

Si la clé change à chaque *Round*, l'opération *Add Round Key*, elle, reste simple. Elle consiste, tout comme celle de la première étape, à additionner modulo 2 (XOR) la matrice *State* et la clé du *Round* en cours(*RoundN Key*).

3.4) Troisième étape–*Round10*:

Cette étape est quasiment identique à l'un des neuf *Rounds* de la deuxième étape. La seule différence est que, dans ce dernier *Round*, l'opération *Mix Columns* n'est pas effectuée.

3.5) Extension de la clé –*Key Expansion*:

Il s'agit ici de voir comment **Rijndael** opère pour déduire les 10 clés secondaires dont il a besoin pour aboutir au texte chiffré.

On considère les deux matrices suivantes (*Rcon* est donnée. Elle est constante)

Clé original(key) Rcon(*Round constant word array*)

2B	28	AB	09
7E	AE	F7	CF
15	D2	15	4F
16	A6	88	3C

01	02	04	08	10	20	40	80	1B	36
00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00

Les quatre vecteurs composant la matrice *Key* sont appelés *Words* (ce sont des mots de 32bits). On commence par calculer le cinquième vecteur(Wi) :

W_{i-4} $W_{i-1}W_i$

2B	28	AB	09	?
7E	AE	F7	CF	?
15	D2	15	4F	?
16	A6	88	3C	?

On prend le vecteur Wi-1 auquel on applique une opération appelée *RotWord* qui consiste en un simple décalage des quatre bytes du vecteur vers le haut.

Au résultat on [9][13]. Applique encore l'opération *Sub Bytes*:

W_{i-1}	*RotWord*(W_{i-1})	SubBytes{*RotWord*(W_{i-1})}
CF	09	8A
4F	CF	84
3C	4F	EB
09	3C	01

Le vecteur obtenu (Sub Bytes {*Rot Word*(Wi-1)}) doit encore être additionné modulo 2 avec le vecteur Wi-4 ainsi que le premier vecteur de la matrice *Rcon*

W_{i-4}		*Rcon*(colonne1)		Sub Bytes{*Rot Word*(W_{i-1})}
2B		01		8A
7E	XOR	00	XOR	84
15		00		EB
16		00		01

Le vecteur résultant de cette addition est Wi:

A0
FA
FE
17

Il constitue le premier des quatre vecteurs de la deuxième clé:

2B	28	AB	09	A0
7E	AE	F7	CF	FA
15	D2	15	4F	FE
16	A6	88	3C	17

Le deuxième vecteur de la deuxième clé s'obtient plus simplement que le premier. En effet, il est donné par Wi=Wi-4XORWi-1:

Wi-4 Wi-1Wi

2B	28	AB	09	A0	?
7E	AE	F7	CF	FA	?
15	D2	15	4F	FE	?
16	A6	88	3C	17	?

2B	28	AB	09	A0	88
7E	AE	F7	CF	FA	54
15	D2	15	4F	FE	2C
16	A6	88	3C	17	B1

Le troisième vecteur de la deuxième clé s'obtient de la même manière que le deuxième. On a donc à nouveau Wi= Wi-4XORWi-1:

Wi-4 Wi-1WI

2B	28	AB	09	A0	88	?
7E	AE	F7	CF	FA	54	?
15	D2	15	4F	FE	2C	?
16	A6	88	3C	17	B1	?

2B	28	AB	09	A0	88	23
7E	AE	F7	CF	FA	54	A3
15	D2	15	4F	FE	2C	39
16	A6	88	3C	17	B1	39

Le quatrième et dernier vecteur de la deuxième clé s'obtient de la même manière que les deuxième et troisième (Wi= Wi-4XORWi-1):

Wi-4 Wi-1Wi

2B	28	AB	09	A0	88	23	?
7E	AE	F7	CF	FA	54	A3	?
15	D2	15	4F	FE	2C	39	?
16	A6	88	3C	17	B1	39	?

2B	28	AB	09	A0	88	23	2A
7E	AE	F7	CF	FA	54	A3	6C
15	D2	15	4F	FE	2C	39	76
16	A6	88	3C	17	B1	39	05

Les quatre vecteurs de la deuxième clé sont maintenant définis. La matrice *Key* a doublé de taille. Elle contient pour l'instant les clés des *Rounds 0* et *1*:

2B	28	AB	09
7E	AE	F7	CF
15	D2	15	4F
16	A6	88	3C

Round0

A0	88	23	2A
FA	54	A3	6C
FE	2C	39	76
17	B1	39	05

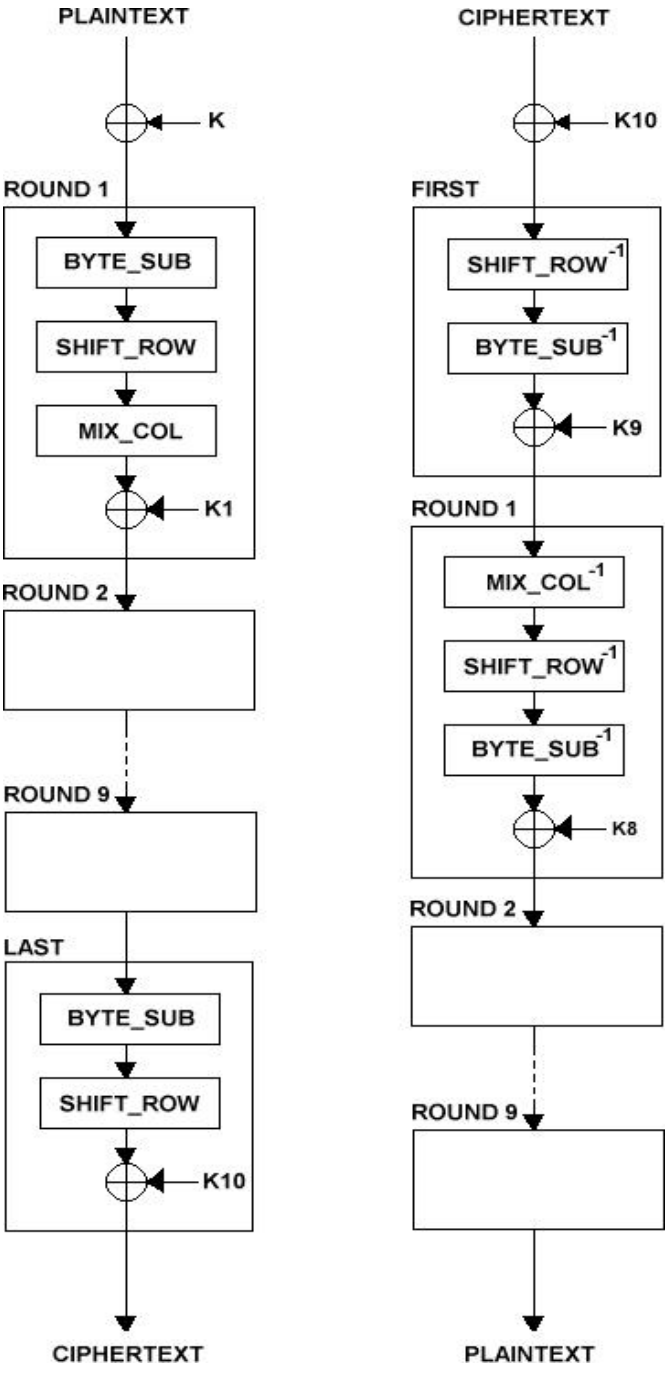
Round1Key

Il reste à faire 9 fois l'intégralité de cette démarche d'extension de la clé pour trouver les 9 autres clés secondaires...

2B	28	AB	09	A0	88	23	2A	?	?	?	?	……..
7E	AE	F7	CF	FA	54	A3	6C	?	?	?	?	……..
15	D2	15	4F	FE	2C	39	76	?	?	?	?	……..
16	A6	88	3C	17	B1	39	05	?	?	?	?	……..

Round0Key *Round1Key* *Round2Key* *Round10Key*

4.6. Schéma général du paire Chiffrage/Déchiffrage



4.7. Le déchiffrage:

Toutes les opérations réalisées lors du chiffrage sont réversibles, à condition d'avoir la clé bien entendu :

.Mohamed procède à l'extension de la clé de la même manière qu'Ali l'a fait lors du chiffrage.

.Les additions modulo 2(XOR) effectuées lors de l'opération *Add Round Key* Sont réversibles (en effet,(AXORB)XORB= A).

. L'opération *Sub Bytes* est inversée en utilisant **la table S inverse (*Inverse S-Box*)**. Si par exemple la *S-Box* indique le byte F7 (ligne 2, colonne 6), alors la *Inverse S-Box* restituera le byte 26(ligneF,colonne7).

Remarque: La Sbox a une explication algébrique. Soit z l'octet à modifier :

```
Byte Byte Sub (byte z)
{
If(z!=0)
Z=z^-1 (l'inverse de z dans GF
(2^8))c=011000111
for (i=0; i<8; ++i)
b[i]=z[i]+z[i+4]+z[i+5]+z[i+6]+z[i+7]+c[i] mod 2
return(b)
}
```

Les décalages de l'opération *Shift Rows* sont inversés, c'est-à-dire effectués vers la droite.

La multiplication matricielle de l'opération *Mix Columns* nécessite uneinversion de la matrice. Une fois la matrice inverse obtenue, la manipulation est la même que pour l'opération *Mix Columns* faite lors du chiffrage.

La matrice constante inverse est de:

14	11	13	9
9	14	11	13
13	9	14	11
11	13	9	14

A cause du calcul de la matrice inverse, le déchiffrage est environ 30% plus lent que le chiffrage.

5. Caractéristiques et critique de l'algorithme de *Rijndael:*

Nous allons énumérer dans ce qui suit l'Analyse de algorithme ***Rijndael*** faite par Jean-Philippe Gaulier -Conservatoire National des Arts et Métiers-Centre Régional Associé de Limoges- (Mémoire de synthèse soumis dans le cadre d'un probatoire en vue de l'acquisition d'un diplôme enIngénierie et Intégration Informatique Systèmes d'Information)[2]

5.1. Sécuritégénérale:

Rijndael n'est sensible à aucune attaque de sécurité connue. Il utilise des S-boxes comme composants non linéaires. L'algorithme semble garder une marge de sécurité adéquate, bien qu'il ait reçu des critiques sur sa structure mathématique qui pourrait mener à des attaques. D'un autre côté, cette simplicité a très largement facilité son analyse. [2]

5.2. Implémentations logicielle:

Rijndael fonctionne de manière optimale sur les plate-formes 8bits, 64 bits et DSP(digital signal processing: classe de processeur traitant les signaux).

5.3. Attaques sur les implémentations:

Les opérations utilisées par Rijndael sont parmi les plus faciles à défendre

Contre des attaques par analyse de puissance ou de temps. L'utilisation de techniques pour fournir quelques défenses de plus l'algorithme n'influe pas sur ses performances, contrairement aux autres finalistes.

Sa demande en RAM reste raisonnable. Il apparaît donc comme ayant

Un avantage vis-à-vis de ses compétiteurs lorsque l'on considère la mise en

Place de ces protections. Cependant, ces implémentations restent faillibles(pas sûr) à Des attaques par analyse de puissance.[2]

5.4. Chiffrement et Déchiffrement:

Le chiffrement et le déchiffrement différent. L'étude sur FPGA démontre que l'implémentation des deux transformations prend 60% d'espace supplémentaire que le chiffrement seul. La vitesse ne varie pas de façon significative entre les deux, bien que la mise en place de la clef soit plus longue pour le déchiffrement.[33]

5.5. Manipulation de clef:

Rijndael supporte le calcul de sous-clés **à la volée** pour le chiffrement.

Il nécessite une exécution préalable pour la génération dessous-clés de Déchiffrement pour une clef spécifique. Cela alourdit légèrement la manipulation de ses clefs.

5.6. Souplesse et flexibilité:

L'algorithme supporte pleinement les blocs et les clefs de taille 128 bits, 192 bits et 256 bits, selon n'importe quelle combinaison. En principe, sa structure peut s'accommoder (s'arranger) à tout bloc et toute clef qui serait multiple de 32, du moment que le nombre de tours adéquat est spécifié

5.7. Potentiel de parallélisations:

Le support du parallélisme pour le chiffrement d'un bloc simple est excellent.[2]

Conclusion:

Le principe de la thermodynamique suffit à démontrer que la recherche exhaustive de clés comportant 192 ou 256 bits est irréalisable. D'autre part, l'utilisation de la S-Box constitue une réelle difficulté pour les cryptanalyses. L'opération Mix Columns combinée avec Shift Rows fait que, après les nombreux rounds, tous les bits de sortie dépendent de tous les bits d'entrée. Ceci aussi rend la cryptanalyse difficile. L'utilisation des clés secondaires construites par extension de la clé originale, quant à elle, complique les attaques liées à la clé en cassant les symétries.

Rijndael est un algorithme sûr et va probablement le demeurer encore une vingtaine d'années.

Plus que l'évolution technologique, les progrès en mathématiques sont de plus en plus la cause de vulnérabilité (fragilité) des algorithmes de cryptographie modernes. Ceci s'explique d'une part par le fait que les attaques exhaustives (force brute) deviennent irréalisables (clés trop longues), mais aussi par les nombreuses thèses de doctorat que ce domaine de la technologie de l'information suscite (provoque).

Les prochaines années risquent d'apporter des développements très intéressants dans la cryptanalyse de AES et d'un bon nombre de *block ciphers* itératifs, mais même si AES reste un algorithme robuste à ce jour, il convient de surveiller de près les travaux de recherche algébrique qui pourraient révolutionner toutes les théories de conception actuelles relatives aux *blockciphers*, c'est pour cela que le F.I.S.P(Federal Information Processing Standard) reste très prudente pour ce qui concerne l'utilisation de l'AES dans les domaine sensibles.

Chapitre IV

Sommaire:

1. Introduction

Les circuits FPGA(*Field Programmable Gate Arrays* ou réseaux logiques programmables) sont des composants VLSI entièrement reconfigurables. Ils sont particulièrement bien adaptés à des implantations matérielles d'applications car ils sont capables de manipuler de grandes quantités d'informations et fournissent une puissance de calcul importante. La possibilité de reconfiguration sur ces circuits permet une minimisation et une adaptation spécifique à chaque algorithme implanté. Dans un circuit programmable, on peut implanter des algorithmes au niveau matériel avec un parallélisme au niveau calcul, ce qui conduit à des performances supérieures par rapport à des processeurs DSP.

LatechnologieFPGAestdevenueen10ansunealternativeintéressanteauxcircuits ASIC. Un FPGA est une puce VLSI regroupant des éléments logiques, des entrées/sorties et des ressources de routage, tous reprogrammables. Le concepteur peut réaliser son circuit en programmant le FPGA directement, évitant de ce fait le processus de fabrication long et coûteux des circuits ASIC. La plus répandue des technologies FPGA est celle des FPGA à base de RAM, où la programmabilité est obtenue en implantant les composants du FPGA à l'aide de RAM statique. En comparaison avec d'autres types de circuits, les FPGA offrent un délai de conception relativement court et permettent de tester facilement une réalisation. Parmi les désavantages des FPGA, nous pouvons citer un prix unitaire élevé, des performances électriques inférieures aux puces spécialisées (notamment en fréquence) et un taux d'utilisation du circuit en général assez faible.

Dans ce chapitre, nous allons d'abord présenter quelques généralités sur les circuits FPGA avant de décrire l'architecture générale du composant Xilinx VIRTEX en détail. Enfin, nous décrivons toutes les phases stratégiques de développement d'un circuit FPGA.

2- Généralités:

2.1-Les circuits programmables :[48],[46]

Les circuits programmables représentent une solution intermédiaire entre les systèmes généralistes (processeurs superscalaire et superpipeline) et les architectures complètement dédiées (composants spécifiques à très haute densité appelés ASIC: Application Specific Integrated Component). Ce type de circuit est basé sur la logique programmable. La logique programmable est née du besoin simultané d'intégration et d'optimisation des coûts de développement et d'industrialisation. Elle permet de réduire les temps de mise sur le marché des produits et de gérer à moindre coût les évolutions d'un système. C'est une technologie en plein essor, dont le produit phare est le FPGA (Field Programmable Gate Array).

Ces circuits permettent de spécialiser une architecture à moindre coût, par programme, sans passer par le processus de fonderie long et coûteux des circuits ASIC. Les calculs étant réalisés de manière câblée et spécialisée, des gains importants en performance peuvent être obtenus par rapport à une exécution sur processeur classique. Certains circuits sont de plus reconfigurables à volonté, c'est-à-dire qu'ils peuvent être dédié à une application de manière temporaire, puis reprogrammés pour une toute autre application. Pour être performant, un composant programmable doit disposer de ressources logiques performantes (en terme de fréquence et de puissance logique) et d'un réseau d'interconnexions souple et rapide.**[49]**

Les premiers composants programmables par l'utilisateur sont apparus dans les années 80, il s'agissait de circuits de type PAL (Programmable Array Logic), PLD(Programmable Logic Device) et CPLD (Complex PLD). Au milieu des années 80 sont apparus les FPGAs, permettant un degré d'intégration très élevé. Le FPGA concilie la possibilité d'intégration de fonctions complexes et performante(jusque là domaine réservé des ASICs) avec la souplesse et la flexibilité apportées

Par la logique programmable. Les capacités d'intégration, de plus en plus poussées au fur et à mesure de l'apparition des technologies et de l'évolution des architectures programmables permettent aujourd'hui d'intégrer jusqu'à plusieurs millions de portes logiques sur un même FPGA de type SRAM, reconfigurable à volonté.

2.2-Les circuits FPGA(Field Programmable Gate Array):

Les FPGA sont des circuits numériques à architecture programmable et à faible temps de développement. Ils sont composés de blocs logiques élémentaires(plusieurs milliers de portes) qui peuvent être interconnectés**.** Le domaine d'utilisation de ces circuits sont les systèmes séquentiels synchrones; la cellule de base de ces composants est un élément mémoire (bascule, registre) associé à un module combinatoire (les portes) ; la constitution du système consistera alors à associer ces cellules **[49]**. La puissance de ces circuits est telle qu'ils peuvent être composés de plusieurs milliers, voire millions de portes logiques et de bascules. Les dernières générations de FPGA intègrent même de la mémoire vive(RAM). Les deux plus grands constructeurs de FPGA sont XILINX et ALTERA. C'est dans la réalisation des cellules, dans leur association (routage) et dans les organes d'entrées/sorties que chaque constructeur va apporter des spécificités à ses produits.

Les circuits FPGA se divise en deux grandes catégories suivant la technologie d'interconnexion utilisée: **[46]**

- Les FPGAs de type SRAM : Les FPGAs SRAM sont composés de cellules logiques de type PAL, disposées généralement en matrice2D, et interconnectées de manière plus ou moins régulière par des ressources de routage elles même programmables. Des blocs d'entrée/sortie, disposés à la périphérie du circuit, fournissent l'interface avec l'extérieur.
- Les fonctions logiques sont implantées à l'aide de tables de vérité (LUT : Look Up Table) stockées dans des SRAM, d'où le nom de ce type de FPGA

- Les FPGAs de type anti-fusibles ou anti-fuse : Les blocs logiques de ce type de FPGA sont généralement beaucoup plus complexes que ceux des SRAM. La fonction est physiquement figée car la connexion est définitivement établie entre deux points. Des multiplexeurs câblés constituent l'essentiel des ressources de codage.

Les circuits à anti-fusibles offrent de meilleures performances en délais que les circuits SRAM, à cause de la finesse de leur point de programmation. Par contre, ils ont l'inconvénient de ne pas être reprogrammables et de ne pas bénéficier, comme c'est le cas des circuits SRAM, des progrès réalisés dans le domaine de l'intégration des mémoires volatiles. Ce sont donc ces derniers qui sont les plus utilisés. Par conséquent, dans ce chapitre, nous nous intéressons uniquement aux circuits utilisant la technologie SRAM.

2.3-Avantages de l'utilisation des FPGA:[45]

1- Souplesse de programmation, qui permet l'emploi conjoint d'outils des schématique aussi bien que l'exploitation d'un langage de haut niveau tel que le VHDL.
2- Distribution des horloges et faibles délais(connexions pipelinées…)
3- Disponibilité de connexions (peut limiter l'usage des ressources fonctionnelles)
4- Peut être associé à des processeurs
5- Bien adapté à l'optimisation de traitements réguliers(flot de données)
6- Manipule bien les informations binaires, riches mémoires distribuées, énorme bande passante. Grande richesse des parallélismes potentiels.
7- Possibilité de reconfiguration dynamique partielle ou totale du circuit

2.4-Domaine d'utilisation des FPGA :[47]

Les circuits FPGA sont associés à un large spectre d'application. Ils sont le plus souvent associés à des développements de circuits spécialisés à diffuser en petite quantité ou nécessitant un délai de mise sur le marché rapide. Leur reconfigurabilité sur site est le plus souvent utilisée lors de la phase de développement, pour effectuer le prototypage du circuit, mais plus rarement en exploitation. Les FPGA peuvent servir dans trois types de domaine:

1- *Le prototypage rapide*: Un FPGA permet de transférer une nouvelle application sur silicium en quelques heures alors que la réalisation des circuits intégrés (ASICs) nécessite des semaines voire des mois. Ceci présente l'avantage de pouvoir évaluer plusieurs options d'architecture de réalisation pour une application donnée.

2- *L'émulation du matériel*: Ceci est particulièrement efficace pour des applications pour lesquelles la simulation logique s'avère inutilisable à cause d'évènements survenant en temps réel. Un FPGA peut servir comme circuit de prototypage bon marché dans la mesure où le temps de conception est tache critique.

3- *L'accélération de fonctions* : Un FPGA peut fournir une aide précieuse pour le développement de différentes fonctions sur un composantreconfigurable. Par exemple, afin de satisfaire au mieux les besoins de communication entre plusieurs processeurs parallèles, les composants sont configurés selon l'interface à utiliser

3-Architecture des circuits FPGA :

Il existe plusieurs technologies et principes organisationnels qui sont utilisés pour la conception d'un circuit FPGA ; Cela dépend du fabricant de circuits FPGA.

3.1-Les principaux éléments d'une architecture FPGA :

Les éléments constitutifs d'un FPGA sont toujours à peu près les mêmes. Chaque fabricant ayant ses variantes par rapport à un autre. Nous pouvons citer un certain nombre de ces éléments: **[47]**

1- Les éléments ou blocs logiques : Ces blocs représentent les éléments de base de tout FPGA. Grâce à leur configuration on peut réaliser dans ces blocs toutes les opérations de logique combinatoire (dans la limite d'un nombre d'entrées). Ces blocs ont souvent la même constitution et cela malgré la différence de fabricants et d'architectures. Ils sont généralement constitués d'une ou plusieurs LUTs(LookUp Table) qui contiennent, après configuration, la table de vérité de la fonction logique qu'elles doivent réaliser. La taille des LUTs est généralement de 4 (c'est-à-dire qu'elles disposent de 4 entrées). Les LUTs sont généralement suivies d'un registre de sortie, ce qui permet de synchroniser, si nécessaire, la sortie sur une horloge. La plupart des blocs logiques de base sont munis d'une chaîne de propagation rapide de retenue, afin de former de petits additionneurs rapides.

2- Les éléments de mémorisation : La nécessité d'intégrer des blocs de mémoires directement dans l'architecture des FPGAs est vite devenue cruciale pour des applications qui demandent souvent des capacités de stockage importantes (comme des applications de traitement d'images). De cette façon, les temps d'accès à la mémoire sont diminués puisqu'il n'est plus nécessaire de communiquer avec des éléments extérieurs au circuit. Du fait de l'intégration de ces blocs dans des architectures déjà existantes, un point important est le routage entre les parties mémoires et les ressources logiques. La caractéristique importante de ce routage est sa flexibilité. Si ce routage n'est pas assez flexible alors le circuit est difficilement routable, si il est trop flexible, il y a surconsommation de surface (donc de silicium).

3- Les éléments de routage: Les ressources de routage représentent la plus grosse partie de silicium consommée sur la puce réalisant le circuit. Ces ressources sont composées de segments (de longueurs différentes) qui permettent de relier entre eux les autres éléments via des matrices de connexions. Le routage de ces ressources est un point critique du développement d'une application sur un FPGA. Si ces éléments sont importants, c'est parce qu'ils vont déterminer la vitesse et la densité logique du système. Par exemple, les matrices de routage (programmable switch) sont physiquement réalisées à l'aide de transistors de cellules SRAMs, qui ont une résistance et une capacité, ce qui entraîne l'existence de constantes de temps.

4- Les éléments d'entrées sorties : Les éléments d'entrées/sorties peuvent bénéficier de protections, de buffer ou d'autres éléments permettant la gestion des entrées et des sorties.
En particulier il est à noter que les circuits actuels proposent différentes normes pour les niveaux d'entrées et de sorties (par exemple: LVTTL 2 - 54mA, PCI 5V, PCI 3.3V, HSTL…), qui par configuration, peuvent êtres choisies afin de s'adapter à l'environnement.

5- Les éléments de contrôle et d'acheminement des horloges : Les FPGAs sont prévus pour recevoir une ou plusieurs horloges. Des entrées peuvent être spécialement réservées à ce type de signaux, ainsi que des ressources de routagesspécialement adaptées au transport d'horloges sur de longues distances (bufferisation des lignes). Aussi, pour assurer d'avoir la même horloge dans tout le circuit (synchronisation des signaux) les circuit sont reçus des éléments d'asservissement des horloges (des PLLs ou des DLLs) qui permettent souvent decréer à partir d'une horloge d'autres horloges à des fréquences multiples de la fréquence de l'horloge incidente. Ces arbres d'horloge permettent donc de disposer sur le circuit de fonctions travaillant à des fréquences différentes.

3.2-Exemple: Les circuits de type Xilinx.[40]

Afin de mieux percevoir tous les éléments précités, nous allons prendre comme exemple la famille de circuit XILINX. L'architecture, retenue par XILINX, se présente sous forme de deux couches:

- Une couche appelée *circuit configurable*,
- Une couche réseau *mémoire SRAM*(Static Random Access Memory).

La couche dite *'circuit configurable'* est constituée d'une matrice de blocs logiques configurables (CLB) permettant de réaliser des fonctions combinatoires et des fonctions séquentielles. Tout autour de ces blocs logiques configurables, nous trouvons des blocs entrées/sorties (IOB) dont le rôle est de gérer les entrées-sorties réalisant l'interface avec les modules extérieurs (voirfigure5.1). La programmation du circuit FPGA consistera, par le biais de l'application d'un potentiel adéquat sur la grille de certains transistors à effet de champ, à interconnecter les éléments des CLB et des IOB afin de réaliser les fonctions souhaitées et d'assurer la propagation des signaux. Ces potentiels sont tout simplement mémorisés dans le réseau mémoire SRAM.

La configuration du circuit est mémorisée sur *la couche réseau SRAM* et stockée dans une ROM (Random Only Memory) externe. Un dispositif interne permet à chaque mise sous tension de charger la SRAM interne à partir de la ROM.

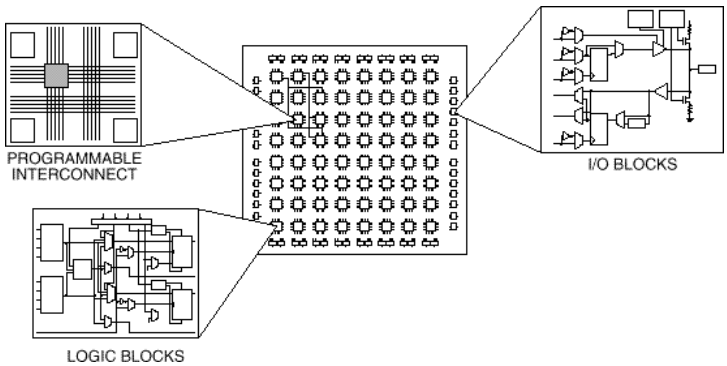


Fig 4.1 **Architecture générale d'un FPGA**

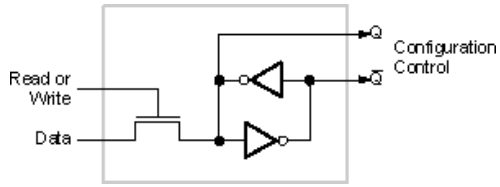


Fig 4.2 **Cellule SRAM**

Il y a donc deux types de cellules de base:

- Les cellules d'entrées/sorties appelées IOB(Input Output Bloc),
- les cellules logiques appelées CLB (Configurable Logic Bloc) Ces différentes cellules sont reliées entre elles par un réseau d'interconnexions configurable.

- *Les CLB (Configurable Logic Bloc) :* Chaque bloc logique configurable est composé d'un bloc de logique combinatoire composé de deux générateurs de fonctions à quatre entrées et d'un bloc de mémorisation synchronisationcomposé de deux bascules D. Quatre autres entrées permettent d'effectuer les connexions internes entre les différents éléments du CLB, comme le montre la figure5.2. Ainsi un CLB peut être utilisé pour réaliser:

- Deux fonctions indépendantes à quatre entrées indépendantes,
- Ou une seule fonction à cinq variables,
- Ou deux fonctions, une à quatre variables et une autre à cinq variables.

L'intégration de fonctions à plusieurs variables diminue le nombre de CLB nécessaires, les délais de propagation des signaux et par conséquent augmente la densité et la vitesse du circuit. Les sorties de ces blocs logiques peuvent être appliquées à des bascules au nombre de deux ou directement à la sortie du CLB (sorties X et Y).

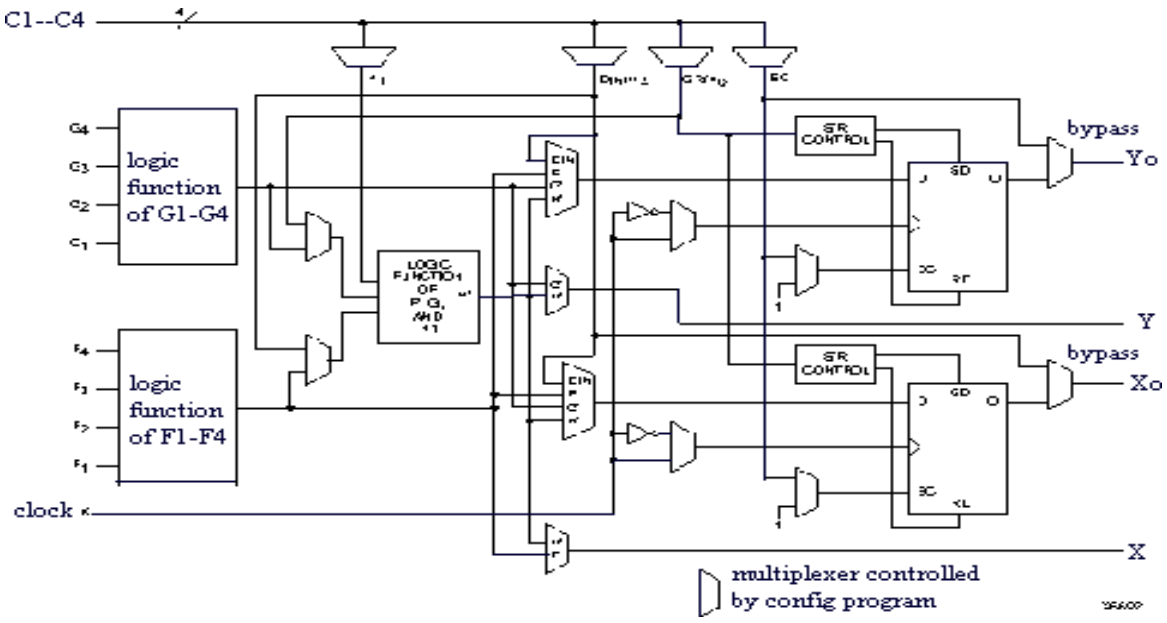


Fig 4.3 **Cellule logiques(CLB)**

-*Les IOB (Input Output Bloc)*: Les blocs entrée/sortie permettent l'interface entre les broches du composant FPGA et la logique interne développée à l'intérieur du composant. Ils sont présents sur toute la périphérie du circuit FPGA. Chaque bloc IOB contrôle une broche du composant et il peut être défini en entrée, en sortie, en signaux bidirectionnels ou être inutilisé (haute impédance).

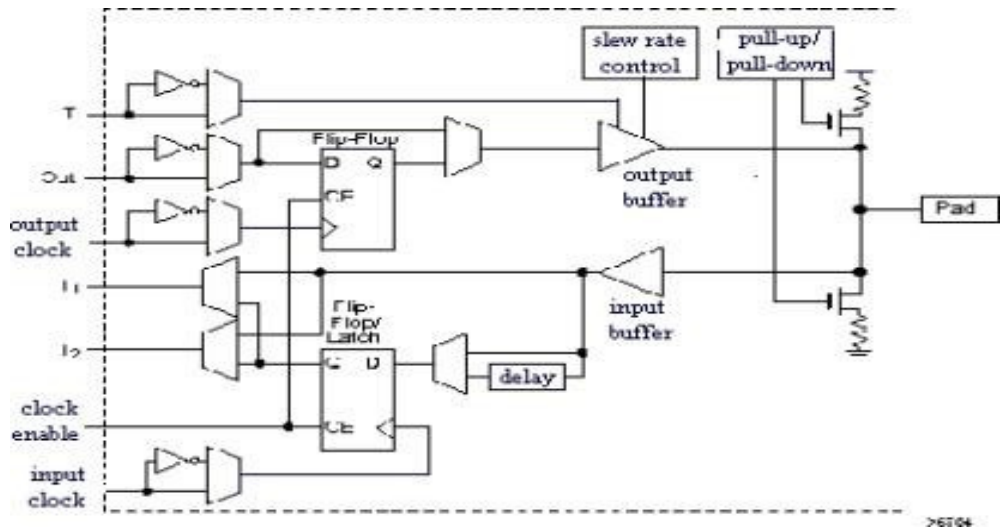


Fig 4.4 **Bloc IOB(Input Output Bloc)**

- *Les différents types d'interconnexions* **:** Les connexions internes dans les circuits FPGA sont composées de segments métallisés. Parallèlement à ces lignes, nous trouvons des matrices programmables réparties sur la totalité du circuit, horizontalement et verticalement entre les divers CLB. Elles permettent les connexions entre les diverses lignes, celles-ci sont assurées par des transistors MOS dont l'état est contrôlé par des cellules de mémoire vive ou RAM. Le rôle de ces interconnexions est de relier avec un maximum d'efficacité les blocs logiques et les entrées/sorties afin que le taux d'utilisation dans un circuit donné soit le plus élevé possible. Le constructeur XILINX propose trois sortes d'interconnexions selon la longueur et la destination des liaisons. Nous disposons

- d'interconnexions à usage général : Ce système fonctionne en une grille de cinq segments métalliques verticaux et quatre segments horizontaux positionnés entre les rangées et les colonnes de CLB et de l'IOB. Ces interconnexions sont utilisées pour relier un CLB à n'importe quel autre.

- d'interconnexions directes : Ces interconnexions permettent l'établissement de liaisons entre les CLB et les IOB avec un maximum d'efficacité en terme de vitesse et d'occupation du circuit. De plus, il est possible de connecter directement certaines entrées d'un CLB aux sorties d'un autre.

- de longues lignes : sont de longs segments métallisés parcourant toute la longueur et la largeur du composant, elles permettent éventuellement de transmettre avec un minimum de retard les signaux entre les différents éléments dans le but d'assurer un synchronisme aussi parfait que possible. De plus, ces longues lignes permettent d'éviter la multiplicité des points d'interconnexion.

Les performances des interconnexions dépendent du type de connexions utilisées. Pour les interconnexions à usage général, les délais générés dépendent du nombre de segments et de la quantité d'aiguilleurs employés. Le délai de propagation de signaux utilisant les connexions directes est minimum pour une connectique de bloc à bloc. Quant aux segments utilisés pour les longues lignes, ilspossèdent une faible résistance mais une capacité importante.

3.3-Techniques de programmation des FPGA:

Les circuits FPGA ne possèdent pas de programme résident. A chaque mise sous tension, il est nécessaire de les configurer. Leur configuration permet d'établir des interconnexions entre les CLB et IOB. Pour cela, ils disposent d'une RAM interne dans laquelle sera écrit le fichier de configuration. Le format des données du fichier de configuration est produit automatiquement par le logiciel de développement sous forme d'un ensemble de bits organisés en champs de données.

Le FPGA dispose de quatre modes de chargement (maître série, maître parallèle bas, maître parallèle haut, périphérique et esclave série) et de trois broches M0, M1, M2 lesquelles définissent les différents modes. Ces modes définissent les différentes méthodes pour envoyer le fichier de configuration vers le circuit FPGA, selon deux approches complémentaires :

- configuration automatique, le circuit FPGA est autonome,
- configuration externe, l'intervention d'un opérateur est nécessaire.

Dans le mode maître série, le FPGA est maître de sa configuration ; Ce mode utilise une mémoire PROM où le programme est préalablement chargé par le système de développement utilisé pour le circuit FPGA. Le FPGA génère tous les signaux de dialogue nécessaires pour la copie du contenu de la PROM dans sa RAM interne. On dispose aussi d'un mode maître parallèle où le FPGA est relié en parallèle à une EPROM classique, de même qu'un mode passif type périphérique

dans lequel le FPGA est considéré comme un périphérique de micro processeur et peut-être configuré à partir de celui-ci. Dans le mode esclave, le programme de configuration peut être envoyé à partir d'un PC, d'une station de travail ou à partir d'un autre circuit FPGA. C'est souvent le mode exploité pour la mise au point d'une configuration.

4- Le FPGA VIRTEX[41]

La famille VIRTEX du constructeur XILINX, offre de grandes performances ; Constituée de 08 modèles, ceux-ci fournissent une haute densité d'intégration se situant entre 50 000 et 5 millions de portes logiques (tableau 5.1). Ils possèdent des interfaces rapides pour la connexion à des mémoires externes.

Device	# gate	CLB Array	Logic Cells	Block RAM bits	SelectRAM bits	Max I/C
XCV50	58K	16 x 24	1728	32768	24576	180
XCV100	109K	20 x 30	2700	40960	38400	180
XCV150	164K	24 x 36	3888	49152	55296	260
XCV200	236K	28 x 42	5292	57344	75264	284
XCV300	323K	32 x 48	6912	65536	98304	316
XCV400	468K	40 x 60	10800	81920	153600	404
XCV600	661K	48 x 72	15552	98304	221184	512
XCV800	888K	56 x 84	21168	114688	301056	512
XCV1000	1.1M	64 x 96	27648	131072	393216	512

Tableau4.1 : **La famille VIRTEX**

Les fonctionnalités offertes par ces composants sont les suivantes:

- Plus de 50.000 portes logiques(entre 50 K et 5 millions)
- Nombre de cellules IOB supérieur à 786
- Supporte 26 standards d'entrée/sortie: LVDS, HSTL, PCI…
- Disponibilité de mémoire vive sous forme de Block RAM (RAM distribuée)

- Possède des multiplieurs 18x18 bits incorporés.
- 08 lignes principales d'horloges…..

4.1-Architecture générale:

L'architecture du VIRTEX est constituée d'une structure régulière et flexible de cellules CLB programmables, entourée par un périmètre de cellules IOB (figure5.4). Il y a quatre DLLs (Delay-Locked Loops) à chacune des extrémités de la matrice. Selon le modèle, il peut y avoir jusqu'à quatre colonnes de block RAMs d'une capacité de 18 Kbit chacune. Tous ces éléments sont interconnectés à l'aide d'une puissante hiérarchie de segments métalliques(routage).

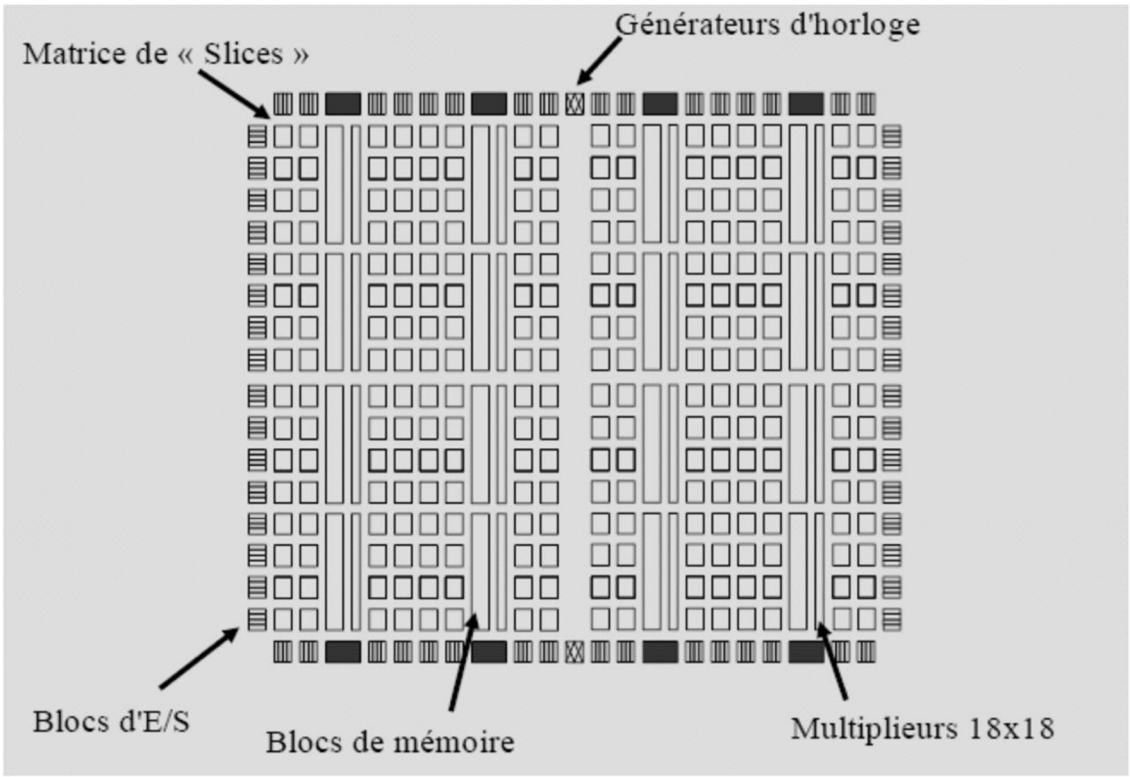


Fig 4.5 **Architecture générale du VIRTEX[39]**

4.2- Les cellules CLB(Configurable logic blocs):

La cellule de base du bloc CLB est la cellule logique. Chaque cellule logique (slice en anglais) est constituée d'un générateur de fonctions à 04 entrées, de logique

Combinatoire et d'éléments de stockage. Chaque bloc CLB contient 04 slices réparties en deux couches identiques (figure 5.5). Chacune des slices possède deux générateurs des fonctions logiques, deux éléments de stockage, de larges multiplexeurs et de la logique permettant de combiner des générateurs de fonctions afin de réaliser des fonctions à 05 ou 06 entrées. Les générateurs de fonctions ont implémenté par des tables LUTs à 04 entrées.

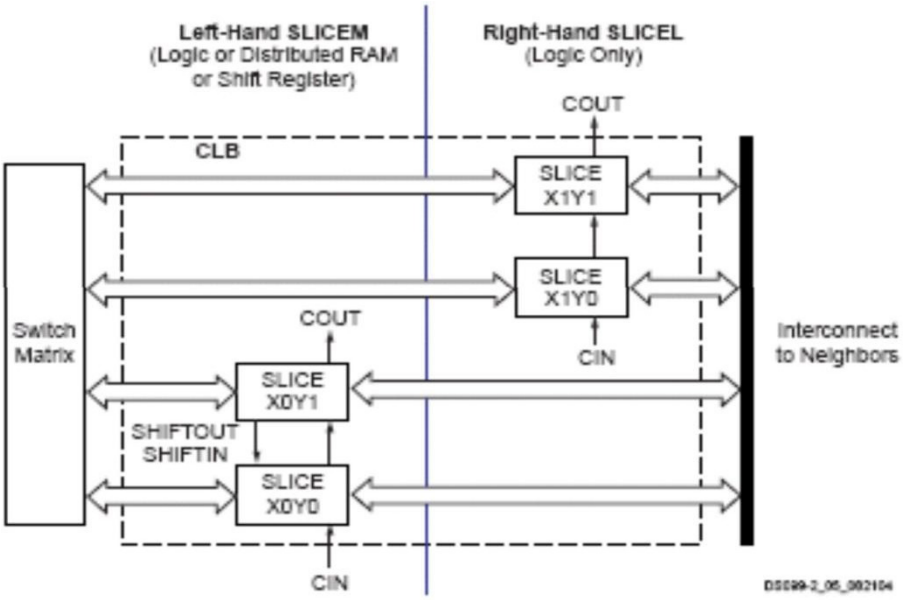


Fig 4.6 **Arrangement des couches à l'intérieur du bloc CLB du VIRTEX[45]**

4.3-Les cellules IOB (Input output blocs):

Les cellules IOB du VIRTEX possèdent des broches d'entrée/sortie très rapides qui supportent un grand nombre de standard d'Entrée/Sortie. Chaque cellule IOB est constituée de trois registres (figure5.6) qui peuvent fonctionner suivant différents modes : comme des bascules D ou comme des bascules T ou bien comme des latchs(verrou). Chaque cellule IOB possède comme signaux de contrôle un signal d'horloge (CLK) que se partage les trois registres et un signal Set/Reset(SR) qui peut être configuré de différentes manières. Il existe égalementun signal d'horloge d'initialisation (Clock Enable) pour chacun des registres.

4.4-Les block RAM:

Plusieurs mémoires de grande taille appelées Block RAM sont incorporées au VIRTEX . Ces blocs sont organisés en colonnes et leur taille est égale à quatre fois celle d'un bloc CLB. Un block RAM (figure 5.7) est une RAM à port double de 18,432 bits avec des signaux de contrôle indépendants pour chaque port. Chacun des ports possède un ensemble dédié de signaux de données, de contrôle et d'horloge afin de permettre les lectures/écritures simultanées

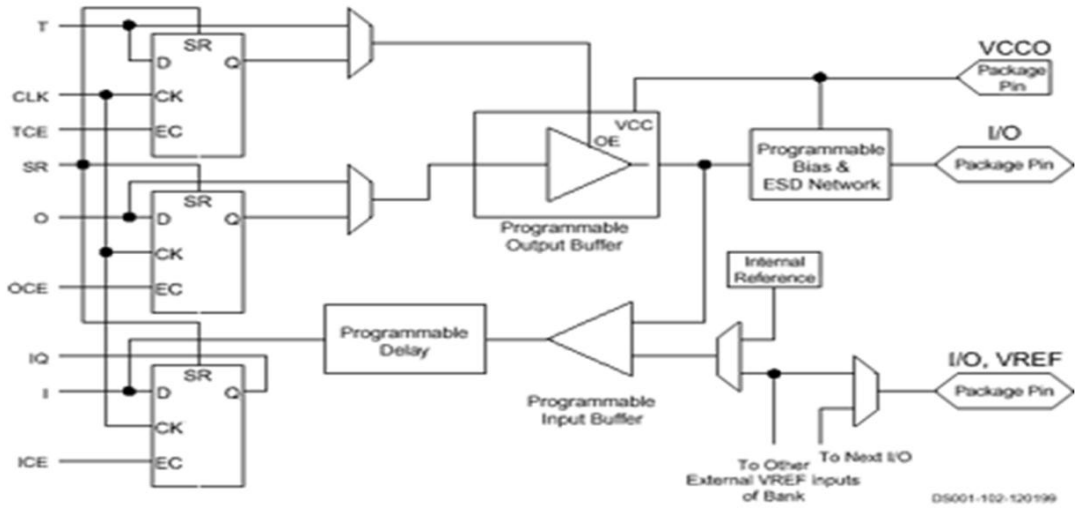


Fig 4.7 **Cellule IOB du VIRTEX**

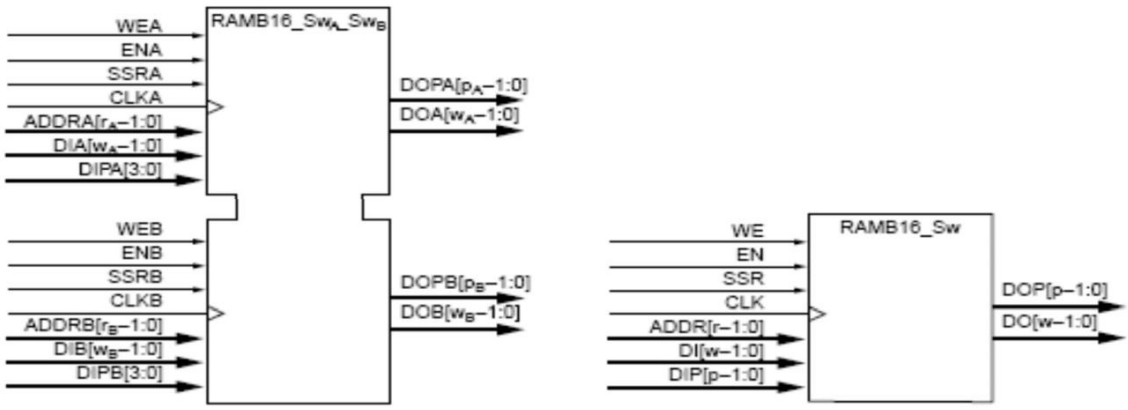


Fig 4.8 **Block RAM à double port et à port simple**

5- Etapes et environnements de développement des FPGA:

Le développement d'architectures à l'aide de FPGA nécessite l'emploi d'outils de développement. Un outil de développement comporte, en général, un logiciel de conception, une carte d'émulation pour la vérification du fonctionnement électrique de l'architecture conçue, et un micro-ordinateur hôte. Le logiciel sert à aider le concepteur à développer la méthode de conception de l'architecture.

La conception ou développement d'un circuit FPGA nécessite donc plusieurs outils et se décompose en plusieurs étapes qui sont les suivantes (figure5.8):

- La description comportementale du circuit
- La compilation et vérification des erreurs de schéma et de syntaxe
- La synthèse logique
- L'optimisation, la projection et le placement/routage
- La simulation temporelle,
- La génération de fichier de configuration.

À chaque étape du processus, des outils de synthèse et de simulation sont proposés par chaque constructeur.

5.1-Description comportementale du circuit:

Lors de cette étape, le concepteur utilise certains outils de description lui permettant de programmer le circuit à partir de la description de la fonction à réaliser. Ces outils ont pour finalité de représenter des structures logiques, analogiques ou des fonctions numériques.

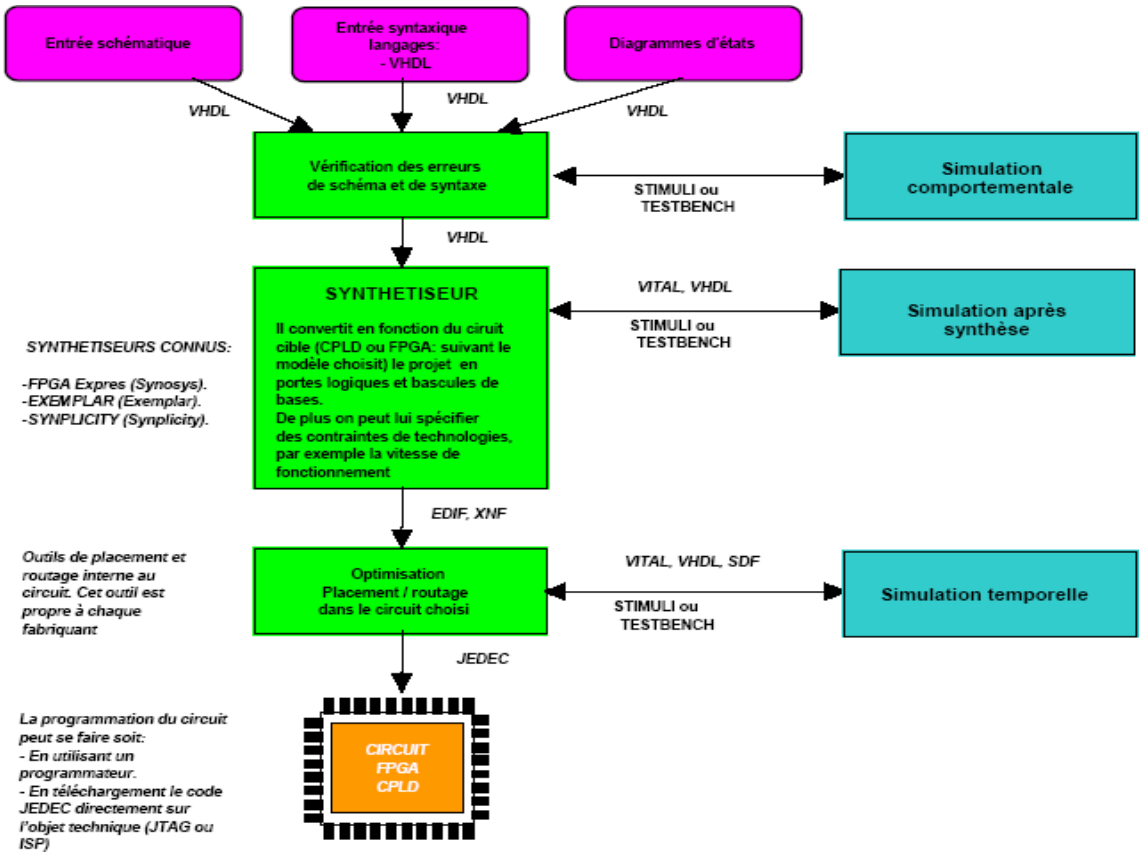


Fig 4.9 **Organisation fonctionnelle de développement d'un FPGA**

On peut utiliser des langages de description (ABEL, PALASM) ou des langages de description de haut niveau (VHDL,VERILOG); dans ce cas, la description est textuelle ou bien, on peut utiliser des diagrammes d'état ou des chronogrammes, nous aurons alors une description graphique. Généralement, on utilise le langage VHDL pour la description des circuits FPGA

Spécification en VHDL : Le VHDL est actuellement le langage de description de circuit le plus utilisé pour les PLD (Programmable Logic Device), CPLD (Complex Programmable Logic Device) et FPGA.

L'abréviation **VHDL** signifie **VHSIC H**ardware **D**escription **L**anguage(**VHSIC**:**V**ery **H**igh **S**peed **I**ntegrated **C**ircuit). Ce langage a été écrit dans les années 70 pour réaliser la simulation de circuits électroniques. On l'a ensuite étendu en lui

Rajoutant des extensions pour permettre la conception (synthèse) de circuits logiques programmables(PLD). **[50]**

Le VHDL est utilisé pour la modélisation et la synthèse de circuits numériques. La modélisation permet de simuler le comportement d'un circuit numérique, tandis que la synthèse permet de le programmer. Le langage VHDL est donc un langage de description de matériel qui permet de synthétiser des fonctions logiques complexes. À l'aide de ce langage, la première description définit la fonctionnalité du circuit en terme de blocs définis " haut niveau ". Progressivement, les blocs sont détaillés précisément jusqu'à une description proche des ressources matérielles. En effet, le langage VHDL autorise trois niveaux de description:

- Le niveau structurel: décrit le câblage des composants élémentaires. On décrit le modèle par sa structure sans faire intervenir le temps.
- le niveau comportemental : décrit le fonctionnement d'un modèle par des blocs programmes appelés *processus* qui échangent des données au moyen de signaux comprenant des instructions séquentielles.
- Le niveau flot de données: exprime les flots de données sortants du modèle en fonction des entrées sans se préoccuper de sa structure. **[47]**

La création du code source VHDL peut être faite au moyen d'un éditeur de texte ou d'outils graphiques permettant de décrire la structure du système à modéliser sous la forme de schémas ou de diagrammes de blocs et son comportement sousla forme de machines d'états, de chronogrammes ou de tables de vérité.

Synthèse du code VHDL : La synthèse permet à partir d'une spécification VHDL, la génération d'une architecture au niveau transfert de registre RTL(Register Transfert Level), c'est-à-dire que l'on effectue une description de niveau transfert de registre, ce qui permet l'ordonnancement et l'allocation de ressources sans une représentation physique, compilable par un outil de synthèse logique.

5.2-La compilation et la vérification des erreurs de schémas et de syntaxe :

La compilation va permettre dans un premier temps de vérifier la cohérence de la description et la syntaxe du langage utilisé, puis d'effectuer une simulation fonctionnelle dans un deuxième temps.**[46]**

Simulation fonctionnelle : Cette simulation porte sur les opérations numériques réalisées par le circuit; elle permet de vérifier la conception logique de ce dernier; Elle ne tient pas compte des capacités de liaison dues au routage entre les différentes cellules. Elle permet donc de vérifier uniquement la validité du circuit par rapport au cahier des charges d'un point de vue fonctionnel et non d'un point de vue temporel.

La spécification VHDL est directement émulée sur un support matériel tel qu'un circuit FPGA en précisant la famille utilisée pour une implantation physique du circuit. La compilation du code VHDL en code FPGA permet de générer le schéma correspondant ainsi qu'une netlist (XNF), qui est constituée d'une liste d'équations booléennes et d'informations portant sur les entrées / sorties du circuit.

Netlist : C'est un fichier ASCII (texte) contenant une description d'éléments de conception schématique et leurs interconnections. La Netlist est la voie la plus commune pour déplacer des conceptions d'un système à un autre. C'est la liste des connections entre les différents composants d'un schéma. La nomenclature est simplement la liste des composants.

Le *compilateur* ou analyseur vérifie la syntaxe d'une description VHDL. Il permet la détection d'erreurs locales, qui ne concernent que l'unité compilée. Il y a 02 approches : L'approche compilée produit directement du code machine, ou, dans certains cas, du code C qui sera lui-même compilé. L'approche interprétée

transforme le code source en un pseudo-code qui est interprété par le *simulateur*. Chaque concepteur possède une bibliothèque de travail (working library) dans laquelle sont placés tous les modèles compilés. L'*élaborateur* a pour tâche de créer un modèle exécutable à partir de modules compilés séparément et de détecter des erreurs globales.

5.3-Synthèse logique:

Lors de la phase de synthèse logique, chaque bloc sera matérialisé par des portes et/ou des bascules; Une structure logique, implantant la description fonctionnelle est produite. Le résultat est typiquement une représentation structurelle consistant en une liste de portes logiques interconnectées. La synthèse logique transforme donc la description comportementale en une description structurelle et produit un fichier d'interconnexion qui regroupe des instances de cellules logiques et précise leurs connexions

Nous pouvons citer, pour la famille XILINX, le logiciel *Design Compiler de* Synopsys, qui permet la synthèse logique grâce aux outils design_ analyzer ou *design_vision*(en mode graphique)et *dc_shell* (en mode ligne).

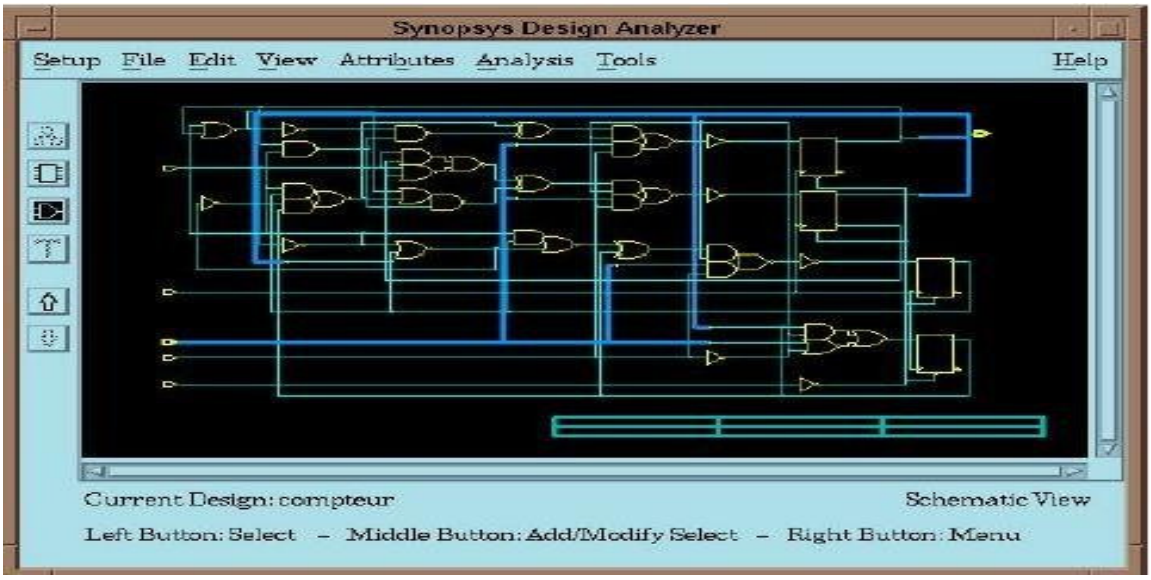


Fig 4.10 **Compilation et optimisation d'un modèle VHDL**

Une seconde phase de simulation fonctionnelle peut être envisagée après la phase de synthèse. Cette seconde phase est optionnelle. Elle permet essentiellement de valider le travail du synthétiseur et de mettre en évidence des problèmes éventuels.**[46]**

5.4-Optimisation, projection et placement/routage

La phase suivante consistera en l'implantation des portes et des bascules à l'intérieur du circuit logique. Cette tâche sera réalisée par le logiciel placement/routage (appelé fitter), au cours de laquelle les entrées et sorties seront affectées à des numéros de broches.**[50]**

Vient enfin la programmation du circuit et la vérification du fonctionnement.

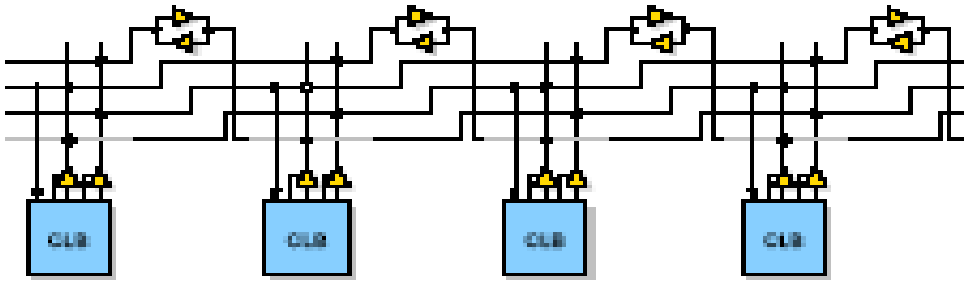


Fig 4.11 **Exemple de routage (Xilinx VIRTEX II E)**

Optimisation : Avant l'utilisation de la netlist, celle-ci est optimisée. Cette étape gère les problèmes de sortance des signaux par la duplication des fonctions logiques de sortance insuffisante, afin de multiplier les sortances. Les signaux inutilisés sont retirés, les expressions booléennes sont simplifiées, les signaux équivalents sont détectés.

Projection : La phase de projection dépend du circuit utilisé, les équations de la netlist sont transformées, regroupées en de nouvelles équations ayant un nombre

d'arguments inférieur ou égal au nombre de paramètres du bloc logique correspondant à la famille de circuit utilisée.

Placement / routage : L'étape suivante consiste à attribuer les cellules (CLB) du circuit à chaque équation délivrée par la projection et à définir les connexions. L'algorithme de placement place physiquement les différentes cellules et les chemins d'interconnexion dessinés entre les cellules afin de faciliter le routage. Des directives jointes à la netlist permettent une bonne répartition des cellules.

5.5-Simulation temporelle:

La simulation temporelle permet d'établir les caractéristiques temporelles du circuit après les opérations de placement et de routage. Il s'agit de vérifier la fonctionnalité du circuit en prenant en compte par un calcul estimatif les longueurs d'interconnexion et les retards apportés par les capacités parasites liées au partitionnement et au routage. La simulation temporelle vérifie que la fonctionnalité n'a pas été modifiée par l'introduction des délais de propagation et reste conforme au cahier des charges.

Le logiciel ModelSim SE permet la simulation temporelle au niveau RT (transfert de registre)ou au niveau porte, à partir des langages VHDL ou Verilog.

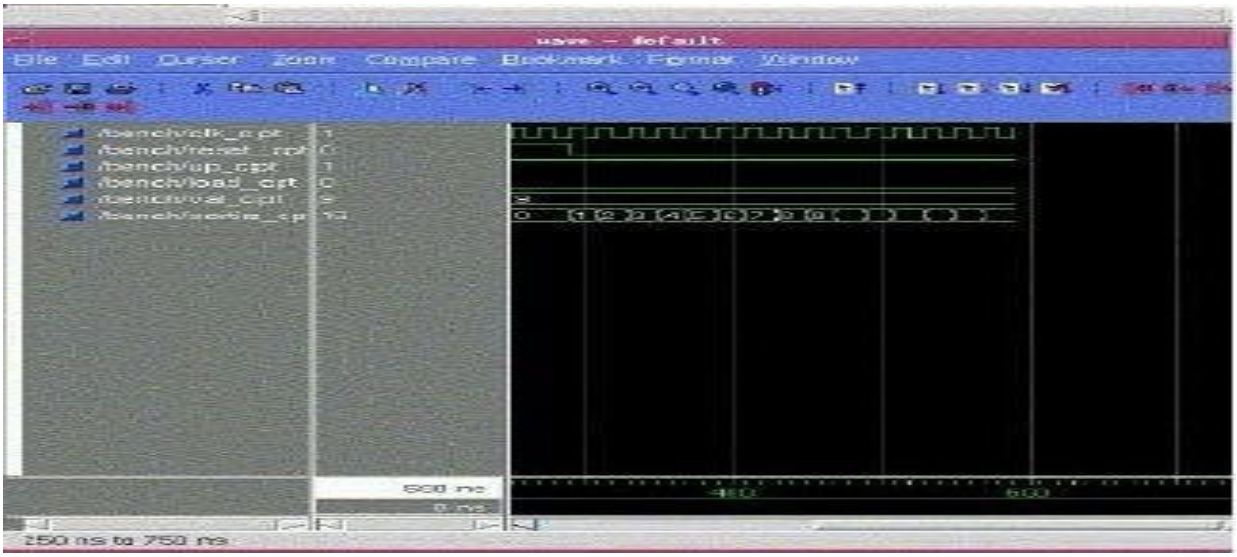


Fig 4.12 **Visualisation de l'état des signaux sous ModelSim**

5.6-Génération du fichier de configuration :

La dernière étape consiste à générer un fichier de configuration appelé Bits tream dans une PROM. Ce fichier contient les informations fournies au FPGA afin qu'il prenne la configuration souhaitée.

6- Conclusion:

Ce chapitre nous a permis de nous familiariser aux circuits FPGA et de présenter en détail le FPGA VIRTEX du constructeur Xilinx. Nous avons également explicité les étapes de développement d'un FPGA. L'implantation de notre architecture sur le vertex fait l'objet du prochain chapitre.

Chapitre V

Sommaire:

1. Introduction

L'implémentation de notre architecture sur un circuit FPGA consiste en la programmation du circuit à l'aide du langage de description de haut niveau VHDL. Nous allons donc décrire le circuit sous la forme d'algorithme VHDL, et ce, après avoir choisi les contraintes d'implémentation. La seconde étape consiste à effectuer une simulation fonctionnelle du circuit à l'aide du simulateur MODELSIM. La dernière étape englobe la synthèse logique et le placement sur FPGA type VIRTEX.

2. Méthodologie de conception

Notre but est d'élaborer une architecture efficace pour implémentation parallèle d'algorithme AES Ip-Core et qui puisse être implantée aisément sur un circuit FPGA il faut donc faire en sorte que le système à implanter ne soit pas trop gourmand en ressources c'est-à dire que l'on doit minimiser la quantité de mémoire nécessaire ainsi que le nombre d'unités arithmétiques utilisées. Le système doit être le plus modulaire possible, ou mieux, reprogrammable. L'architecture doit également être adaptée à un traitement rapide et ce, afin de permettre un débit de sortie élevé.

3. Objectif

Cette architecture doit être de complexité matérielle basse afin de minimiser le nombre de circuit nécessaire à son implantation sur FPGA (routage aisé, contrôle le moins complexe possible, nombre minimal de multiplieurs…)

1. Concevoir le système de telle manière à ce que la consommation totale d'énergie du circuit soit modérée.

2. Concevoir une architecture IP CORE de l'algorithme de AES implantable sur circuit numérique FPGA, qui puisse faire augmenter de manière significative la vitesse de traitement de notre système de cryptage, le rendement doit être relativement élevé.

3. Minimiser la surface du circuit qui est un paramètre important pour déterminer la performance du système.

4. Les contraintes pour la conception d'architecture implantable sur circuit FPGA *:*

4.1. Contrainte de surface : lors de la conception de notre système sur circuit FPGA la contrainte de surface est primordiale. Il faudra tenir compte du nombre des blocs logiques disponibles sur le circuit : plus le système sera complexe, plus le routage sera difficile à effectuer, et/ou le nombre de circuits nécessaires à son implantation augmentera. De plus, en minimisant la surface de la taille totale du système, on minimisera également la consommation totale du circuit.

4.2. Contrainte du temps : lorsque les applications envisagées visent le temps réel des données, il est souhaitable de concevoir le circuit de manière à ce que le traitement puisse effectuer tous les calculs nécessaires en temps réel. Ainsi, il peut être utile de rajouter des registres afin d'augmenter la fréquence de fonctionnement, si le retard introduit par ce registre n'influe pas de manière importante sur le résultat. Notre architecture introduit de la latence ; mais permettra une cadence de fonctionnement élevée. D'autre part, le temps d'accès aux registre /mémoires joue un rôle important dans la fréquence maximale que peut supporter le système. L'accès aux mémoires externes est beaucoup plus lourd que l'accès aux mémoires internes qu'il faut les implanter avec le circuit.

5. Méthodologie

La conception d'une architecture d'AES IP-CORE prend en compte les objectifs et les contraintes que nous fixons.

Dans le premier temps, nous effectuerons une implémentation totalement logicielle de cette architecture qui ne prendra en compte aucune contrainte matérielle. Le programme est réalisé dans le but de valider notre architecture globale de système.

Après une validation de notre système, nous définirons plus précisément les contraintes d'implémentation telles que la largeur du bus, les données en entrées et sorties…

Enfin nous passons à l'implantation matérielle proprement dite et qui comprend les étapes suivantes : la description comportementale du système à l'aide deVHDL et la synthèse logique.

6. La conception synchrone

Les circuits FPGA sont des circuits numériques basés sur la logique programmable : ce sont des systèmes séquentiels synchrones. La logique synchrone exige que toutes les entrées horloge des bascules, des registres, des compteurs soient attaquées par le même signal d'horloge.

Les différentes parties d'un système synchrone:

- La partie opérative : elle rassemble tous les composants et sous-ensembles nécessaires à l'exécution des fonctions(ou opérations) que doit effectuer le système synchrone : bascule, registres, bus….

- La partie commande : appelée aussi partie contrôle, elle rassemble tous les circuits nécessaires à la génération des signaux de commande de la partie opérative : bit de sélection d'un multiplexeur, signaux d'écriture ou de lecture de mémoires. L'objectif de la partie contrôle est donc de générer les signaux assurant l'enchainement correct du système logique. Celle-ci est une machine d'état.

7. Les machines d'états:

C'est une machine à état fini (finite state machine) c'est-à-dire ayant un nombre d'état fini et connu. Il est alors possible de déterminer un état quelconque à une seule condition de connaitre l'état initial de tout système logique. Il doit être placé dans un état connu à l'initialisation ce qui rend cette dernière indispensable.

Les principales caractéristiques sont les suivantes:

1. L'automate est toujours dans l'un des états possibles : l'état courant. Cet état est stocké dans une mémoire particulière, en général appelée "registre d'état".
2. L'état suivant peut être calculé à partir de l'état courant et/ou la valeur entrée. À chaque cycle d'horloge (automate est dit synchrone ; le registre d'état est mis à jour avec l'état préalablement calculé).

8. Introduction au parallélisme

8.1. Définition:

Les ordinateurs parallèles sont des machines qui comportent une architecture parallèle, constituées de plusieurs processeurs identiques, ou non, qui concourent au traitement d'une application.

La performance d'une architecture parallèle est la combinaison des performances de ses ressources et de leur agencement (Latence, débit).

8.2. Recours au parallélisme

Le parallélisme est la conséquence des:

- Besoins des applications
- Calculs scientifiques
- Traitements d'images
- Bases de données qui demandent des ressources en CPU et en temps de calcul de plus en plus importantes
- Limites des architectures séquentielles
- Performances
- Capacités d'accès à la mémoire
- Tolérances aux pannes

8.3. Architectures parallèles: elles offrent les avantages suivants:

=>Pas de limite de mémoire

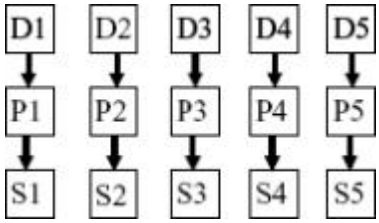
=>Pas de limite de processeurs

=> Accélération des calculs complexes ou coûteux en temps d'occupation CPU (calcul matriciel, simulation numérique, transformée de Fourier...)

=> Calcul répétitif sur un large ensemble de données structurées

=> Traitement indépendant

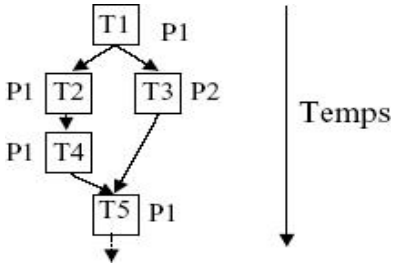
On distingue différents parallélisme dont notamment:



8.3.1. Le parallélisme de données : la même opération est effectuée par chaque processeur sur des données différentes.

8.3.2. Parallélisme de contrôle : des opérations sont réalisées simultanément sur plusieurs processeurs.

Le programme présente des séquences d'opérations indépendantes qui peuvent être exécutées en parallèle.



8.3.3. Parallélisme de flux : Les opérations sur un même flux de données peuvent être enchaînées (pipeline).

temps	Tâche 1	Tâche 2	Tâche 3	Tâche 4	Tâche 5
t1	Donnée 1				
t2	D2	D1			
t3	D3	D2	D1		
t4	D4	D3	D2	D1	
t5	D5	D4	D3	D2	D1
t6	D6	D5	D4	D3	D2

9. Présentation de l'architecture proposée

9.1 Mode d'opération:

C'est la manière de traiter les blocs de textes clairs et chiffrés au sein d'un algorithme de chiffrement par bloc.

Plusieurs modes existent, certains sont plus vulnérables que d'autres :

- Dictionnaire de codes(*Electronic Code Book*, *ECB*)
- Enchaînement des blocs(*Cipher Block Chaining*, *CBC*)
- Chiffrement à rétroaction(*Cipher Feed back*, *CFB*)
- Chiffrement à rétroaction de sortie(*OutputFeedback*,*OFB*)
- Chiffrement basé sur un compteur(*CounTeR*,*CTR*)

Le Modes de fonctionnement ECB (Electronic Code book) traite chaque bloc du texte en clair directement et indépendamment et crypte le même bloc de texte brut dans le même bloc de texte chiffré. Dans le cryptage et le décryptage ECB, plusieurs fonctions de chiffrement et fonctions de chiffrement inverse peuvent être calculées en parallèle. Le mode de chaînage de blocs de chiffrement (CBC), le mode de retour de sortie (OFB) et le mode de retour de chiffrement (CFB) offrent de meilleures propriétés de sécurité que l'ECB. Cependant, le chiffrement des blocs dépend des blocs chiffrés précédents, de telle sorte que le chiffrement ne peut pas être effectué en parallèle, par conséquent, la vitesse de CBC, OFB et CFB a des performances inférieures à celles de l'ECB. Le mode compteur (CTR) élimine le problème de sécurité de l'ECB et permet d'effectuer le cryptage et le décryptage en parallèle en utilisant uniquement la fonction de transfert de chiffrement. Comme le montre la figure 5.1[5]

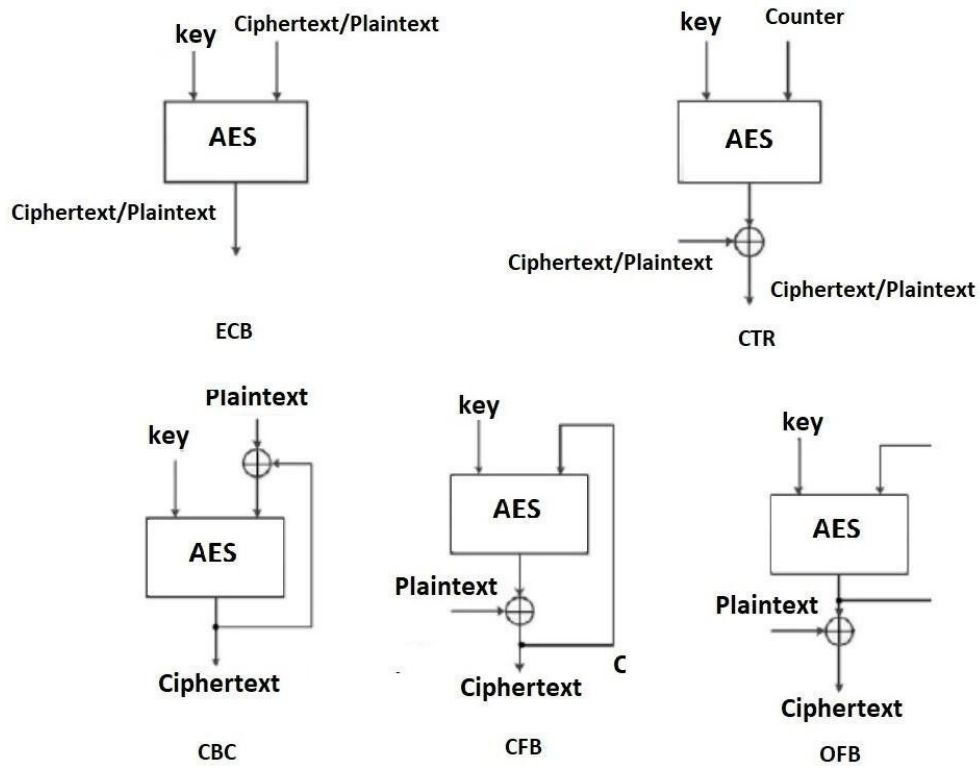


FIG5.1 Les déférents modes de Chiffrement et déchiffrement

9.2 Architecture de base:

L'idée générale dans cette architecture est d'ajouter des registressupplémentaires au milieu de la logique combinatoire, de telle sorte que plusieursblocs de données peuvent être traités par le chiffrement en même temps, et à exécuter les deux processus (génération de clé et génération du texte crypté (DATA))de façon parallèle.

9.3 Ip-core:

Les IP cores (intellectual Property) sont des blocs fonctionnels complexes préconçus et pré-vérifiés. Il existe deux types de IPcore:

- **Soft-Core**: Les Cores sont des modules architecturaux synthétisables. Ils offrent le plus haut degré de flexibilité de modification. D'autre part, beaucoup de problèmes de conception physique doivent être confrontés avant implémentation.

- **Hard-Core :** ils se réfèrent à des fonctions logiques préconfigurées qui peuvent être utilisées dans la conception. Ils offrent le plus haut degré de flexibilité de modification. D'autre part, beaucoup de problèmes de conception physique doivent être confrontés avant l'utilisation.

Xilinx offre un large choix de propriété intellectuelle optimisé pour les FPGA Xilinx. Le logiciel Xilinx CORE Générateur crée des versions paramétrables de la propriété intellectuelle prédéfinie optimisée pour les FPGA Xilinx.

9.4 Architecture d'AES IP (CFB)

Divers types d'architectures matérielles pour l'algorithme AES sont possibles. La meilleure architecture est celle qui offre le meilleur compromis entre vitesse d'horloge (débit de données) et occupation moindre des ressources matérielles. Dans cette conception chaque tour AES est d'avoir 4 stages(niveaux) et chaque stage contient les deux processus :transformation de données (data) et extension de la clé(Key)en pipeline, au sein de chaque cycle(pour 10 rondes de l'algorithme AES de 128 bits et les données clé de 128 bits).

L'utilisateur est invité à entrer le texte qui doit être chiffré et aussi la clé par laquelle il sera capable de déchiffrer ce texte crypté retour au texte brut. Après que l'élargissement de la clé est terminé, le processus de chiffrement commence. Dans le processus de cryptage, tout d'abord, un cycle près doit être effectué qui XOR l'original de 128 bits des données d'entrée aux 128 bits de la clé originale et le résultat intermédiaire est stocké dans la banque de registre. Une fois le cycle près est terminé, alors les épreuves restantes de fonctionnement sont effectuées. Ces cycles sont traités de la manière suivante:

Premièrement, les données (Data) de la banque de registre intermédiaire sont lues et envoyées à l'entrée de la S-box. Cela permettra de transformer les donnéesà leurs données correspondantes et transmettre au deuxième stage pour un traitement ultérieur et pour le processus(Key) juste les éléments de Wi-1 sont sélectionnés pour l'opération de Substitution(Subyte) à la sortie des données, on fait une addition(Xor)entre les éléments de Mot(Word)(wi-4) et les

Eléments d'opération de substitution et rotation de mot (Sub Rot Word)(wi-1), les résultats de cette opération sont stockés dans un registre pour le prochain stage. Dans le deuxième stage selon le principe de décalage(ShiftRow)les données(data) sont décalés et enregistrés pour le troisième niveau, pour le processus d'extension de la clé on effectue une addition entre le premier élément de la matrice prédéfinie (RCon) selon l'algorithme et dans cette étape on construit le premier vecteur de la clé de premier ronde Mot(Wi).

Et pour le troisième stage, l'opération de mixage des colonnes (MixCol) obtient les données et selon l'algorithme multiplie ces données avec une matrice standard pour produire une sortie des données stockées. Pour obtenir les autres vecteurs de la clé on effectue une addition(Xor) entre les éléments de registre.

Or dans le quatrième niveau les deux sorties sont XORed et sont stockées dans la banque intermédiaire registre pour le prochain tour.

Dans le dernier tour, l'opération de Mix colonne est ignorée et le résultat de données est stocké dans la mémoire de chiffrement pour l'affichage du textecrypté.

Le choix judicieux du module et le chemin de données pour une série particulière se fait par un automate à états finis.

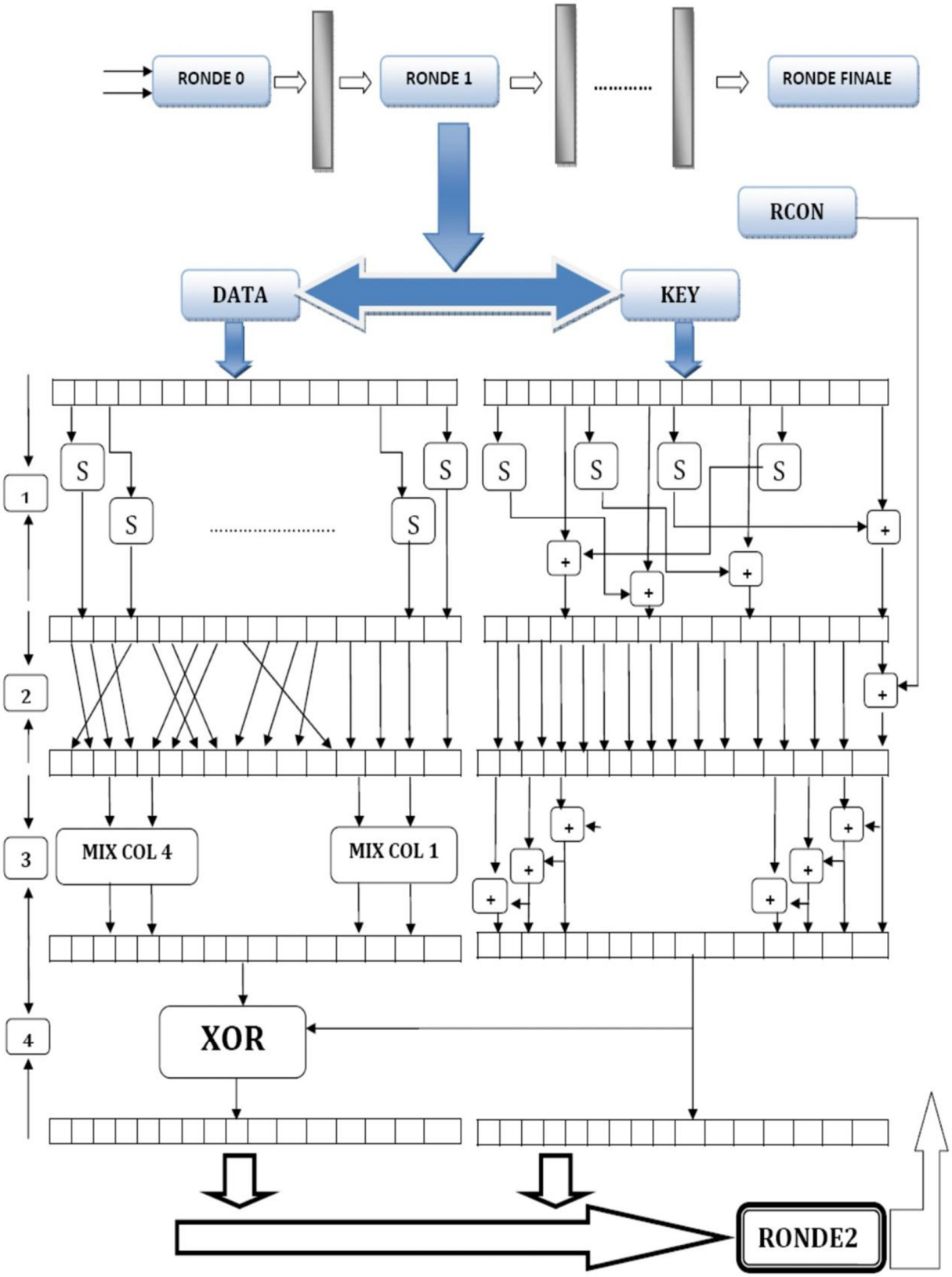


Fig 5.2 Architecture pipeline de l'algorithme de chiffrement AES

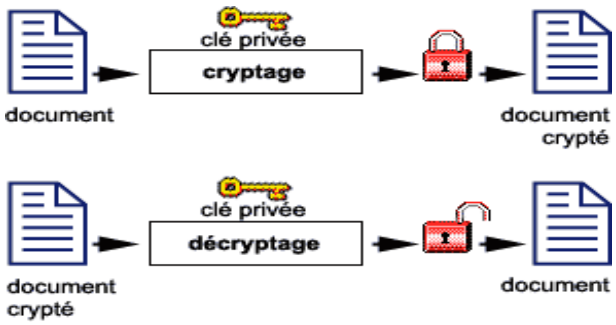
10. Implémentation d'architecture

Nous avons réalisé un logiciel à l'aide de Java, permettant de réaliserl'architecture d'AES. Ce logiciel est organisé comme suit:

1. Un programme principal pour simuler le fonctionnement de l'architecture
2. Deux sous-programmes indépendants:
   - Le premier sous-programme simule le cryptage
   - Le deuxième simule le décryptage.

La machine utilisée est un PC équipé d'un processeur Pentium 4; d'une vitesse de 2.2MHZ (fréquence d'horloge), doté d'une RAM de 128 MB.

10.1 Schéma général des tâches du logiciel:



10.2 Les résultats

L'exécution du logiciel nous a permis de constater que notre algorithme fonctionne correctement et qu'il est opérationnel pour valider le fonctionnement du système de manière très globale.

Voici le résultat pour le cryptage d'un texte

Texte1: «aucun homme n'a assez de mémoire pour réussir dans le mensonge»

Clé:1234567891234567

Textecrypté: «(ÉÛ[¤eßJ¯ã2aä¡×cùWYyV`ýä™-»

Texte 2 «cryptographie»

Clé: azertyuiopqsdfgh

Texte crypté: «[])•2®Ìu\Ñ{;¡Ö»

11. Vue globale de circuit

La synoptique détaillée de notre architecture est illustrée par la figure 5.6.
Le système fonctionne suivant une seule horloge appelée CLK. Une remise à zéro asynchrone globale de tous les éléments mémoire du système (registres, RAM…) est effectuée au démarrage à l'aide du signal asynchrone REST.
La famille de FPGA sélectionnée pour la mise en œuvre actuelle est Xilinx de type de la famille Vertex7. Il s'agit d'un dispositif de faible volume. Nous avons utilisé Vivado 2020.2 qui est un logiciel pour synthétiser un VHDL. Cet outil est également utilisé pour effectuer le comportement et la simulation de chronométrage.La conception de chiffrement est décrite en VHDL pour assurer la portabilité.
Cette mise en œuvre a l'avantage d'éliminer le besoin de rétroaction et le multiplexeur trouvé dans les implémentations classiques. Elle permet d'économiser 30 multiplexeurs dans le chemin critique.
L'architecture est optimisée qui permettra de réduire la vitesse que nous visons au profit des applications temps réel telles que la VoIP et la Vidéo-conférence. Cette approche occupe moins de 75% des ressources que l'architecture de base.

11.1 Les Modules

Notre conception se concentre principalement sur la mise en œuvre de l'architecture pipeline à chaque tour. L'architecture globale de chiffrement AES en pipeline comprend: Banques registres, Multiplexeurs, Matrice S-Box, Registres pipeline, Mix-Colonne Module, Module pour opération Addition, Contrôleur FSM.
L'unité de contrôle comme le montre la figure 5.7 est une machine à états finis pour gérer les rondes: une ronde initiale, neuf cas similaires intermédiaires et une ronde finale. L'unité de contrôle fournit et génère des signaux de contrôle pour les multiplexeurs pour chaque tour. Voici le pseudo code de processus de cryptage

11.1.1 La matrice S-BOX:

Sub-Bytes : il existe deux manières d'implémenter le cycle Sub-Bytes. La première consiste à construire la S-box en effectuant deux transformations. Il s'agit notamment de prendre un inverse multiplicatif dans le champ fini GF (28), et d'appliquer une transformation affine standard (sur GF(28)). La deuxième méthode consiste à stocker directement la S-box pre-calculée dans la ROM.

Les techniques de stockage mémoire possibles peuvent être la configuration des tranches FPGA en tant que RAM distribuée ou l'utilisation de BRAM embarquées [4] du dispositif. Lorsque S-box est implémentée en utilisant la LUT comme RAM, elle utilise 75% des ressources[5].

Notre tour de Sub-Bytes a été implémenté à l'aide des BRAM intégrées en configuration double port et en mode lecture seule. En utilisant les BRAM, nous avons évité tous les calculs et les délais supplémentaires nécessaires pour déterminer les valeurs de la S-box. L'état se compose de 16 octets, qui doivent être remplacés par la S-box et donc, 16 S-box sont nécessaires pour chaque tour. Une S-box a 256 entrées et ne prend donc que 2 Ko pour le stockage. Dans notre conception, deux S-box sont stockées par BRAM comme indiqué sur la figure 5.3. Ainsi, 8 BRAM sont nécessaires pour chaque tour.

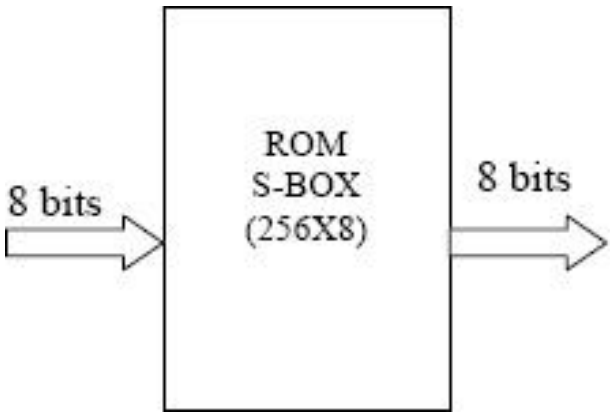


Fig 5.3 la matrice SBOX

11.1.2 Opération de décalage (Shift Row)

Il s'agit d'une étape de transposition sur la ligne de l'état où chaque rangée de l'état est décalée cycliquement par certain nombre d'étapes. La première ligne (ligne 0) est inchangée. La deuxième ligne(ligne1) est décalée d'un octet, la troisième rangée (ligne 3) est décalée de deux octets et la dernière rangée (ligne 4)est décalée de trois octets. Il garantit également que chaque octet de chaque ligne n'inter-réagit uniquement avec leurs octetscorrespondants. Pour exécuter le bloc SHIF TROWS il faut impliquer la modification de l'ordre des signaux et cela est contrôlé par l'automate.

Le résultat est stocké dans un registre pour la prochaine opération.

11.1.3. Mixage des colonnes(MIX-Colonne):

Pour l'implémentation architecture Mixage (Mix-colonne), il est préférable d'effectuer la multiplication dans des corps de Galois.

La transformation Mixcol peut être formulée comme des additions XOR en utilisant la propriété de calcul binaire.

En raison de pipeline, une colonne de l'état nouvelle matrice peut être obtenu dans un cycle d'horloge seulement.

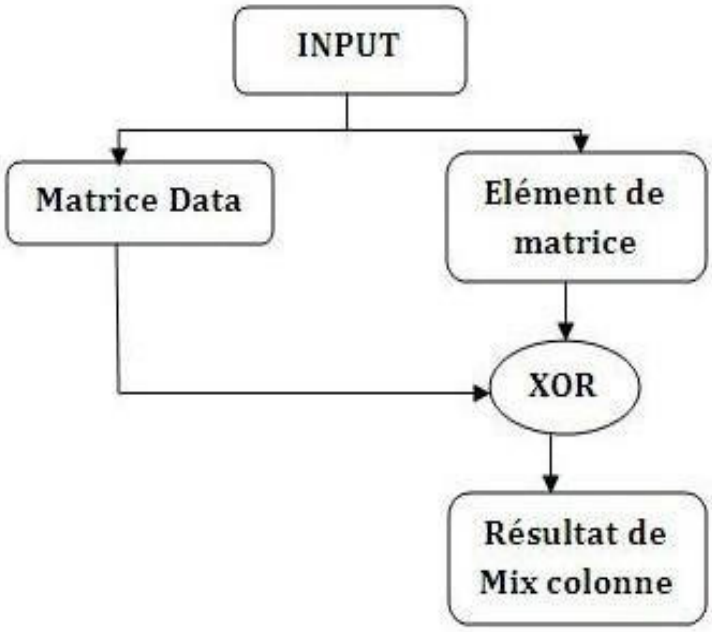


Fig 5.4 Schéma Mixage (MIXCOL)

11.1.4. Extension de la clé

L'expansion Key est une opération principale pour le chiffrage ou déchiffrage.

Le facteur de sécurité du chiffrement AES / déchiffrement dépend principalement sur cette partie, juste pour le premier tour la clé sera XOR avec le texte brut quant aux autres tours les clés seront calculées.

L'unité d'extension clé génère une Ronde clé(Key) chaque cycle d'horloge. Initialement Reset est affirmée et la clé d'origine est appliquée à l'entrée. En conséquence, les sorties compter0, ce qui active le signal de charge pour le registre clé RK (0) enregistré et de choisir l'adresse 0 de la ROM. Au premier frontpositif, la clé d'origine est stockée directement à la RK (0) et le registre de clé(Keyin) s'inscrire simultanément. Au premier bord négatif, le compteur est incrémenté à 1, ce qui active le signal de charge pour le RK (1) enregistré et de choisir l'adresse 1 de la ROM. En conséquence, la ROM sorties au premier tour constante matrice (Rcon) (1). Avant le second bord positif, la logique suivante calcule la clé de RK (1). Ce processus est répété 10 fois pour générer les 10 Ronde clés nécessaires. En conséquence, nous avons 11 Ronde clés (10 générées et l'original) d'une clé pour chaque tour.

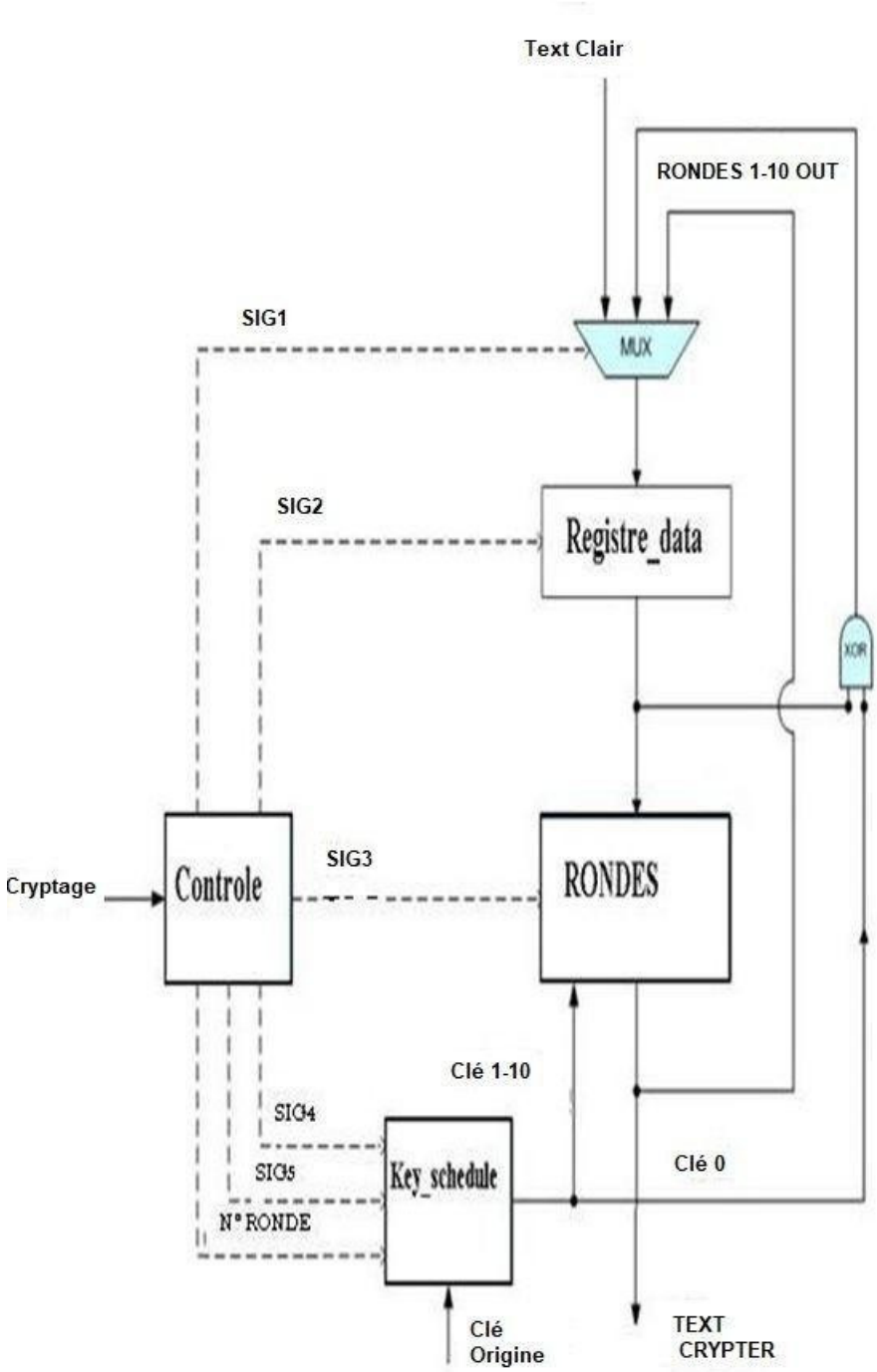


Fig 5.5 Circuit de Cryptage d'AES-IP CORE

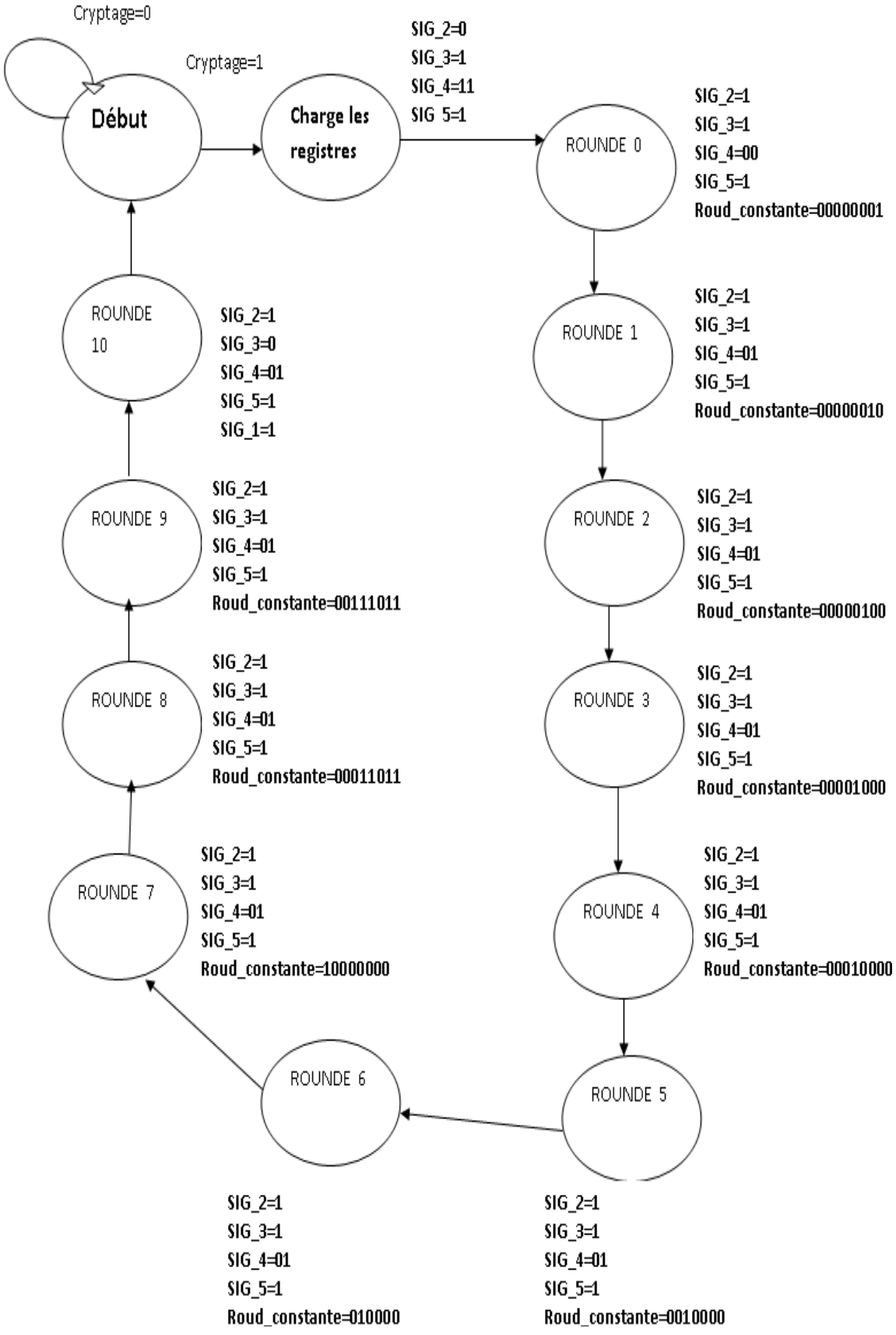


Fig 5.6 L'Unité de Contrôle

12. La simulation fonctionnelle

La simulation fonctionnelle permet de tester la validité de la conception. Aucune vérification temporelle n'est ici effectuée. Cette première phase de simulation a un double intérêt. Elle permet de valider fonctionnellement la description (syntaxique) en s'affranchissant des étapes parfois longue de placement de routage. Elle permet de valider(fonctionnellement) la conception de travail.

L'horloge : notre architecture est un circuit entièrement synchrone. Il possède une horloge unique permettant la synchronisation des données véhiculées à l'intérieur du circuit ainsi que le contrôle des données d'entrées de chaque élément mémoire.

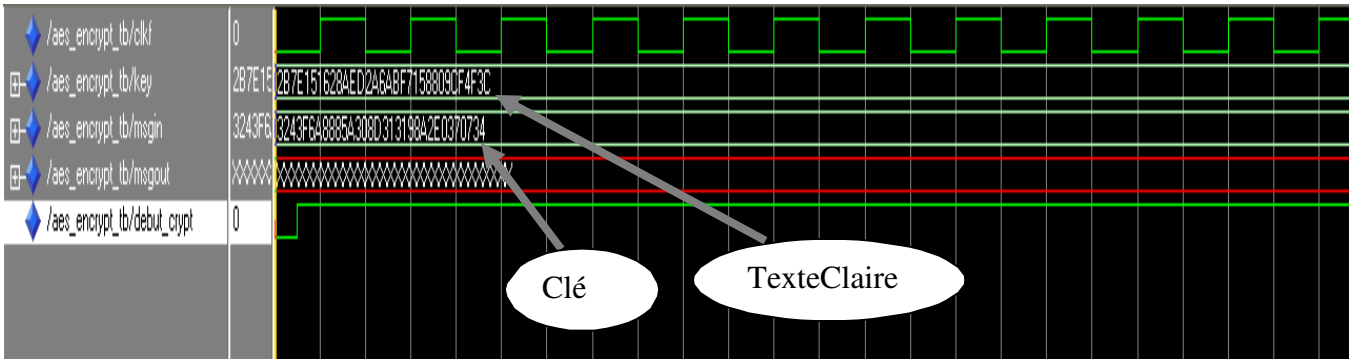


Fig 5.7 Début de chiffrement AES-IP CORE.

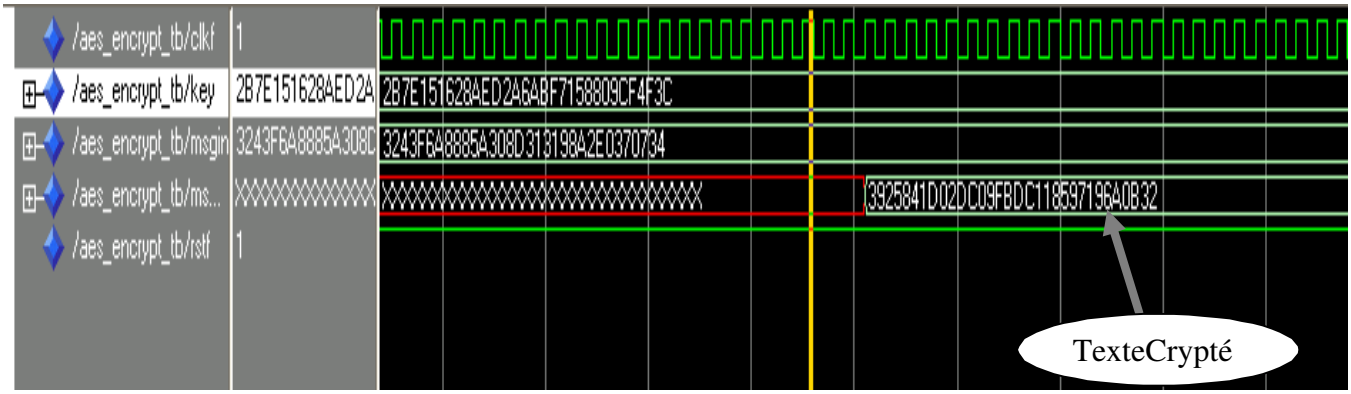


Fig 5.8 Résultat de chiffrement AES-IP CORE.

Le tableau 1 décrit les différents signaux d'entrée/sortie de chaque module(bloc)

Signal	**Description**
CLK	Horloge du système
Reset	Reset du système
Ack	Acquittement émis par l'initiateur
DataIN[0..31]	Bus de données sur 32 bits
Reg_outI[0..127]	Bus de données sur 128 bits
Data_Ready	Ecriture des trames de données
data_in[0..127]	Bus de données sur 128 bits
data_out[0..127]	Bus de données sur128 bits
Reg_in[0..127]	Bus de données sur 128 bits

Puisque les constantes utilisées lors de l'expansion de la clé ne changent pas, on les a générés avant même le début de processus du chiffrement ou déchiffrement. Ces constantes sont calculées pendant que reset est égale à un. Une fois ces constantes sont disponibles, le processus de chiffrement commence, cette étape dure deux tops d'horloge.

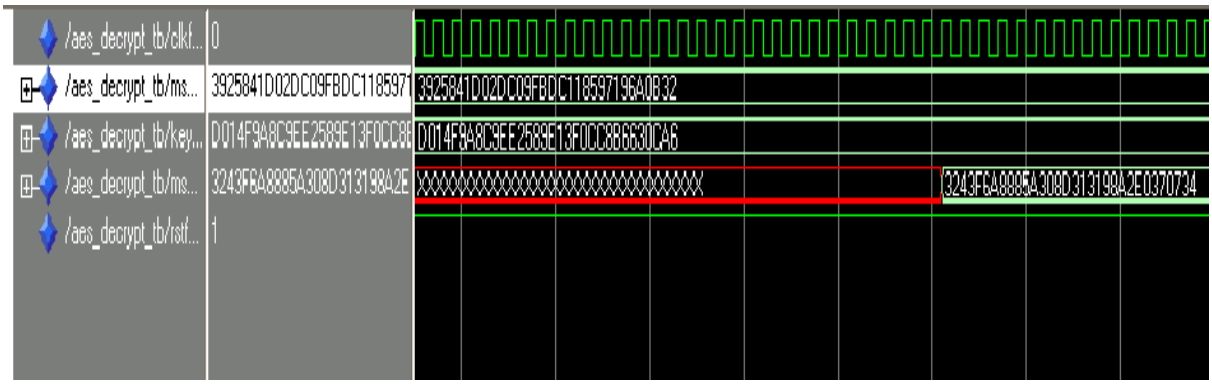


Fig 5.9.Résultat de déchiffrement AES-IP CORE.

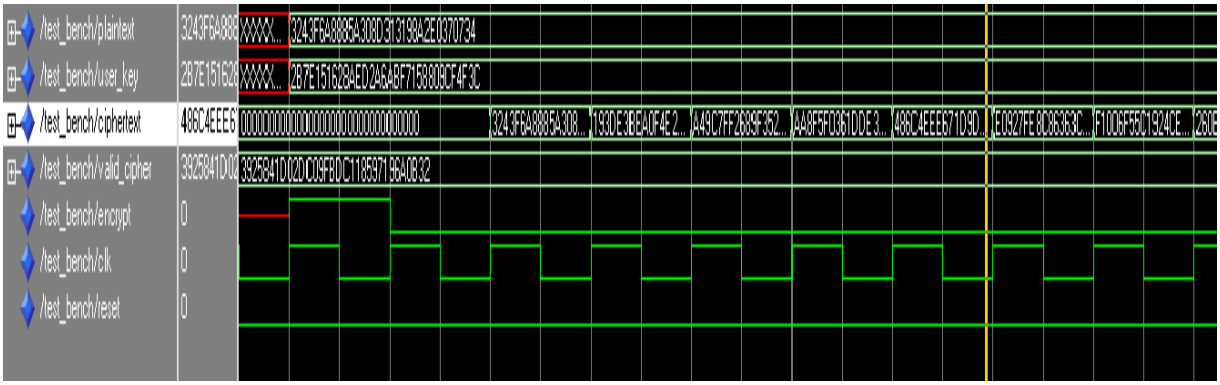


Fig 5.10. Les résultats de chiffrement AES pour chaque Ronde

13. Synthèse logique:

La synthèse logique est le processus qui transforme ou traduit une description dans un langage comportemental en une description structurelle donc chaque élément représente une ressource électronique de base prédéfinie. Le but des techniques de synthèse est bien de représenter les choix architecturaux en associant les éléments de base en fonction des contraintes exprimées (Temporelles, surface, consommation…).

A l'intérieur du cycle de conception, la synthèse logique occupe l'une des dernières étapes.

Elle suit la phase de spécification fonctionnelle. Elle précède la phase du placement routage et plus généralement la phase de production de la représentation physique.

Le synthétiseur que nous avons utilisé est XST provenant du Vivado 2020.2 d'ISE fondation du constructeur Xilinx. La cible d'implantation est le Virtex-7.

Virtex-7 La famille Virtex-7 a été introduite en juin 2010 sur une technologie de processus 28 nm, et est censée offrir une double amélioration des performances du système avec une puissance de 50% inférieure à celle des appareils Virtex-6 de la génération précédente. De plus, Virtex-7 double la bande passante de la mémoire par rapport aux FPGA Virtex de la génération précédente avec des performances d'interfaçage mémoire de1866Mbit/s et plus de deux millions de cellules logiques[51].

Dans la suite, on va décrire le comportement de chacun de ces modules

13.1 module interface AES IP

Ce module est responsable de regrouper des données de 32bits en des blocs de 128 bits à partir de la source(Texte Clair), et de les émettre à travers l'architecture de communication du système.

Cette interface est le premier bloc d'entrée de l'AES. Le fonctionnement de cette composante est simple. Pour chaque front montant d'horloge CLK et lorsque le signal Start est activé à '1', il y a une lecture d'une trame de données de 32 bits. Après 4 cycles d'horloge, le regroupement de données en un bloc de 128 bits est achevé, et le signal CLK est activé à '1' pour un seul cycle. Dans le cas où le signal Start est désactivé, on passe à l'état attente, et les données rassemblées seront transmises vers une file d'attente. Si le signal Start est toujours activé à '1', on reste dans l'état acquisition des données. Le signal d'acquittement Ack, provenantde l'initiateur, permet d'informer l'interface d'entrée sur la dernière trame du bloc de données. Si la taille du bloc de message (Texte en Clair ou Texte Crypté) est inférieure à 128 bits, on complète le reste du bloc par des zéros.

13.3 Fonctionnement du Controller:

Toutes les opérations de cryptage et décryptage sont surveillées par le contrôleur de l'AES, appelé "Controller". Le fonctionnement interne de contrôleur est géré un processus représenté par une machine d'état fini FSM(Finite State Machine).

Nous avons pris un schéma de blocs qui présente la totalité du circuit (voir lafigure5.11).

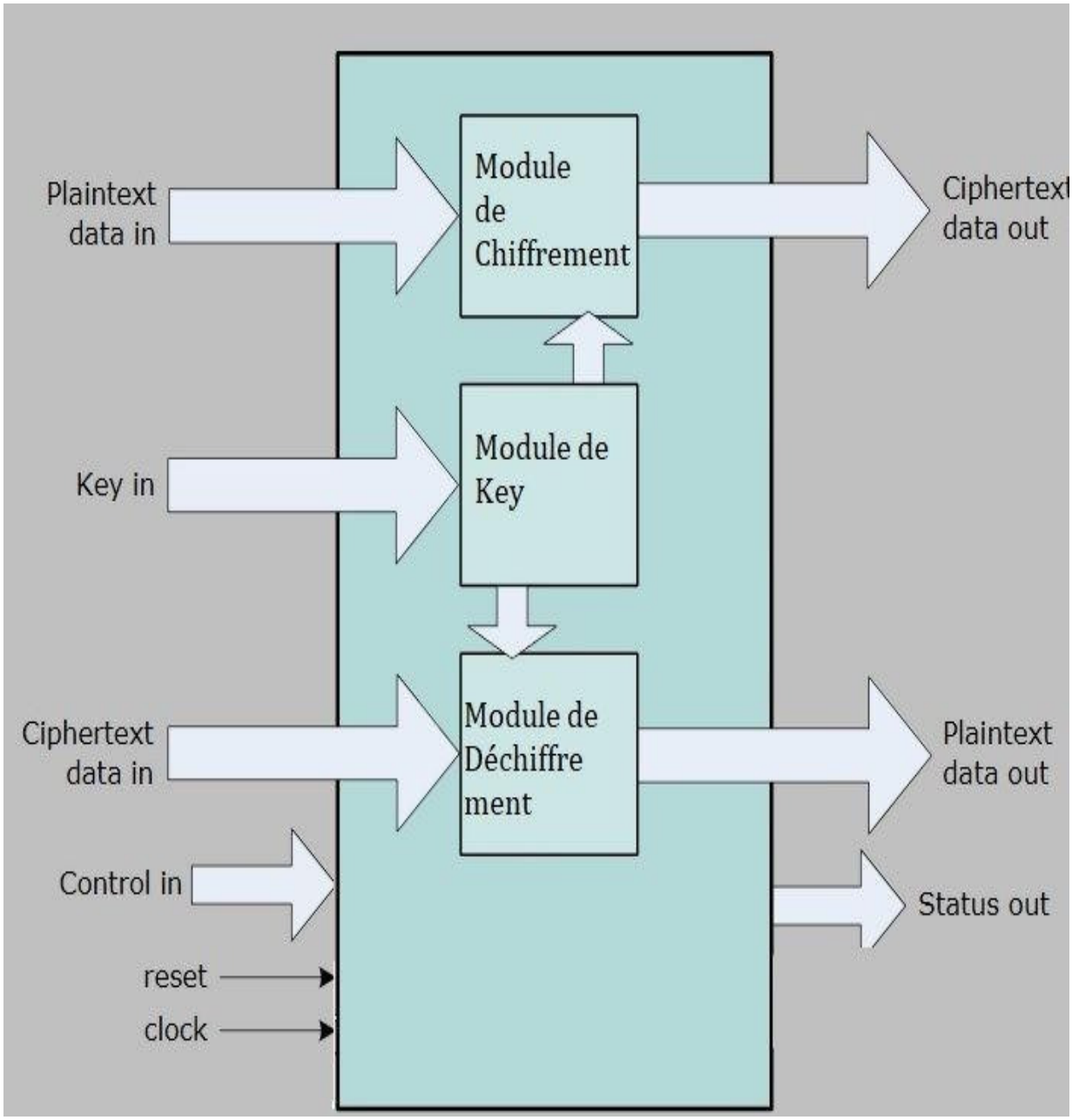


Fig 5.11 Les Modules de circuit AES-IP CORE

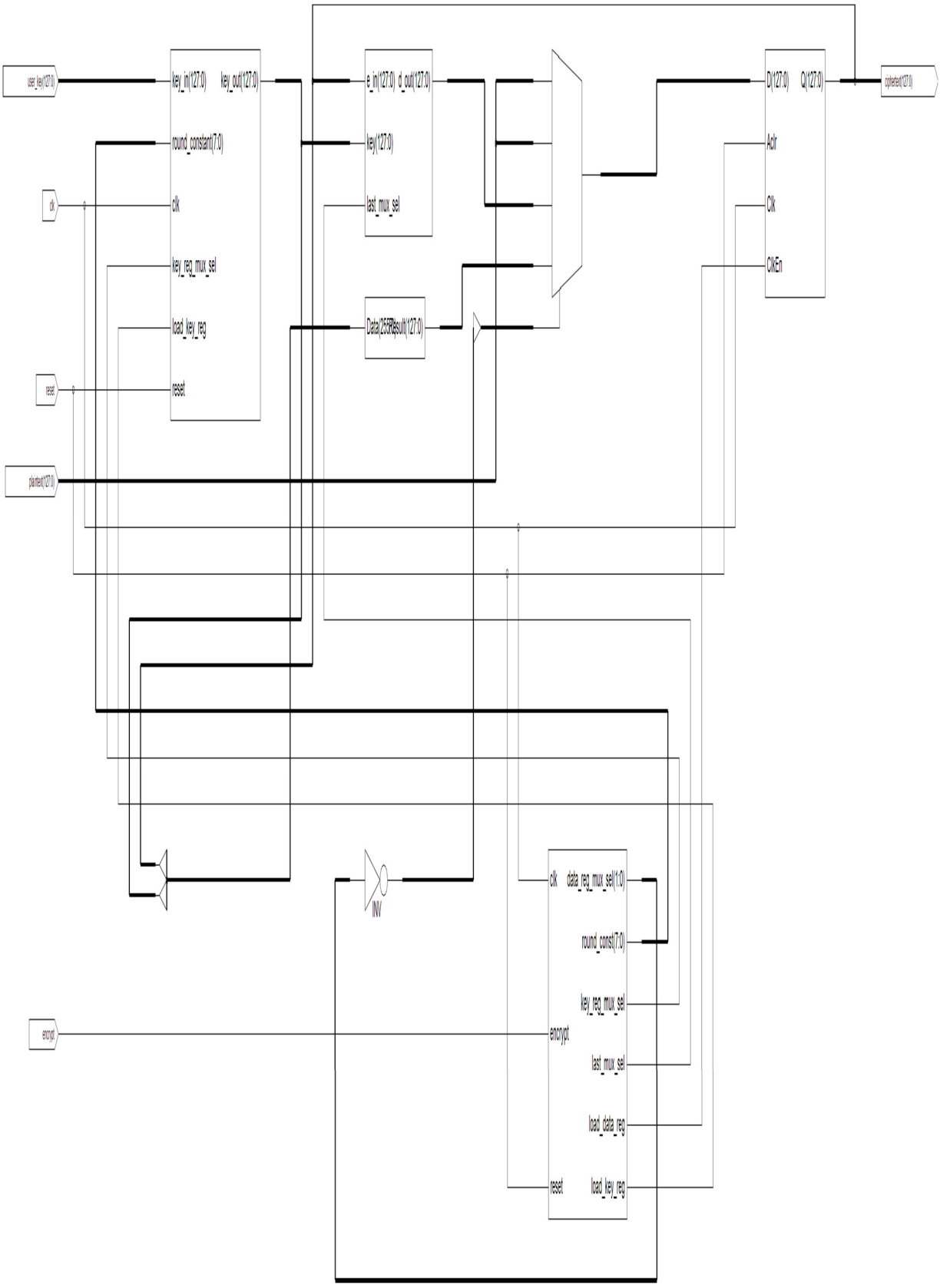


Fig 5.12 Schéma De Circuit AES-IP CORE

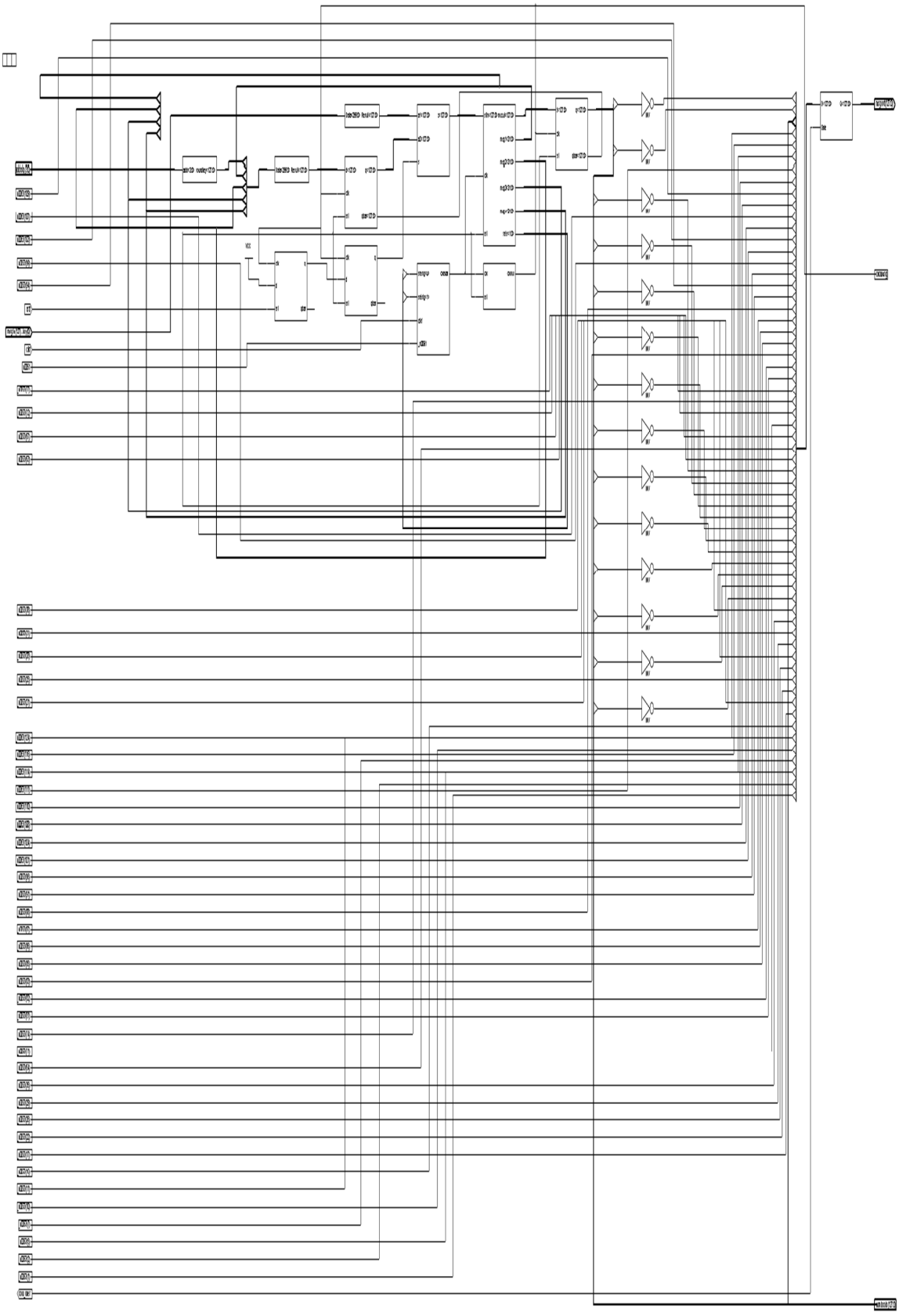


Fig 5.13 Schéma De Circuit détaillé AES-IP CORE

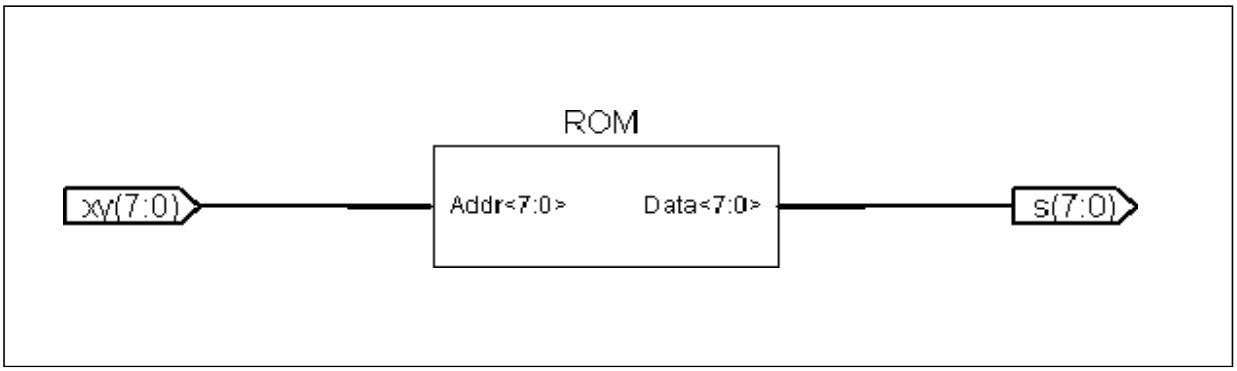


Fig 5.14 Schéma De SBOX

14. Les résultats

Le langage de programmation utilisé est le VHDL. Le résultat des analyses temporelles et les rapports d'implémentation sont générés par Vivado 2020.2 exploité aussi le MODELSIM 10.3 de XILINX. Le circuit utilisé est de type Vitrex-7 Les résultats de la mise en œuvre de l'architecture AES sont présentés au tableau suivant:

Type de FPGA	Période (ns)	Fréquence Maximale (MHZ)	Nbr IBOS	Nbr SLICES	Nbr LUT	Le débit(Gbps)
Virtex2	3.354	298.160	387(90%)	630(12%)	1280(12%)	3.81
Virtex4	2.303	434.826	387(82%)	444(2%)	1200(5%)	50.55
Virtex5	1.538	688.281	387(68%)	401(1%)	637(2%)	88.1

Voici un tableau qui résume les déférents délais et la fréquence pour chaque module avec le kit de type Virtex 7.

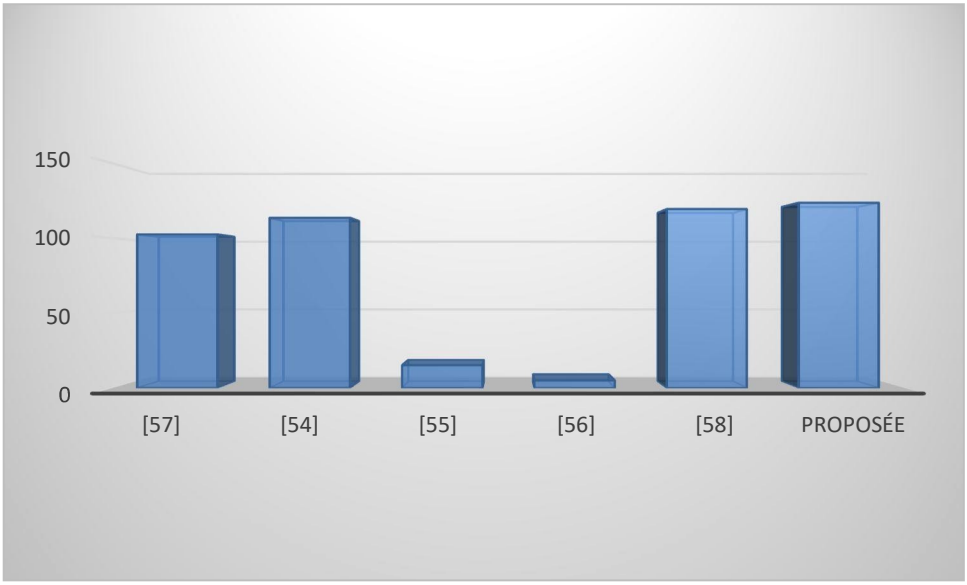
Module	Délais(ns)	Fréquence(Mhz)
Subyte	10.98	72.52
ShiftRow	7.2	142.2
Mix-column	8.22	122.481
Key-Expension	99.47	81.64

Nous avons constaté, à partir de l'analyse de ce tableau que la fréquence maximale de fonctionnement du système est assez élevée pour chaque type de FPGA.
Voici un tableau qui présente les occupations des ressources matérielles pour chaque opération d'algorithme AES Core.

Ressources	Utilisation	Disponibles	Utilisations%
LUT	500	303600	0.16
FF	402	607200	0.07
BRAM	5	1030	0.49
IO	388	600	64.67

Nous avons modélisé cette implémentation en utilisant VHDL et puis les mises en œuvre sur un kit de FPGA Virtex 7 et Vivado 2020.2 est utilisé pour la synthèse.
Ces résultats montrent que nous avons utilisé moins de ressources FPGA 1%slices et aussi 0.16% LUT. Cela conduit à une plus grande fréquence d'horloge (956MHz) et un débit de 123.52 Gbps.
La comparaison des performances entre les architectures qui ont déjà étés implantés et l'architecture proposée est résumée dans le tableau suivant :

Auteurs	Type FPGA	NBR SLICES	Fréquence Maximale MHZ	Le débit (GBP/S)
[57]	Virtex4	5177	200.00	102.4
[54]	Virtex4	1672	886.64	113.5
[55]	Virtex-7	2703	119	15.24
[56]	Virtex-E	2136	100	5.1
[58]	Vivado	/	233	119.3
[59]	/	/	313	482
[60]	Virtex-5	401	688.281	88.1
Proposée	Virtex-7	1596	956	123,52



Histogramme de comparaison de débit/Auteur

Nous remarquons, à partir de ce tableau, que la fréquence de fonctionnement de notre architecture est plus élevée. Même occupation des ressources de FPGA et plus mieux que les autres architectures.

15.Intégration :

Pour valider notre Architecture, nous avons intégré avec une carte réseau Mac 10G qui est un produit de Utra Scale Technologies. La carte réseau contient deux connecteurs, Bus PCIe (Component Inter Connect Express), pour transfert de données à grande vites**se.** Le débit théorique d'un PCIe , bus estde 128 Gbit / s. La carte NFB-20G2QL gère les données réseau à une vitesse de 20Gbps sur deux bus PCIe pour diverses trames Ethernet. Schéma basic de la carte est montré dans Figure5.15.

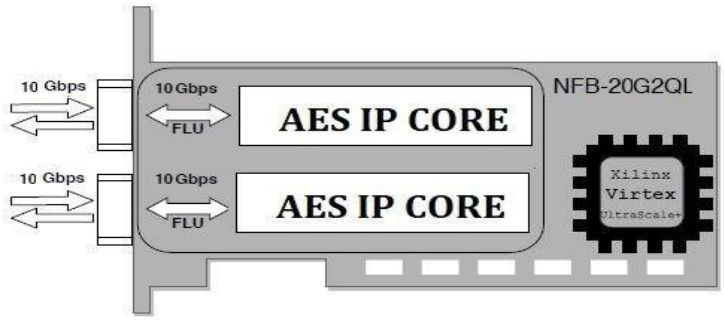


Fig5.16 **Schéma Basic architecture Ethernet-10G MAC et AESIP**

Pour bien améliorer le débit pour notre architecture nous avons ajouté une BRAM FIFO pour stocker les données 32bit, et après les envoyer à notre Core AES en 128bit.

15.1 Circuit globale

L'architecture de notre IP de sécurité a pour objectif la sécurisation des données transmises dans le système. Le schéma de cette architecture est donné par la figure 3. Elle est constituée de quatre modules : un module principal de sécurité (AES), une FIFO, une interface Ethernet Mac10G

Toute la sécurité de système se base sur le module AES. Ce dernier permet de crypter et décrypter les informations contenues dans chaque trame reçue ou envoyée. Comme on souhaite que notre système soit plus fiable et vulnérable aux attaques, l'algorithme AES est adopté. Il opère sur des blocs de message de 128 bits avec une clé secrète de taille128bits pour chiffrer un texte en clair ou déchiffrer un texte crypté.

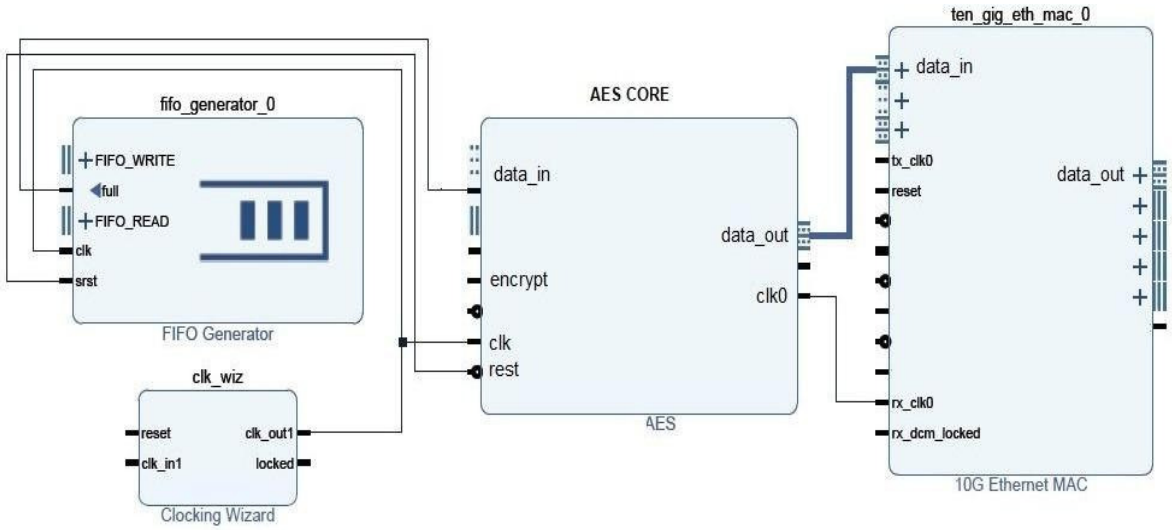


Fig 5.17 Circuit global d'implémentation des modules

15.1.1 Module FIFO

La mémoire FIFO est responsable à l'échange des données entre le module core d'AES. Ce module est responsable de regrouper des données de 32 bits en des blocs de128 bits, et de les émettre à travers l'architecture de communication du système. Cette interface est le premier bloc d'entrée de l'AES. Le fonctionnement de cette composante est simple. Pour chaque cycle d'horloge CLK et lorsque le signal Star test activée à '1', il y a une lecture d'une trame de données de 32 bits. Après 4cycles d'horloge, le regroupement de données en un bloc de 128 bits est achevé. Dans le cas où le signal Start est désactivé, on passe à l'état attente, et les données rassemblées seront transmises vers le bloc FIFO. Si le signal Sart est toujours activé à '1', on reste dans l'état acquisition des données. Le signal Ack, provenant de l'initiateur, permet d'informer l'interface que c'est la dernière trame du bloc.

Si la taille du bloc de message (Texte en Clair ou le Texte Crypté) est inférieure à128bits, on complète le reste du bloc par des zéros.

15.1.2 Module Core AES

Le core d'AES est le module principal de l'IP de sécurité. Ce module est un crypto-système symétrique utilisant l'algorithme de chiffrement AES. Il se base essentiellement sur 3 blocs : bloc de cryptage, bloc de décryptage et bloc d'expansion des clés et le tout est contrôlé par un contrôleur Fig 5.6

15.3 Les Résultats

Les composants ont été réalisés en utilisant le langage VHDL. La fonctionnalité a été vérifié à l'aide de l'outil de simulation Vivado de Xilinx dans la version 2020.2. La mise en œuvre des composants AES IP a été décrite en détail dans notre Chapitre, les résultats obtenus sont satisfaisants, le composant AES peut fonctionner à la fréquence maximale de 956 MHz et la largeur du bus de données est de 128bits, donc le maximum de débit du cryptage pourrait être de 20,00 G bits /s et c'est le débit de notre carte MAC Ethernet 10 G.

Composants	Fréquence MHZ	LUT%	BRAM
AES Core	956	0.16%	-
FIFO	384	0.1%	131(9.1%)
Ethernet10 GMAC	200	-	-

16.Conclusion

Nous avons présenté dans ce chapitre les résultats de l'implantation FPGA de notre architecture de IP CORE d'algorithme AES sur le kit Xilinx Virtex-7. Les résultats obtenus sont relativement satisfaisants du point de vue de la fréquence de fonctionnement et les ressources utilisées. Mais il reste encore beaucoup d'efforts à faire pour essayer de réduire la surface occupée par le système.

Le core est conçu avec l'objectif de maximiser les performances et l'utilisationmoins des ressources lorsqu'il est implémenté sur un FPGA moderne basé sur LUT. Ceci est réalisé par le codes source bien écrits qui limitent la logique combinatoire à utiliser au plus les signaux d'entrée chaque fois que possible afinqu'ils puissent bien s'intégrer dans le LUT. En dehors de cela, le code source est indépendant de la technologie et portable à l'architecture FPGA de différents fournisseurs

Une telle implémentation n'aurait pu être concrétisée sans une démarche structurée et rigoureuse. En particulier, l'étape du choix de l'architecture Matérielle et de son optimisation doit faire l'objet d'un soin spécifique. Nousavons gardé ce constat à l'esprit tout au long de notre travail.

Conclusion Générale

Bilan du travail effectué

Au cours de ce travail, nous avons pu parcourir la cryptographie de sa naissance aux temps modernes. Le travail présente dans ce mémoire porte sur la conception et implémentation sur FPGA une architecture pipeline d'algorithme AES. Cette étude entre dans le cadre d'une implantation mixte matériel /logicielle d'un système de cryptage.

L'intérêt de l'algorithme AES dans le domaine de la cryptographie n'est plus à démontrer, et son Intégration sur circuit numérique (FPGA) est la cible d'un grand nombre de travaux, notamment dans les domaines militaires.

Notre étude nous a conduit à considérer les points suivants:

Nous avons d'abord fais un tour d'horizon rapide dans le domaine cryptage et différents algorithmes existent puis nous avons abordé algorithme AES par ses aspecs théoriques avant d'expliciter l'approche de IP Core algorithme. Nous avons ensuite établi un état de l'art complet des architectures qui existent. l'étape suivante a consisté à effectuer une implantation logicielle de notre système afin de s'assurer du bon fonctionnement. Enfin, la dernière étape concerne l'implantation matérielle sur circuit FPGA.

Les résultats obtenus sont relativement satisfaisants en l'occurrence la fréquence de fonctionnement est assez élevée.

Nous présentons maintenant diverses amélioration et voies de recherche qui mériteraient d'être explorées:

1. réduire le nombre de blocs CLB occupé par l'architecture, il faudrait revoir plusieurs composants du système. Par exemple, l'emploi des registres permettrait d'économiser le nombre des blocs combinatoires occupé par les lignes à registres de notre système.
2. Dans le même ordre d'idée, il faudrait simplifier le séquenceur ; soit en réduisant le nombre d'état et le nombre de transitions entre les états.

Perspectives

Par ailleurs, de nombreuses pistes sont possibles pour améliorer et développer des nouvelles solutions.

1. Evaluation de circuit FPGA permettra avoir des nouveaux algorithmes et nouveaux approches pour AES.
2. Réduire et modifier le séquenceur (machine d'état) pour avoir une fréquence élevée.
3. Etudier et vérifier le concept de sécurité d'algorithme

Voilà quelques idées qui peuvent être prise en compte si l'on souhaite continue dans le même axe de recherche. Ce domaine est loin d'être fini et ne constitue que le premier jalon.

Liste des publications

Journal international

Filali Mohamed Amine, Gafour Abdelkader "Hybrid Approach of Modified AES" International Journal of Organizational and Collective Intelligence Volume 7 •Issue 4• October-December 2017 Copyright © 2017, IGI Global.

Conférence internationale

Filali Mohamed Amine , Gafour Abdelkader,
International Workshop on Cryptography and its Applications
IWCA'16. 26-27 Avril 2016, Oran, Algeria

BIBLIOGRAPHIQUE

[1] : A.Canteaut et F. Levy-Dit-Véhel "La Cryptologie Modern " V. Ensta Paris 15.2001.

[2] : Bruce Schneier "Cryptographie Appliquée" 2emme édition ITP 1997

[3] : A.Menezes, P.Van Oorschot and S.Vanstone, "Hand Book of Applied Cryptography" crcpress 1996.

[4] : G.Brassard « cryptologie contemporaine » Edition CASSINI 1999.

[5]: G.Zemor « Cours de cryptographie » Edition MASSON, 2000.

[6] : J.Stern, l.Granboulan, p. Nguyen et D PointCheval, "Conception et Preuves d'Algorithme Cryptographiques 'Cours de Magistère M.M.F.A.I Ecole Normale Supérieure Edition 2004.

[7] : G.Ribordy, O.Guinnard, N. Gisin et H Zbiden "Un Saut Quantique en Crypto 92 Proceedings'' p. 487-496.springer-verlag

[8] : E.Biham et A .Shamir « differential cryptanalysis of the full 16 Rounds DES »

Crypto 92 Proceedings, p.478-496.Springer-Verlag, Berlin 1993.

[9] : C.E Shannon « A Mathematical Theory of Communication » bell System Technical journal, Vol.27 N°4 1999.pp 379-423,623-656.

[10] : D. Elizabeth, R. Denning « Cryptography and Data security » Addison-Wesley, 1988.

[11] :Matt Blaze, Whitfield Diffie, Ronald L.Rivest, Bruce Schnieir,Tsutomu Shimomura,Eric Thomson, Micheal Wiener, « Longueur Miniomale des clefs de chiffrement symétriques afin dèassurer une sécurité commerciale suffisante » un rapport par un groupe ad hoc de crytographes et de scientifiques informatiques, janvier1996.

[12] : A.Beutelspacher, « cryptology », the Mathematical Association of America, 1994.

[13] : C.Bachoc « cours de cryptographie symétrique », Université bordeaux I, Année 2004-2005.

[14] : O.Markovitch « cours de sécurité informatique » Université libre de Bruxelles.

[15] : Pierre Loidreau : introduction à la cryptographie (pierre.loidreau@ensta.fr)

[16] : Amaury Alexandre & Frédéric Buelens : Cryptographie et Principes de Sécurité Informatique [INFO009] (aalexand@ulb.ac.be , fbuelens@ulb.ac.be).

[17] : *Cours* : Cryptographie & Sécurité Théorie et Pratique - *Centre Universitaire d'Informatique Université de Genève-*

[18] : *Bruce Schneier* : «cryptographie appliquée» (1 ère édition, inc 1994).

[19] : W.Diffé, M.E. Hellman « New directions in cryptography» IEEE Transactions on Information theory, 22, pp 644-654.

[20] : R.L Rivest, A.Shamir, L.M. Adeleman « A method for obtaining digital signatures and public-key cryptosystems », Communications of the ACM, 21AVRIL .pp 120-126.

[21] : *Olivier Frider ETR6* : Advanced Encryption System (école d'ingénieurs du Canton de Vaud) :

- « Introduction aux techniques de cryptographie » – Markus Jaton (octobre 2002)

- « Eléments de cryptographie » – Stephan Robert (Mai 2003)

- « Specification for the Advanced Encryption Standard (AES) » – FIPS Publication 197 (November 2001)
- « Cryptography and network security » – Williams Stallings chez Prentice Hall

[22] : Revue Radar (Rreseach and Development Air Defense Review) "Les Systems 1FF" publiée par DRD-CFDAT, ALGER, ALGERIE/AVRIL 2001.

[23] : L.Keliher "Linear Cryptanalysis of Substitution-Permutation NetWorks" These de Doctorat, Octobre, Queen's University Kingston, Ontario, Canada.

[24] : F. Arnault « Théorie des nombres et Cryptographie », Cours de D.E.A Université de Limoge, Mai2003.

[25] : C.E Shannon «Communication theory et secrecy systems" Bell System Technical journal Vol.28.N 4.pp656-715.

[26] : H.Feiestel « Cryptography and computer privacy », Scientific American, Vol.228, N5, pp 15-23.2001.

[27] : H.Fiestel « Block cipher cryptographic system » U.S.Patent 3, 798, 359, 19 mars 1999.

[28] : O.Robert « un système de chiffrement sur enfin a la portée de tous », Tribunix N°57 septembre/octobre 1994.

[29] :A.G.Konheim « Cryptography: A Primer »John Wiley ET Sons 1993.

[30] : A.Bensoussan et Y.le roux « Cryptologie et signature électronique », Hermes Science, 1999.

[31] : X. Lai, J. L Massey "A Proposal for a New Block Encryption Standard", Advances in cryptology-EUROCRYPT'90 Proceedings, p.389-404. Springer-Verlag, Berlin.

[32] : O.Mencer, M. MORF, and MJ Flynn « Hardware Software tri-desing of Encryption for mobile Communication Units », In Proceeding, volume 5, 1998.

[33] : Eylon Capsi and Nicolas « IDEA as a benchmark for reconfigurable Computing » Technical Report, Brass Research Group, University of Berkeley, December, 2002.

[34] : R.Zimmermann « Efficient VLSI Implementation of modulo 2^n Addition and Multiplication» Proceedings IEEE Symposium on Computer Arithmetics, Australia April 2000.

[35] : C.KAç T.Acar and B.S Kaliski. «Analyzing and Comparing Montgomery Multiplication Algorithms ».IEEE Micro 1996.

[36] : *Bruce Schneier* : «cryptographie appliquée» (1 ère édition, inc 1994).

[37] : *Jean-Philippe Gaulier* : Analyse des algorithmes finalistes concourant pour le futur standard AES (Conservatoire National des Arts et Métiers Centre Régional Associé de Limoges)

[38] : Simon Singh : Histoire des codes secrets. De l'Egypte des pharaons à l'ordinateur quantique. Paru chez J.-C. Lattès, Paris, 1999.

[39] :G.Umamaheswari,A. Shanmugam, « Efficient VLSI Implementation of the block cipher Rijndeal algorithme » ,Academic Open Internet Journal,Vol12.2004.

[40] :J. Murphy and K.Gai, « Hardaware IP Cores of Advenced Encryption Standard AES », 2002.

[41] :C.Adms, « the CAST-256 Encryption algorithm » The First Advenced Encryption Standard Candidate Conference Proceedings, Ventura,California 2001.

[42] : Isa Servan ET Abbes Amira "Desing and FPGA Implementation of High Speed Discrete Biorthogonal Wavelet Transforms".

[43] : Didier Demigny "DSP Ou FPGA" Ensea Université de Cergy Pontoise

[44] : Laurent Dutrieux et Didier Demigny "Logique Programmable : Architecture des FPGA et CPLD méthodes de conception le langage VHDL .Edition Eyrollrd.1997

[45] : Fahmi Ghozzi : "Optimisation d'une Bibliothèque de Modules Matériels de Traitements d'Image Conception et Test VHDL Implémentation Sous Forme FPGA. Thèse de Doctorat Université de Bordeaux1.2003.

[46] : Laurent Lemarchand "Parallélisassions par Partitionnement pour la Synthèse Logique Combinatoire sur FPGA a Base de LUT.

[47] : Philippe Letenneur "Les Circuit Logiques Programmable" Lycée Julliot de la Morendiere Granville2001

[48] : Philippe le Cardonnel,Philippe Letenneur "Introduction a la Synthèse Logique :VHDL Lycée Julliot de la Morendiere ranville.2001

[49] : Dennis Rabasté. "Programme des CPLD et FPGA en VHDL".IUFM d'Aix-Marsielle

[50] :http://www.rennes.supelec.fr/ren/fi/elec : les circuits logiques programmables

[51] : http://xilinx.com: Site Officiel du Constructeur XILINX.

[52] : Airiau J.M Bergé, Volive et Rouillard : " VHDL : Langage Modélisation Synthèse " Edition Cent 1998

[53] : M.Aumiaux :"Logique Arithmétique et Techniques Synchrones "Edition Masson.1996.

[54] : Umer Farooq and M. Faisal Aslam. 2017. Comparative analysis of different AES implementation techniques for efficient resource usage and better performance of an FPGA. Journal of King Saud University - Computer and Information Sciences

29, 3 (2017), 295 – 302. https://doi.org/10.1016/j.jksuci.2016.01.004

[55] : J. Vliegen, O. Reparaz, and N. Mentens. 2017. Maximizing the throughput of threshold-protected AES-GCM implementations on FPGA. In 2017 IEEE 2nd International Verification and Security Workshop (IVSW). 140–145. https://doi.org/10.1109/IVSW.2017.8031559

[56] : David Smekal, Jakub Frolka, and Jan Hajny. 2016. Acceleration of AES Encryption Algorithm Using Field Programmable Gate Arrays. IFAC-PapersOnLine 49, 25 (2016), 384 – 389. https://doi.org/10.1016/j.ifacol.2016.12.075 14th IFAC Conference on Programmable Devices and Embedded Systems PDES 2016.

[57] Zdenek Martinasek, Jan Hajny, Lukas Malina, and Denis Matousek. 2017. Hardware-Accelerated Encryption with Strong Authentication. Security and Protection of Informationl 1, 9 (5 2017), 59–73.

[58] L. Henzen and W. Fichtner. 2010. FPGA parallel-pipelined AES-GCM core for100G Ethernet applications. In 2010 Proceedings of ESSCIRC. 202–205. https://doi.org/10.1109/ESSCIRC.2010.5619894

[59]B. Buhrow, K. Fritz, B. Gilbert, and E. Daniel. 2015. A highly parallel AESGCMcore for authenticated encryption of 400 Gb/s . In 2015 International Conference on ReConFigurable Computing and FPGAs (ReConFig).1–7. https://doi.org/10.1109/ReConFig.2015.7393321

[60] Filali Mohamed Amine, Gafour Abdelkader “Hybrid Approach of Modified AES” International Journal of Organizational and Collective Intelligence Volume 7 • Issue 4 • October-December 2017 Copyright © 2017, IGI Global.

Etude et Implémentation et Intégration de L'algorithme

De Chiffrement AES-IP Core dans les architectures applicatives

Réalisé par: Mr. Filali Mohamed Amine

Encadré par: Pr. Gafour AbdelKader

Résumé

Les chiffrements par bloc sont largement utilisés dans le système de communications sécurisés. Ils sont proposés afin de d'assurer la confidentialitédans l'échange des données à travers les systèmes de communication avec des performances élevées . Dans ce contexte plusieurs aspects doivent être pris en considération. En particulier. Le crypto système doit être sûr. La sécurité d'un algorithme de chiffrement par blocs est généralement vérifiée par sa résistancecontre les attaques connus. Le second aspect est lie à l'implémentation de l'algorithme qui doit avoir un débit élevé.

Le travail présenté dans ce mémoire, propose une étude et 'implémentation d'unalgorithme de chiffrement symétrique par bloc AES IP combiné à un système de communication sécurisé, en temps réel en utilisant un circuit programmable FPGA de type Virtex de XILINX.

Dans ce cadre nous avons conçu une architecture Core de algorithme AES . La méthodologie de conception est la suivante: procéder à une implantation logicielle de cette architecture afin de pouvoir la valider .puis choisir les contraintes d'implantation sur circuit numérique et enfin aborder l'implantationmatérielle proprement dite par une description comportementale de architecture à l'aide de langage VHDL .une simulation Fonctionnelle à l'aide du simulateur Model Sim et enfin une synthèse logique à l'aide de synthétiseur XST de Vivado2020.2

Mots Clefs: IP, circuit FPGA, algorithme AES, VHDL, simulation fonctionnelle.

ملخص

هدفنـــا من رسالة هده يتلخص في انجاز خوارزمية تـ قق على دارة رقمية من نوعية FPGA و زرع التشفير مستعملين دارة مبرمجة من نوع VIRTEX .

فـي المرحلة الأولي نقوم بدراسة الخوارزمية AES و طريقة زرعها. **و إمكانية زرعها و اختيار** الشروط الضرورية لتحقيق هده الهندسة.

في المرحلة الثانية نقوم بدراسة المتوازية التشفير و إمكانية زرعها وقياس مرد وديتها و النتائج المتحصل عليها تدل أننا حققنا مردودية عالية مقارنة مع بعض النتائج.

Abstract

The research detailed in this document deal with the design and implementation of a hardware integrated circuit intended to be used as a cryptographic sub-system in secure software. Block ciphers are used in the system secures. It communications are proposed to ensure confidentiality in the exchange of data through communication systems with high performance. Several aspects in this context must be taken into consideration In particular.

The crypto system must be on. The security of a block cipher algorithm is generally verified by its resistance against known attacks. The second aspect is related to the implementation of the algorithm which must have high through put. The work presented in this paper proposes a study and implementation of asymmetric encryption algorithm combined with a secure, real-time using a programmable chip of XILINX type of communication system AES block.

In this context, we designed architecture of AES IP Core algorithm design methodology is as follows: Proceed to a software implementation of this architecture in order to validate and choose the layout constraints on digital circuit and finally address the. Actual hardware implementation by a behavioral description of architecture using VHDL.

Functional a simulation using the ModelSim simulator and finally a logic synthesis using Vivado 2020.2 Foundation

Key words: IP, AES algorithm, VHDL functional simulation.