N°d'ordre : 2017/IMI08/97/482

جامعة محمد بوضياف -المسيلة
Université Mohamed Boudiaf - M'sila
كلية التكنولوجيا
Faculté de Technologie
قسم الإلكترونيك
Département d'Électronique

MEMOIRE DE MASTER

Présenté par : DAHMANE Zouhir
ABDELLI Lyamine

DOMAINE : SCIENCES ET TECHNOLOGIES

FILIERE : ELECTRONIQUE

OPTION : INSTRUMENTATION ET MAINTENANCE INDUSTRIELLE

SUJET

Implémentation d'un algorithme de cryptage sur un circuit FPGA

Soutenue le : 21 Mai 2017 devant le jury composé de :

Mr. GUERMAT Noubeil	**M.C.B - Université Mohamed Boudiaf - M'sila**	**Président**
Mr. BENHACENE Madani	**M.C.B - Université Mohamed Boudiaf - M'sila**	**Encadreur**
Mr. OUDIRA Houcine	**M.C.B - Université Mohamed Boudiaf - M'sila**	**Examinateur**

Promotion : Mai 2017

Remerciements

Nous tenons à remercier le dieu de nous avoir donné la patience de terminer ce travail.

Au terme de cette étude, nous souhaitons remercier chaleureusement notre encadrant de mémoire ***Dr. BENHACENE Madani*** *pour leur suivi et ses encouragement, ses orientations et ses précieux conseils.*

Nous remercions également tous les professeurs et les enseignants du département d'électronique pour leurs conseils constructifs et encouragements qui ont été à la fois un complément et supplément indispensable dans notre projet de fin d'étude.

Enfin, nous exprimons nos reconnaissances à tous les membres de jury d'avoir acceptés de lire ce manuscrit et d'apporter les critiques nécessaires à la mise en forme finale de cet travail.

Merci à tous

Dédicaces

Je dédie ce mémoire :

À la mémoire de mon père Bilkacem, *parce que il me tendit son encouragement et son grand sacrifice et il espérait toujours de ma regarder succès dans ma vie, il est pu créer le climat affectueux et propice à la poursuite de mes études.*

À ma très chère mère, *qui m'a encouragé à aller de l'avant, en lui exprimant mon amour, mon respect et vive gratitude pour sa patience, son amour et ces prières qui sont toujours m'encouragent à la réussite tous les long de mes études que le dieu la protège.*

À la mémoire de mon frère Hassen, *qui m'a aidé à surmonter les obstacles de la vie par ses conseils constructifs et ses encouragements, je souhaite que nous se réunions au Paradis.*

À mes frères, à mes sœurs, *je leur souhaite la belle vie avec mes sincères gratitudes, je les remercie pour leurs sacrifices.*

Tous mes chers amis, tous mes camarades de la promotion IMI 2017

À toute ma grande famille.

D. Zouhir

Dédicaces

Je voudrais dédier cet humble travail :

À ma chère mère ;

À la mémoire de mon père ;

À tous mes proches de la famille ABDELLI, et plus

Particulièrement, mes sœurs et mes frères tout à son nom.

Tt aussi, Je dédie ce travail :

Aux personnes qui m'ont toujours aidé et encouragé, qui étaient

Toujours à mes côtés, et qui m'ont accompagnaient durant mon

Chemin d'études supérieures, mes aimables amis, collègues d'étude,

De l'Université de M'sila.

A. Lyamine

SOMMAIRE

Chapitre2. Algorithme cryptographique DES

Chapitre3. Implémentation sur FPGA de l'architecture DES

"NOMENCLATURE"

DES	Data Encryption Standard
AES	Advanced Encryption Standard
RSA	Revenu de Solidarité Active
FPGA	Field Programmable Gate Array
DSS	Digital Signature Standard
ECB	Electronic Code Book
CBC	Cipher Block Chaining
OFB	Output Feedback
CFB	Cipher Feedback
XOR	Exclusive OR
IV	Initial Value
IDEA	International Data Encryption Algorithm
FIPS	Federal Information Processing Standard
ANSI	American National Standards Institute
NIST	National Institute of Standards and Technology
GSM	Global System for Mobile
RC4	Rivest Cipher 4
SSL	Secure Socket Layer
PI	Permutation initiale
PF	Permutation finale
IBM	International Business Machines
S-Box	Substitution-Box
PC 1	Permuted Choice 1
TDES	Triple DES
IEEE	Institute of Electrical and Electronic Engineers
VHDL	VHSIC (Very High Speed Integrated Circuit) Hardware Description Language
CLB	Configurable Logic Block
CPU	Central Processing Unit
POS	Point Of Sale
ASIC	Application Specific Integrated Circuit
GPP	General Purpose Processor

DSP	Digital Signal Processing
LUT	Look-Up-Table
ISE	Integrated Software Environment
IO	Input Output
GCLK	Global clock
RTL	Register-Transfer Level
MUX	Multiplexer
OV	Output Validator
FDE	D Flip-Flop with Clock Enable

Liste des figures

Liste des tableaux

Introduction général

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 et confidentielle. La dissimulation d'information a été utilisée comme instrument de sécurisation très important pour les stratégies militaires et échange de données secrètes entre les utilisateurs [1]. La nécessité de transfert sécurisé d'information a traversé le temps et est encore énormément utilisée dans le monde numérique puisque la plupart de ces informations sont stockées sur des ordinateurs électroniques et envoyées à d'autres ordinateurs via internet.

Pour éviter que les données confidentielles ne tombent dans les mauvaises mains, les scientifiques et les chercheurs ont étudié des différentes méthodes et techniques mathématiques reliées aux aspects de sécurité et dissimulation de l'information, parmi de ces méthodes, on a le chiffrement qui considère l'une des techniques utilisées pour protéger l'information importante. La cryptographie, ou art de chiffrer, coder les messages, est devenue aujourd'hui une science à part entière car il joue un rôle essentiel dans la protection des informations électroniques contre les menaces d'une variété d'attaquants potentiels et aussi, elle permet ce dont les civilisations ont besoin depuis qu'elles existent le maintien du secret. Pour éviter une guerre, protéger un peuple, il est parfois nécessaire de cacher des choses.

Le domaine de la cryptographie, longtemps resté un domaine réservé aux services des gouvernements et des services d'espionnage et contre-espionnage des militaires, est devenu depuis les années 70 un domaine de recherche grand public porté par de nombreux laboratoires académiques.

La cryptographie étant un sujet très vaste, ce document se focalisera essentiellement sur les méthodes de chiffrement dites modernes, c'est-à-dire celles étant apparues et utilisées après la seconde guerre mondiale. On passera en revue la saga du DES et de l'AES, en passant par le fameux RSA, le protocole le plus utilisé de nos jours. Ayant longtemps été des sociétés possédant de gros moyens financiers, la cryptographie s'est au fil du temps ouverte au grand public, et est donc un sujet digne d'intérêt. [2]

Actuellement, il existe deux types de cryptographie moderne : la cryptographie asymétrique et la cryptographie symétrique. Pour cette dernière catégorie, nous allons présenter deux algorithmes très connus DES et AES en mettant l’accent sur le DES car

c’est un standard ancien recommandé pour le chiffrement symétrique, de plus il est l'épine dorsale des systèmes de chiffrement en général.

Le travail que nous aurons présenté aujourd’hui est une étude et simulation puis l’optimisation d'une implémentation matérielle de l'algorithme cryptographique DES *(Data Encryption Standard)* sur un circuit reconfigurable basée sur FPGA.

Pour faciliter la compréhension de notre projet et pour donner un aperçu général, nous présentons ci-dessous brièvement la structure de notre travail :

- Dans le premier chapitre, nous présenterons **Les concepts fondamentaux de la cryptographie**, comme ils sont définis par les standards internationaux de la sécurité.
- Le deuxième chapitre décrit **l’Algorithme cryptographique DES**, on présentera le DES, ses caractéristiques, ses étapes de fonctionnement.
- l’objectif principal de troisième chapitre est **l’Implémentation sur FPGA de l’architecture DES**. Nous présenterons plusieurs variantes d’algorithmes en fonction du choix des caractéristiques des circuits utilisés ainsi que des objectifs visés.

Et à la fin, nous terminerons notre travail par une conclusion générale et des perspectives.

Chapitre

01

LES CONCEPTS FONDAMENTAUX DE LA CRYPTOGRAPHIE

Chapitre 1. Les concepts fondamentaux de la cryptographie

I.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 Jules César l'utilisait déjà. [3] Jusqu'à récemment, elle est restée confinée aux domaines militaires et diplomatiques. Toute fois le développement des télécommunications et des échanges d'information sous forme électronique au cours des vingt-cinq dernières années a permis d'émanciper la cryptographie de son statut de science (top secrète).

I.2. Définition de la cryptographie

La cryptographie est la science qui utilise les mathématiques pour le cryptage et le décryptage de données.

Elle vous permet ainsi de stocker des informations confidentielles ou de les transmettre sur des réseaux non sécurisés (comme l'internet), afin qu'aucune personne autre que le destinataire ne puisse les lire.

Alors que la cryptographie consiste à sécuriser les données, la cryptanalyse est l'étude des informations cryptées, afin d'en découvrir le secret. La cryptanalyse classique implique une combinaison intéressante de raisonnement analytique, d'application d'outils mathématiques, de recherche de modèle, de patience, de détermination et de chance. Ces cryptanalyses sont également appelés des pirates.

Cryptologie = Cryptographie + Cryptanalyse.

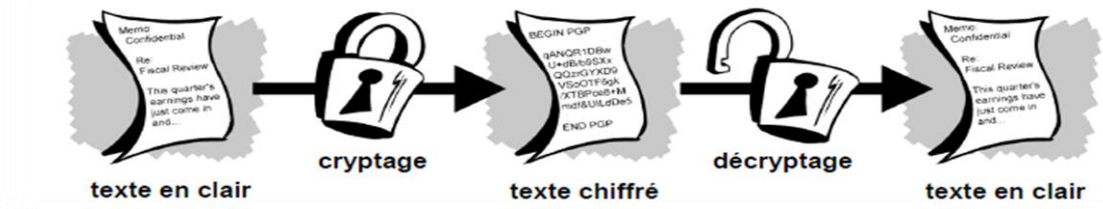

Fig. I.1 : Schéma général de la cryptographie. [4]

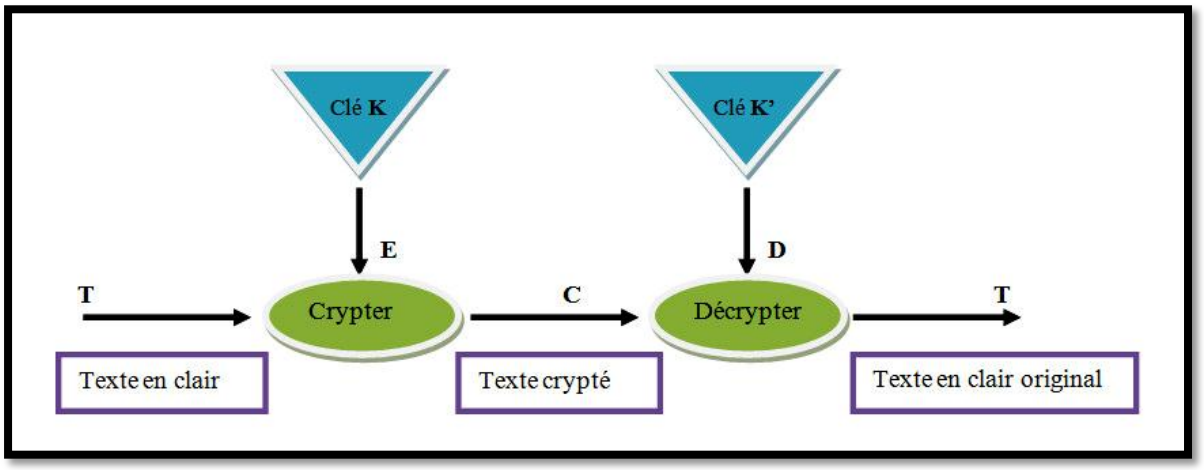

Fig. I.2 : Schéma de processus cryptage et décryptage.

T : Le texte en doit être crypter.

E : Le procédé de cryptage.

D : Le procédé de décryptage.

C : Le texte crypté.

K : La clé de cryptage.

K' : La clé de décryptage.

- Pour faire une vraie sécurité, tous les algorithmes modernes de cryptage utilisent une clé **(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 :

  $\mathbf{E_K (T) = C.}$

  $\mathbf{D_{K'} (C) = T.}$

  $\mathbf{D_{K'} (E_K (T)) = T.}$

I.3. Histoire de la cryptographie

Voici donc un rapide descriptif des faits marquants et des hommes 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. [5]

Le destinataire avait un bâton identique permettant d'enrouler le cuir afin de déchiffrer le message.

- **Au 9eme 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 décryptage appelée analyse des fréquences.
- A partir de **1226** une timide politique de cryptographie apparait dans les archives de Venise, ou des points ou des croix remplacent les voyelles dans quelques mots épars (répandu). [6]

Citons 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** inventions de RSA le premier système concret de cryptographie a clé publique.
- **1985** inventions du système cryptographie el Gamal.
- **1991** adoptions du premier standard de signature, ISO9796, basé sur RSA.
- **1994** adoption du DSS, standard de signature basé sur l'algorithme discret.
- **2000** adoption de rijndeal comme AES (successeur du DES). [7]

I.4. Les principaux services offerts par la cryptographie

Les principaux services à garantir par l'application de la cryptographie sont:

Confidentialité: un mécanisme pour transmettre des données de telle sorte que seul le destinataire autorisé puisse les lire.

Intégrité des données: un mécanisme pour s'assurer que les données reçues n'ont pas été modifiées durant la transmission, frauduleusement ou accidentellement.

Authentification: un mécanisme pour permettre d'authentifier les utilisateurs de façon limiter l'accès aux données, serveurs et ressources aux seules personnes autorisées (un mot de passe par un nom de login ou un certificat numérique).

Non-répudiation: un mécanisme pour enregistrer un acte ou un engagement d'une personne ou d'une entité de telle sorte que celle-ci ne puisse pas nier avoir accompli cet acte ou pris cet engagement. Ce mécanisme se décompose:

- non-répudiation d'origine l'émetteur ne peut nier avoir écrit le message.
- non-répudiation de réception le receveur ne peut nier avoir reçu le message.
- non-répudiation de transmission l'émetteur du message ne peut nier avoir envoyé le message. [8]

I.5. Les lois de Shannon

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. [9]

Les principes fondamentaux d'un algorithme de cryptographie sont bases 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 chiffre dépendant d'un nombre aussi grand que possible d'éléments d'information du texte clair. [10]

I.6. Principe de Kirchhoff

C'est un principe fondamental de la cryptographie, il a été énoncé par Kirchhoff à la fin du dix-neuvième siècle. Il exprime que la méthode de chiffrement utilisée doit *" pouvoir tomber sans inconvénient aux mains de l'ennemi ".* Autrement dit, la sécurité d'un chiffrement doit reposer uniquement sur la protection de la clé. [11]

I.7. Cryptographie classique

La cryptographie classique a été conçue avant la création des ordinateurs et qui ont donné les concepts et les bases pour l'évolution de plusieurs algorithmes symétriques encore utilisés de nos jours. Les crypto-systèmes classiques sont groupés en chiffrement mono alphabétique et poly alphabétique .Le chiffrement mono alphabétique est très primaire, il s'agit d'une substitution simple. Chaque lettre est remplacée par une autre lettre ou symbole conformément à un certain algorithme. [12]

I.7.1. Chiffrement par décalage

Un des systèmes les plus anciens et simples est le chiffrement par décalage. Il est aussi appelé code de César, la clé secrète de codage est le nombre d de décalage circulaire que l'on applique à chaque lettre de l'alphabet.

Si ce codage est très simple à mettre en place comme la figure **(I.3)**, il est aussi aisé de le casser. En effet, on peut utiliser une méthode exhaustive qui consiste à tester une à une les **25** valeurs possibles de la clé **d**. Sinon, on peut se baser sur une analyse des fréquences d'apparition des lettres dans le message codé, et les comparer à celles usuelles dans un texte français par exemple. La lettre la plus récurrente dans la langue française étant **"e"**, on peut facilement la lier à la lettre la plus présente dans le message codé. On peut ainsi trouver la valeur de **d**.

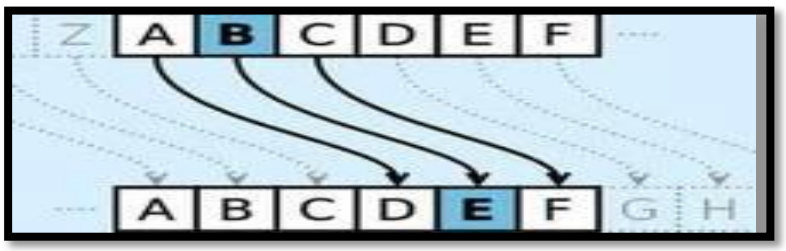

Fig. I.3 : Code de César. [13]

I.7.2. Chiffrement par substitution

Il s'agit d'une méthode plus générale qui englobe le chiffrement par décalage. En effet, à chaque lettre de l'alphabet on fait correspondre une autre, c'est-à-dire que l'on effectue une permutation de l'ensemble des lettres.

Pour la première lettre a, on a 26 possibilité de substitutions. Pour la lettre b, on n'en a plus que 25 et ainsi de suite. Ainsi, il y a :

26*25*24*...*1 = 26 ! = 403 291 461 126 605 635 584 000 000 permutations possibles de l'alphabet français.

Ce chiffrement est beaucoup plus complexe et évolué que le chiffrement par décalage, mais reste néanmoins cassable. [13]

I.7.3. Le code de Vigenère [13]

Le chiffrement de Vigenère est une amélioration décisive du chiffre de César. Il a été élaboré par Blaise de Vigenère (1523-1596), diplomate français du XVIe siècle.

Sa force réside dans l'utilisation non pas d'un, mais de 26 alphabets décalés pour chiffrer un message.

Ce chiffrement utilise une clef qui définit le décalage pour chaque lettre du message (A : décalage de 0 cran, B : 1 cran, C : 2 crans, ..., Z : 25 crans).

Par exemple dans le tableau **(I.1)**, Je veux envoyer un message codé "INSTRUMENT" grâce à la clé "ELECTRIQUE" :

Tableau. I.1 : Exemple sur le code de Vigenère.

Message clair	I	N	S	T	R	U	M	E	N	T
Clé	E	L	E	C	T	R	I	Q	U	E
Décalage	4	11	4	2	19	17	8	16	20	4
Message codé	*M*	*Y*	*W*	*V*	*K*	*L*	*U*	*U*	*H*	*X*

La grande force du chiffrement de Vigenère réside en le multi codage d'une même lettre de l'alphabet. On ne peut donc plus utiliser l'analyse des fréquences classiques pour la cryptanalyse.

I.7.4. Chiffrement de Vernam

Le chiffrement de Vernam (on appelle aussi le masque jetable) a été mis au point lors de la Première Guerre Mondiale, en 1917.Son but était d'être incassable lorsque correctement utilisé, mais ce ne fut qu'un an plus tard que l'objectif fut réellement atteint. Ce chiffrement a notamment été utilisé pendant l'épisode du Téléphone Rouge entre Moscou et Washington.

Il s'agit en réalité d'un chiffrement de Vigenère dont la clé de codage a la même longueur que le message clair. Une clé est constituée de caractères choisis aléatoirement et n'est utilisée qu'une seule fois.

Lorsqu'on examine la clé secrète, on ne peut en déduire aucune information sur le message clair (à part peut-être sa longueur). C'est ce qu'on appelle un code parfait. [13]

I.8. Cryptographie moderne

Dans la société de l'information d'aujourd'hui, l'usage de la cryptologie s'est banalisé. On le retrouve quotidiennement avec les cartes bleues, téléphones portables, Internet ou encore les titres de transport. La cryptologie moderne a pour objet l'étude des méthodes qui permettent d'assurer les services d'intégrité, d'authenticité et de confidentialité dans les systèmes d'information et de communication. Elle recouvre aujourd'hui également l'ensemble des procédés informatiques devant résister à des adversaires.

La cryptologie se partage en deux sous-disciplines :

– La cryptographie qui propose des méthodes pour assurer ces services.

– La cryptanalyse qui recherche des failles dans les mécanismes proposés. [1]

Pour assurer les objectifs de la cryptographie moderne nous pouvons utiliser des algorithmes basés sur des clés. Ces algorithmes sont définis par plusieurs types de cryptographie moderne, on distingue deux approches :

a) La cryptographie symétrique.

b) La cryptographie asymétrique.

I.8.1. La cryptographie symétrique [14]

Le chiffrement symétrique (aussi appelé chiffrement à clé privée ou à clé secrète), ce type se base sur l'utilisation d'une clé pour crypter et décrypter les messages. La sécurité de cette solution repose sur le fait que la clé est connue uniquement par l'émetteur et le récepteur du message.

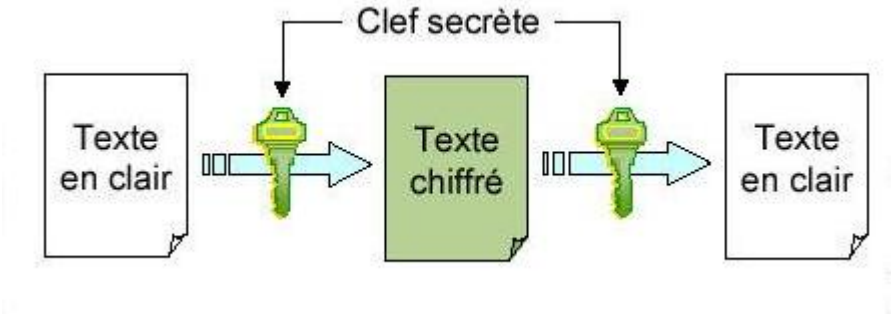

Fig. I.4 : Chiffrement symétrique.

Les algorithmes symétriques sont plus rapides et facilement implémentés sur le matériel.

Ils sont de simples opérations de substitution et de transposition et sont mieux adaptés pour une utilisation sur le réseau Internet filaire et en mobilité. Cependant ces systèmes ne sont pas entièrement adéquats pour résoudre tous les problèmes de sécurité.

Le processus de chiffrement et déchiffrement dépend de la même clé secrète partagée entre l'émetteur et le récepteur. L'émetteur et le récepteur doivent donc se mettre d'accord sur la clé secrète avant qu'ils puissent communiquer d'une façon sécurisée. Un canal sûr d'échange de clé est difficile à établir.

Les crypto-systèmes symétriques ont un problème de gestion de clé. En effet lorsqu'un grand nombre de personnes désirent communiquer ensemble, le nombre de clés augmente de façon importante et la situation devient impraticable. Pour n participants à la communication, on aura besoin de : **n*(n-1)/2** clés secrètes enregistrés.

Pour une meilleure sécurité, elle était détruite et réinitialisée après chaque conversation.

Le cryptage symétrique fonctionne selon deux procédés différents :

- **Le cryptage par bloc** : le cryptage s'effectue sur les blocs de bits.
- **Le cryptage par flot (ou flux)** : le cryptage s'effectue en continu, bit par bit.

I.8.1.1.Le cryptage par bloc (Block Cipher)

C'est une des deux grandes catégories de chiffrements modernes en cryptographie symétrique. Il consiste à un découpage des données en blocs de taille généralement fixe (souvent une puissance de deux comprise entre 32 et 512 bits). Les blocs sont ensuite chiffré ses uns après les autres. Le chiffrement par bloc utilise quatre modes opératoires : Electronic Code Book**(ECB)**, Cipher Block Chaining **(CBC)**, Output Feedback **(OFB)** et Cipher Feedback**(CFB)**. [15]

I.8.1.1.1. Les modes opératoires [7]

Un message réel en générale composé de nombreux blocs. La façon plus immédiate pour chiffrer un tel message est de chiffrer successivement chaque bloc, avec la même clé. Toutefois cette méthode, dite **ECB** (Electronic Code-Book), présente des inconvénients. En particulier, lorsque deux blocs du message(ou de deux messages) clair sont identiques, cela se voit sur le chiffré. De plus un attaquant actif peut permuter des blocs chiffrés/ou en supprimer de telle façon que le clair modifié ait encore un sens, différent du sens initial. La méthode **ECB** à toutefois l'avantage d'être parallélisable, de laisser la liberté de chiffrer/déchiffrer les blocs dans n'importe quel ordre et, en cas de perte d'un bloc chiffré, de ne pas bloquer le déchiffrement des blocs restants.

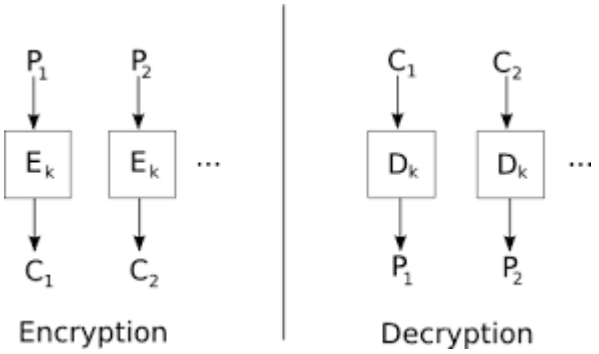

Fig. I.5 : Le mode ECB.

Pour pallier à ces inconvénients, ou utilise souvent des modes opératoires permettant de chaîner les blocs. Ainsi le mode **CBC** (Cipher Block Chaining) consiste, avant le chiffrement d'un bloc, à le masquer par le résultat du chiffrement du bloc précédent au moyen de l'opération **XOR.** Le premier bloc clair est lui aussi masqué, par une valeur habituellement notée **IV** (Initial Value) et de préférence variable (la date et l'heure peuvent

faire une bonne (IV) pour que les chiffrements successifs du même message soient différents.

La valeur initiale **IV** n'a pas besoin d'être secrète, et elle est en générale transmise en clair avant le message chiffré. Noter que si le destinataire reçoit un bloc chiffré avec des bits erronés, cela affecte le déchiffrement de ce bloc et du suivant mais pas des autres.

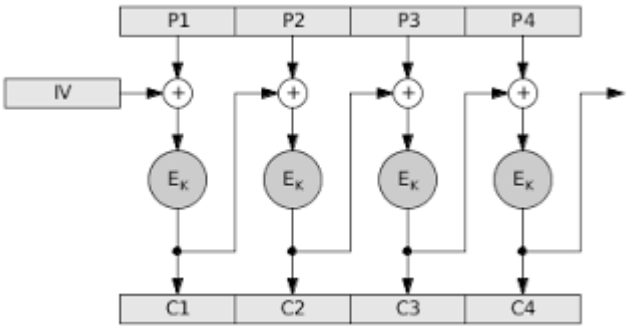

Fig. I.6 : Le mode CBC.

Pour déchiffrer un message en mode **ECB** ou **CBC**, il faut avoir reçu chaque bloc chiffré complet avant de pouvoir obtenir n'importe quel bit du bloc clair correspondant. Cependant, certains applications nécessitent de pouvoir déchiffrer le flux des données au fur et à mesure de leur arrivée, même si la transmission se fait par petits morceaux. Cela est possible en utilisant le mode **OFB** (Output Feed-back). Celui-ci consiste à itérer la fonction de chiffrement sur une valeur initial **IV** et à utiliser le flot de bit pseudo-aléatoires obtenus pour masquer les bits clairs à l'aide de l'opération **XOR**. Il est cette fois –ci très important que la valeur initiale **IV** soit différente pour chaque nouveau message. A noter que l'opération de déchiffrement est identique à celle de chiffrement, et utilise la fonction de chiffrement de bloc et non celle de déchiffrement. Remarquons un inconvénient de cette méthode : un attaquant actif peut modifier des bits du message clair en modifiant les bits correspondants du chiffré (cela peut être aussi un avantage, si un bit du message chiffré est modifié par erreur au cours de la transmission, seul le bit correspondant du message clair sera affecté).

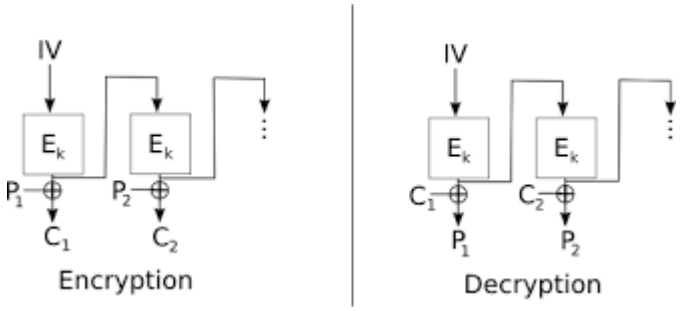

Fig. I.7 : Le mode OFB.

Enfin, le mode **CFB** (Cipher Feedback) est proche du mode **OFB** mais le flot de bits pseudo-aléatoires dépend cette fois des blocs chiffrés. Un attaquant peut encore modifier un bit du clair en modifiant le bit correspondant du chiffré, mais alors le bloc suivant une fois déchiffré sera complètement différent du bloc original.

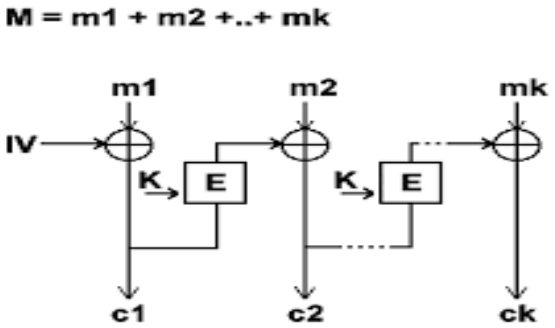

Fig. I.8 : Le mode CFB.

Un exemple de taille de bloc et de clé utilisés par les algorithmes les plus connus:

- **DES** : blocs de 64 bits, clé de 56 bits.
- **IDEA** : blocs de 64 bits, clé de 128 bits.
- **AES** : blocs de 128 bits, clé de 128 à 256 bits.

Pour cette catégorie, nous allons présenter deux algorithmes très connus DES et AES en mettant l'accent sur le DES car c'est un standard ancien recommandé pour le chiffrement symétrique.

- **DES** est le système de chiffrement à clé secrète le plus célèbre et le plus utilisé, il a été adopté comme standard américain en 1977 (standard FIPS 462) pour les communications commerciales, puis par l'ANSI en 1991. Le DES opère sur des blocs de 64 bits et utilise une clé secrète de 56 bits.
- **AES** est le nouveau standard de chiffrement à clef secrète. Il a été choisi en octobre 2000 parmi les 15 systèmes proposés en réponse à l'appel d'offre lancé par le NIST

(National Institute of Standards and Technology). Cet algorithme, initialement appelé RIJNDAEL, a été conçu par deux cryptographes belges, V. Rijmen et J. Daemen. Il opère sur des blocs de message de 128 bits et est disponible pour trois tailles de clé différentes: 128, 192 et 256 bits. [16]

I.8.1.2. Le cryptage par flot (Stream Cipher)

Les algorithmes de chiffrement de flux peuvent être définis comme étant des algorithmes de chiffrement par blocs, où le bloc a une dimension unitaire (1 bit, 1 octet, etc.) ou relativement petite. Leurs avantages principaux viennent du fait que la transformation (méthode de chiffrement) peut être changée à chaque symbole du texte clair et du fait qu'ils soient extrêmement rapides. De plus, ils sont utiles dans un environnement où les erreurs sont fréquentes car ils ont l'avantage de ne pas propager les erreurs (diffusion). Ils sont aussi utilisés lorsque l'information ne peut être traitée qu'avec de petites quantités de symboles à la fois [17]. Quelques algorithmes de cryptographie symétrique par flot:

- **A5**: utilisé dans les téléphones mobiles de type GSM pour chiffrer la communication par radio entre le mobile et l'antenne-relais la plus proche.
- **RC4**, le plus répandu, conçu par Ronald Rivest, utilisé notamment par le protocole WEP, un algorithme récent de Eli Biham – E0 utilisé par le protocole Bluetooth.

I.8.1.3. Les avantages du cryptage symétrique

Les avantages de la cryptographie symétrique sont:

- La rapidité d'exécution (jusqu'à 100 fois plus rapide que les solutions asymétriques).
- La simplicité d'implémentation (gestion d'une seule clé).
- Clés relativement courtes.

I.8.1.4.Les inconvénients du cryptage symétrique

Les inconvénients de la cryptographie symétrique sont:

- Gestion des clés difficiles (nombreuses clés).
- Point faible = l'échange de la clé secrète.
- Dans un réseau de N entités susceptibles de communiquer secrètement il faut distribuer N*(N-1)/2 clés.

I.8.2. La cryptographie asymétrique [14]

Le cryptage asymétrique, contrairement au symétrique, se base sur l'utilisation des 2 clés : publique (pour crypter, elle est accessible publiquement) et privée (pour décrypter le message, elle est gardée secrète). Ce type de cryptage élimine la problématique de la transmission de la clé .Ce mode de cryptage est également nommé le cryptage à clé publique. Il est essentiel que l'on ne puisse pas déduire la clé privée de la clé publique.

Pour bien comprendre le principe, on peut l'illustrer avec l'échange d'une lettre entre un émetteur et un destinataire :

- L'émetteur possède deux clés : privé et publique. Il envoie sa lettre contenant la clé publique au destinataire.
- Le destinataire utilise la clé publique pour crypter son message; il envoie tout à l'émetteur initial.
- L'émetteur utilise sa clé privée pour décrypter le message.

Un exemple d'utilisation du cryptage asymétrique est la transmission d'une clé secrète dans SSL.

Dans la première phase de l'échange, le serveur envoie sa clé publique au client, ensuite le client valide sa fiabilité. Si la validation est correcte, il génère un pré clé principale avec l'utilisation de la clé publique du serveur. Le résultat de cette génération est ensuite envoyé au serveur.

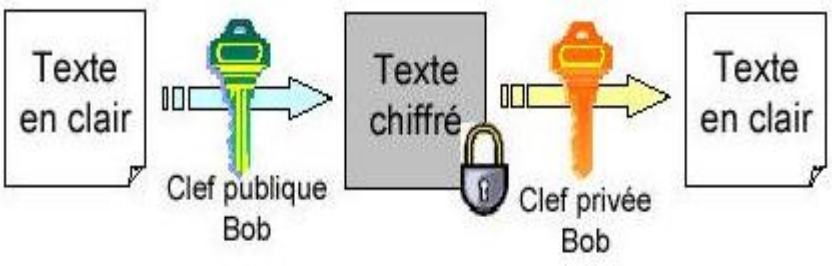

Fig. I.9 : Chiffrement asymétrique.

I.8.2.1. Les avantage du cryptage asymétrique

Les avantages de la cryptographie asymétrique sont:

- L'élimination de la problématique de la transmission de la clé.
- La possibilité d'utiliser la signature électronique.
- L'impossibilité de décrypter le message dans le cas de son interception par une personne non autorisé.
- Une paire de clés (publique/secrète) peut être utilisée plus longtemps qu'une clé symétrie.

I.8.2.2. Les inconvénients du cryptage asymétrique

Les inconvénients de la cryptographie asymétrique sont:

- Le temps d'exécution : plus lent que le cryptage symétrique.
- Le danger des attaques par substitution des clés (d'où la nécessité de valider les émetteurs des clés).
- Taille des clés, plus grand que celle des systèmes symétriques.

I.9. Conclusion

Dans ce premier chapitre, nous avons donnés une présentation des différents types de cryptographies et les propriétés de chaque type.

La cryptographie moderne a pris son essor dans les années 1970 avec le DES que l'on peut considérer comme la première génération devrai (ou bien la base) des algorithmes de cryptage. Il sera le centre de notre étude. Nous allons le développer et proposer quelques méthodes pour son implémentation.

Chapitre

02

ALGORITHME CRYPTOGRAPHIQUE DES

Chapitre2. Algorithme cryptographique DES

II.1. Généralité

Le développement considérable des réseaux de communication a favorisé l'expansion de nouveaux moyens de paiement à distance avec notamment la carte à puce, le e-commerce, et l'e-banking. Ces nouveaux types de communications nécessitent de plus en plus de moyens de transaction dits sûrs, pouvant résister aux diverses attaques crypto-analytiques. De cette lutte effrénée entre d'une part les cryptographes qui mettent en place des systèmes ou algorithmes cryptographiques, et d'autre part les crypto-analystes qui développent des techniques pour briser les systèmes de sécurité. L'algorithme DES est parmi les solutions les plus adoptées dans les systèmes de communication. Il est devenu populaire et largement utilisé aujourd'hui, car il est encore relativement sûr et rapide. [18]

II.2. Le data encryption standard (DES)

II.2.1. Définition

Le *Data Encryption Standard* (DES) est un algorithme de cryptographie qui a été sélectionné comme un standard pour la *Federal Information Processing Standard* (FIPS) pour les Etats-unien 1976, et qui a connu un succès international par la suite.

L'algorithme DES opère un chiffrement par bloc a clé symétrique, il prend une chaîne de caractères (ou nombres) d'une taille fixée à 64-bits du texte en clairet le transforme à l'aide d'opérations compliquées vers un texte crypté de la même taille. Cette transformation dépend d'une clé, et donc seulement celui qui connaît la clé est, à priori, capable de décrypter le message.

En apparence la clé est constituée de 64-bits ; en fait seulement 56 de ces 64-bits sont vraiment utilisés dans l'algorithme. Huit bits sont employés pour faire la vérification de parité. Il existe 16étapes de calcul identiques, dites **rondes** *(rounds).* Il y a aussi des permutations, l'une initiale*(PI)* et l'autre finale *(PF),* où *PF* défait l'opération réalisée par *PI* et vice-versa. Il semblerait que *PI* et *PF* ne sont pas de signification cryptographique, mais qui ont été inclus dans l'algorithme pour faciliter le chargement de données dans le matériel des années 70. Avant les itérations principales, le bloc est divisé en deux moitiés de 32-bits chacune, qui sont calculées de façon alternée. Ce croisement est connu comme le schéma de Feistel.

II.2.2. Le réseau de Feistel

Un réseau de Feistel est une construction utilisée dans les algorithmes de chiffrement par bloc, nommée d'après le cryptologue d'IBM, Horst Feistel. Elle a été utilisée pour la première fois dans Lucifer et DES. Cette structure offre plusieurs avantages, le chiffrement et le déchiffrement ont une architecture similaire voire identique dans certains cas.

L'implémentation matérielle est aussi plus facile avec un tel système même si les choses ont passablement changé depuis la fin des années 1970. Un réseau de Feistel repose sur des principes simples dont des permutations, des substitutions, des échanges de blocs de données et une fonction prenant en entrée une clé intermédiaire à chaque étage.

Il est vraisemblable que Feistel ne soit pas le seul inventeur de cette architecture. Durant une conférence, Don Copper Smith a laissé entendre que Bill Notz et Lynn Smith (de l'équipe d'IBM travaillant sur DES) avaient été en grande partie à l'origine du réseau de Feistel tel que nous le connaissons.

- Soit une fonction f qui prend comme argument un mot de n bits.

L'algorithme de chiffrement va procéder en chiffrant des blocs de 2n bits, qu'on partage en 2 parties de n bits chacune, comme la montre la figure (**II.1)** : les parties gauche (G) et droite (D).L'image du bloc (G_i, D_i) est le bloc (G_{i+1}, D_{i+1}), avec $G_{i+1}=D_i$ et $D_{i+1} = G_i$ XOR $f(D_i)$.

La partie droite n'a pas été transformée (juste envoyée à gauche). Il faut donc répéter le schéma de Feistel un certain nombre de fois (on parle de tours).

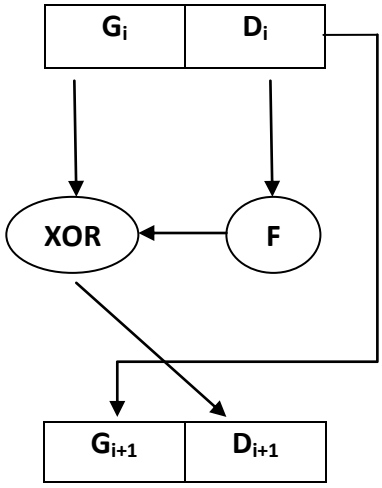

Fig. II.1 : Réseau de Feistel utilisant l'opérateur XOR.

Dans le cas de DES, le réseau de Feistel possède 16 tours, chacun avec une sous-clé. Ces différentes clés permettent d'améliorer la robustesse d'un algorithme face à la cryptanalyse.

L'avantage de ce type d'algorithmes est que chiffrement et déchiffrement sont structurellement identiques.

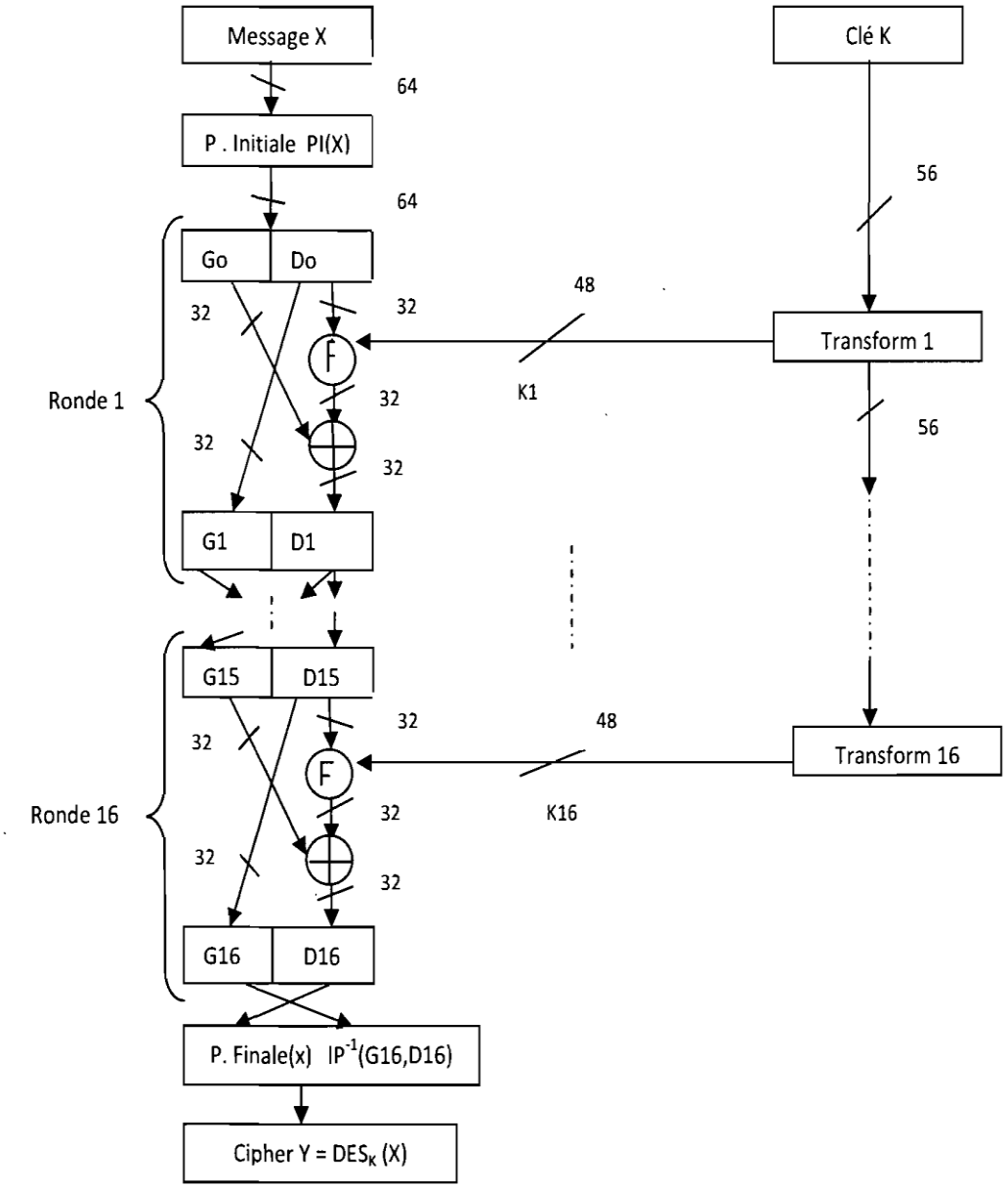

Fig. II.2 : Schéma Feistel de l'Algorithme DES. [19]

II.2.3. Les avantages de DES

L'algorithme DES contient les S-Boxes qui utilisent des ressources logiques, sur lesquelles repose la grande sécurité, étant non linéaires très efficaces pour diluer les informations. Par conséquent, nous pourrons appliquer la même approche sur les algorithmes successeurs de DES tels que: AES, 3DES, tant qu'ils contiennent les S-Boxes.

Quant à la performance, les algorithmes symétriques, en général, nécessitent une capacité de calcul moins intensive que les algorithmes asymétriques [20]. Cela occasionne une rapidité décryptage et de décryptage atteignant des centaines ou des milliers de fois supérieure à celle des algorithmes à clé publique.

II.2.4. Les applications du DES

Le DES est présenté dans bien des applications associées à des opérations électroniques, comme par exemple la sécurité du courrier électronique, le transfert sécurisé de fichiers (notamment entre banques), ainsi que le paiement électronique.

II.2.5. Le fonctionnement du DES

Il s'agit d'un système de chiffrement symétrique par blocs de 64 bits, dont 8 bits (un octet) servent de test de parité (pour vérifier l'intégrité de la clé). Chaque bit de parité de la clé (1 tous les 8 bits) sert à tester un des octets de la clé par parité impaire, c'est-à-dire que chacun des bits de parité est ajusté de façon à avoir un nombre impair de '1' dans l'octet à qui il appartient. La clé possède donc une longueur « utile » de 56 bits, ce qui signifie que seuls 56 bits servent réellement dans l'algorithme.

L'algorithme consiste à effectuer des combinaisons, des substitutions et des permutations entre le texte à chiffrer et la clé, en faisant en sorte que les opérations puissent se faire dans les deux sens (pour le déchiffrement). La combinaison entre substitutions et permutations est appelée **code produit**.

La clé est codée sur 64 bits et formée de 16 blocs de 4 bits, généralement notés *k1* à *k16*. Etant donné que « seuls » 56 bits servent effectivement à chiffrer, il peut exister 2^{56} clés différentes.

L'algorithme DES utilise seulement **3** types d'opérations différentes : des permutations, des rotations et des substitutions. Ceci est un point fort du DES, il n'utilise pas de lourdes opérations comme des additions ou des multiplications qui demanderaient beaucoup de ressources de calcul. Comme la montre **Fig. II.3** et **Fig. II.4**, les blocs à chiffrer sont d'abord sujets à une permutation initiale IP, puis on leur applique 16 itérations de chiffrement avant de les faire passer finalement dans une permutation initiale inverse IP^{-1}.

Le Key Schedule transforme la clé de 56 bits en 16 clés partielles ou 'sous-clés' de 48 bits chacune.

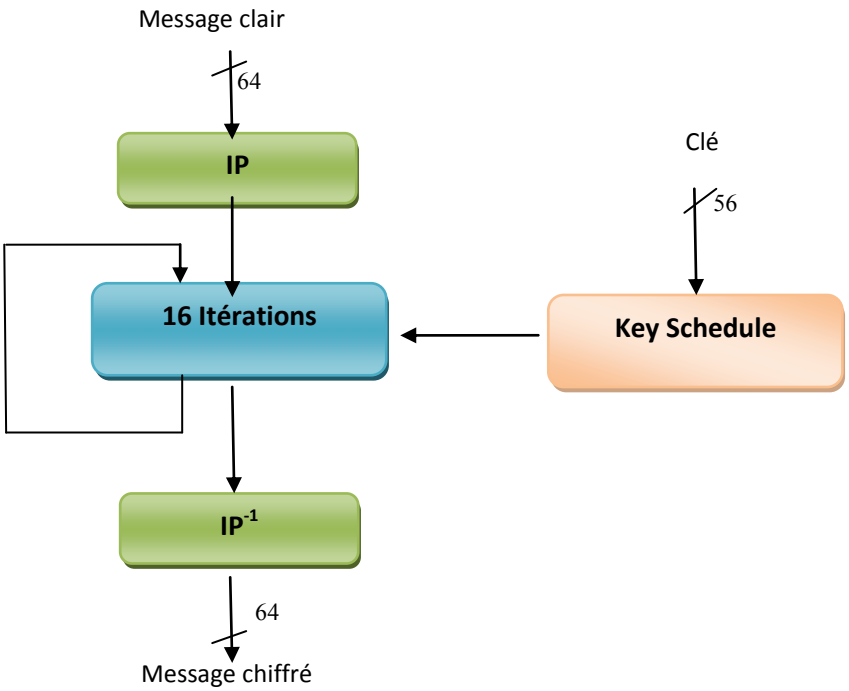

Fig. II.3 : Vue d'ensemble du DES.

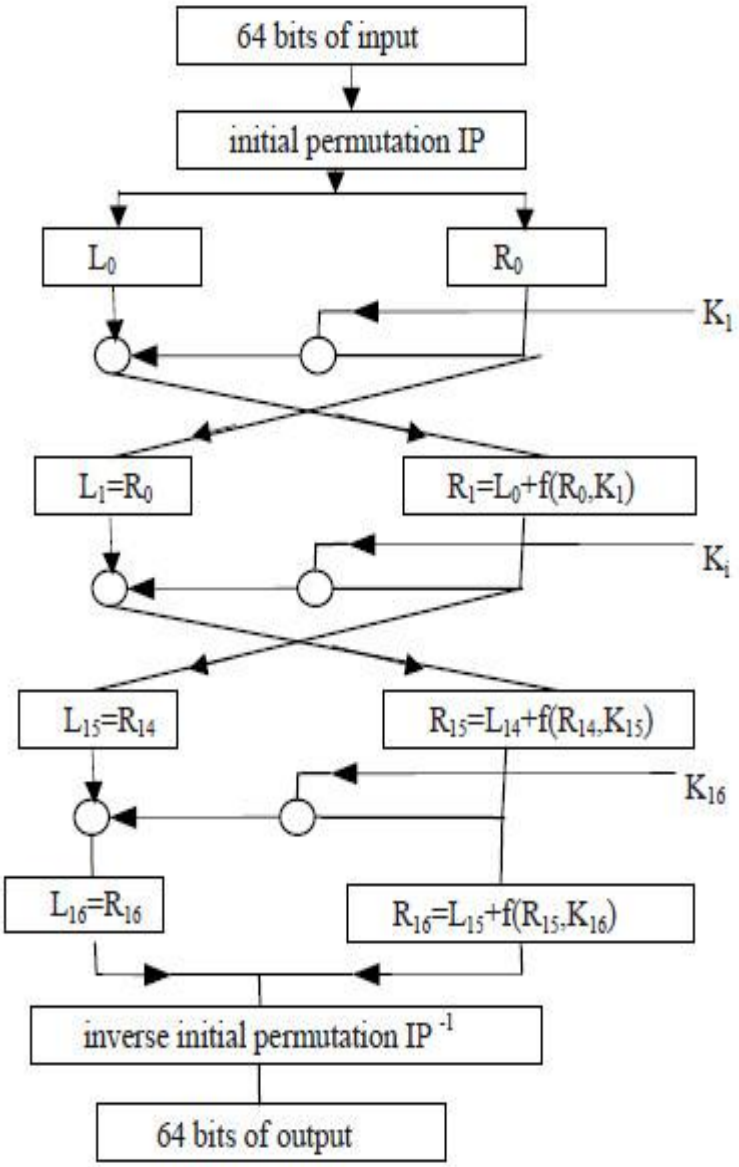

Fig. II.4 : Le fonctionnement de DES. [21]

II.2.6. Algorithme de DES [22]

Les grandes lignes de l'algorithme DES, sont décrites comme suit:

- Fractionnement du texte en blocs de 64 bits (8 octets).
- Permutation initiale des blocs.
- Découpage des blocs en deux parties: gauche et droite, nommées G et Dou (Let R).
- Etapes de permutation et de substitution répétées 16 fois (appelées rondes).
- Recollement des parties gauche et droite puis permutation initiale inverse,

Ou bien (permutation final).

Les étapes du processus de l'algorithme DES :

A. Fractionnement du texte en blocs de 64 bits (8 octets) :

Dans un premier temps le message en clair est découpé en blocs de 64 bits.

B. Permutation initiale :

Chaque bit d'un bloc subit une permutation selon l'arrangement du tableau ci-contre, pouvant être représentée par la matrice de permutation initiale (notée PI) suivante:

Tableau. II.1 : Matrice de permutation initiale de DES.

PI	**58**	**50**	**42**	**34**	**26**	**18**	**10**	**2**
	60	**52**	**44**	**36**	**38**	**20**	**12**	**4**
	62	**54**	**46**	**38**	**30**	**22**	**14**	**6**
	64	**56**	**48**	**40**	**32**	**24**	**16**	**8**
	57	**49**	**41**	**33**	**25**	**17**	**9**	**1**
	59	**51**	**43**	**35**	**27**	**19**	**11**	**3**
	61	**53**	**45**	**37**	**29**	**21**	**13**	**5**
	63	**55**	**47**	**39**	**31**	**23**	**15**	**7**

Cette matrice de permutation indique, en parcourant la matrice de gauche à droite puis de haut en bas, que le 58ème bit du bloc de texte de 64 bits se retrouve en première position, le 50ème en seconde position et ainsi de suite.

C. Scindement en blocs de 32 bits :

Une fois la permutation initiale réalisée, le bloc de 64 bits est scindé en deux blocs de 32 bits, notés respectivement G et D (pour gauche et droite, la notation anglo-saxonne étant L et R pour (Left and Right). On note G_0 et D_0 l'état initial de ces deux blocs:

Tableau. II.2 : Blocs G_0 de 32 bits de DES.

G_O	**58**	**50**	**42**	**34**	**26**	**18**	**10**	**2**
	60	**52**	**44**	**36**	**38**	**20**	**12**	**4**
	62	**54**	**46**	**38**	**30**	**22**	**14**	**6**
	64	**56**	**48**	**40**	**32**	**24**	**16**	**8**

Tableau. II.3 : Blocs D_0 de 32 bits de DES.

D_0	57	49	41	33	25	17	9	1
	59	51	43	35	27	19	11	3
	61	53	45	37	29	21	13	5
	63	55	47	39	31	23	15	7

On remarque queG_0contient tous les bits pairs du message initial et D_0 tous les bits impairs.

D. Rondes :

Les blocs G_0et D_0sont soumis à un ensemble de transformations appelées rondes. Une ronde est elle-même composée de plusieurs étapes :

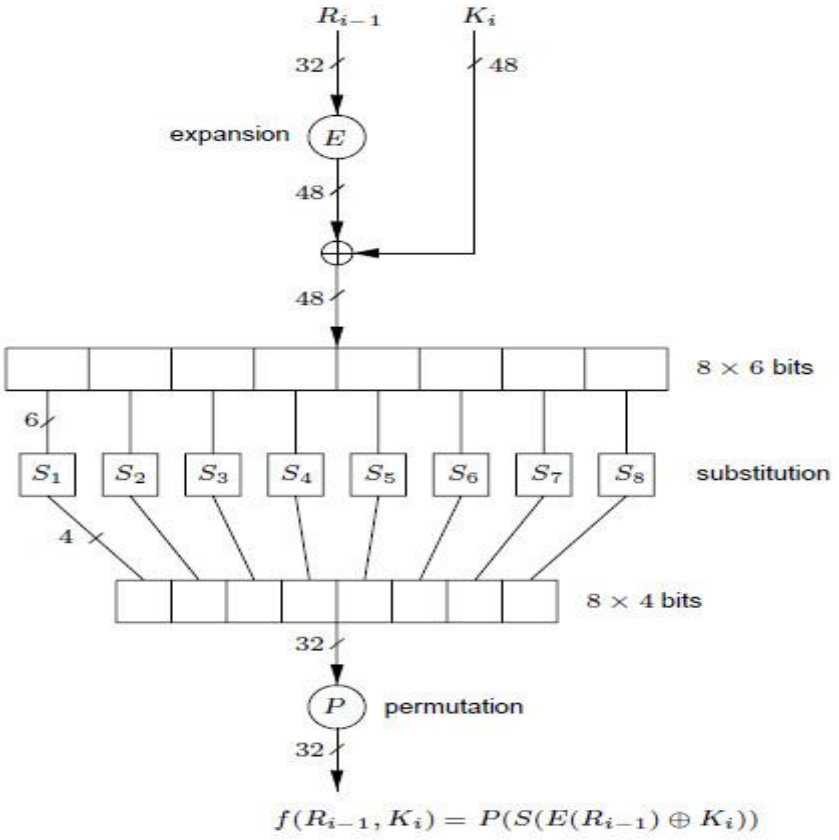

Fig. II.5 : Schéma détaillé du ronde1.

Les 64 bits de texte sont initialement séparés en 2 parties égales de 32 bits : une partie gauche $L_0(G_0)$ et une partie droite $R_0(D_0)$. Ces 2 parties sont ensuite propagées à travers les 16 rounds de la façon suivante : $\begin{aligned} L_i &= R_{i-1} \\ R_i &= L_{i-1} \oplus f(R_{i-1}, K_i) \end{aligned}$

où le symbole ⊕ définit l'opération booléenne XOR. A la fin des 16 rounds, les 2 parties L_{16} et R_{16} sont remises ensemble.

1. Fonction d'expansion :

Les 32 bits du bloc Do sont étendus à 48 bits grâce à une table (matrice) appelé *table d'expansion* (notée E), dans laquelle les 48 bits sont mélangés et 16 d'entre eux sont dupliqués:

Tableau. II.4 : La table de la fonction d'expansion.

E	**32**	**1**	**2**	**3**	**4**	**5**
	4	**5**	**6**	**7**	**8**	**9**
	8	**9**	**10**	**11**	**12**	**13**
	12	**13**	**14**	**15**	**16**	**17**
	16	**17**	**18**	**19**	**20**	**21**
	20	**21**	**22**	**23**	**24**	**25**
	24	**25**	**26**	**27**	**28**	**29**
	28	**29**	**30**	**31**	**32**	**1**

Ainsi, le dernier bit de D_0 (c'est-à-dire le 7ème bit du bloc d'origine) devient le premier, le premier devient le second, ...
De plus, les bits 1,4,5,8,9,12,13,16,17,20,21,24,25,28 et 29 de D_0 (respectivement 57, 33, 25, 1,59, 35,27, 3, 61,37, 29, 5, 63, 39, 31 et 7 du bloc d'origine) sont dupliqués et disséminés dans la matrice.

2. OU exclusif (XOR) avec la clé :

La matrice résultante de 48 bits est appelée D'_0 ou bien E [D_0]. L'algorithme DES procède ensuite à un OU exclusif entre la première clé K_1 et E [D_0]. Le résultat de ce OU exclusif est une matrice de 48 bits que nous appellerons D''_0 le résultat de cette opération.

3. Fonction de substitution (S-Box) :

D''_0 est découpée ensuite en 8 blocs de 6 bits, noté D''_{0i}. Chacun de ces blocs passe par des boîtes de substitution(S-boxes), notées généralement S_i. Les premier et dernier bits de chaque D_{0i} déterminent la ligne de la fonction de substitution, les autres bits déterminent la colonne. Grâce à cela la fonction de substitution choisit une valeur codée sur 4 bits (de 0 à 15). Voici les fonctions de substitution, représentée par une matrice de 4 par 16 :

Tableau. II.5 : Les matrices de la fonction de substitution.

		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
S_1	0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
	1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
	2	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
	3	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
S_2	0	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
	1	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
	2	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
	3	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9

		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
S_3	0	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
	1	13	7	0	9	3	4	5	10	2	8	5	14	12	11	15	1
	2	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
	3	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12

		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
S_4	0	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
	1	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
	2	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
	3	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14

		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
S_5	0	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
	1	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
	2	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
	3	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3

S_6		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	0	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
	1	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
	2	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
	3	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13

S_7		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	0	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
	1	13	0	11	74	9	1	10	14	3	5	12	2	15	8	6	6
	2	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
	3	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12

S_8		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	0	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
	1	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
	2	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
	3	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

Soit D_{0i} égal à 011010. Les premiers et derniers bits donnent 00, c'est-à-dire 0 en décimal. Les bits 2, 3,4 et 5 donnent 1101, soit 13 en décimal. Le résultat de la fonction de sélection est donc la valeur située à la ligne n°0, dans la colonne n°13. Il s'agit de la valeur 9, soit en binaire 1001. Chacun des 8 blocs de 6 bits est passé dans la fonction de sélection correspondante, ce qui donne en sortie 8 valeurs de 4 bits chacune. Chaque bloc de 6 bits est ainsi substitué en un bloc de 4 bits. Ces bits sont regroupés pour former un bloc de 32 bits.

4. Permutation

Le bloc de 32 bits obtenu est enfin soumis à une permutation P dont voici la table:

Tableau. II.6 : La table de Permutation P.

P							
16	7	20	21	29	12	28	17
1	15	23	26	5	18	31	10
2	8	24	14	32	3	27	9
19	13	30	6	22	11	4	25

5. OU Exclusif

L'ensemble de ces résultats en sortie de P est soumis à un OU Exclusif avec le G_0 de départ pour donner D_1, tandis que le D_0 initial donne G_1.

6. Itération

L'ensemble des étapes précédentes (rondes) est réitéré 16 fois.

E. Permutation initiale inverse(PF)

A la fin des itérations, les deux blocs G_{I6} et D_{I6} recollés pour reformer un seul bloc de 64 bits puis subit la permutation initiale inverse selon l'arrangement du tableau ci-contre. On obtient alors le bloc initial chiffré.

Tableau. II.7 : La table de la permutation initiale inverse.

PI^{-1}	**40**	**8**	**48**	**16**	**56**	**24**	**64**	**32**
	39	**7**	**47**	**15**	**55**	**23**	**63**	**31**
	38	**6**	**46**	**14**	**54**	**22**	**62**	**30**
	37	**5**	**45**	**13**	**53**	**21**	**61**	**29**
	36	**4**	**44**	**12**	**52**	**20**	**60**	**28**
	35	**3**	**43**	**11**	**51**	**19**	**59**	**27**
	34	**2**	**42**	**10**	**50**	**18**	**58**	**26**
	33	**1**	**41**	**9**	**49**	**17**	**57**	**25**

Le résultat en sortie est un texte codé de 64 bits.

F. Reconstruction du message chiffré

Tous les blocs sont collés bout à bout pour obtenir le message chiffré.

G. Génération des clés

Nous allons décrire l'algorithme qui permet de générer à partir d'une clef de 64 bits, 8 clefs diversifiées de 48 bits chacune servant dans l'algorithme du DES.

D'abord 56 bits de la clé sont sélectionnés des 64 bits du départ, à travers le choix permuté 1(PC 1) ; les 8 bits restant sont ignorés, ou alors utilisés comme bits de parité. Les 56 bits sont alors divisés en deux parties de 28 bits chacune. A partir de ce moment, chaque moitié est traitée séparément. Dans des itérations (rondes) successives, les deux parties de la clé sont rotationnels d'un ou de deux bits (spécifique à chaque itération) ; et alors une sous-clé de 48bits est choisie par le PC2, 24bits de la moitié gauche et 24bits de

la moitié droite. Les rotations à gauche signifient qu'un ensemble de bits est utilisé à chaque sous clé, chaque bit étant utilisé dans environ 14 des 16 sous-clés.

L'algorithme ci-dessous montre comment obtenir, à partir d'une clé de 64 bits (composé de 64caractères alphanumériques quelconques), 8 clés diversifiées de 48 bits chacune servant dans l'algorithme du DES :

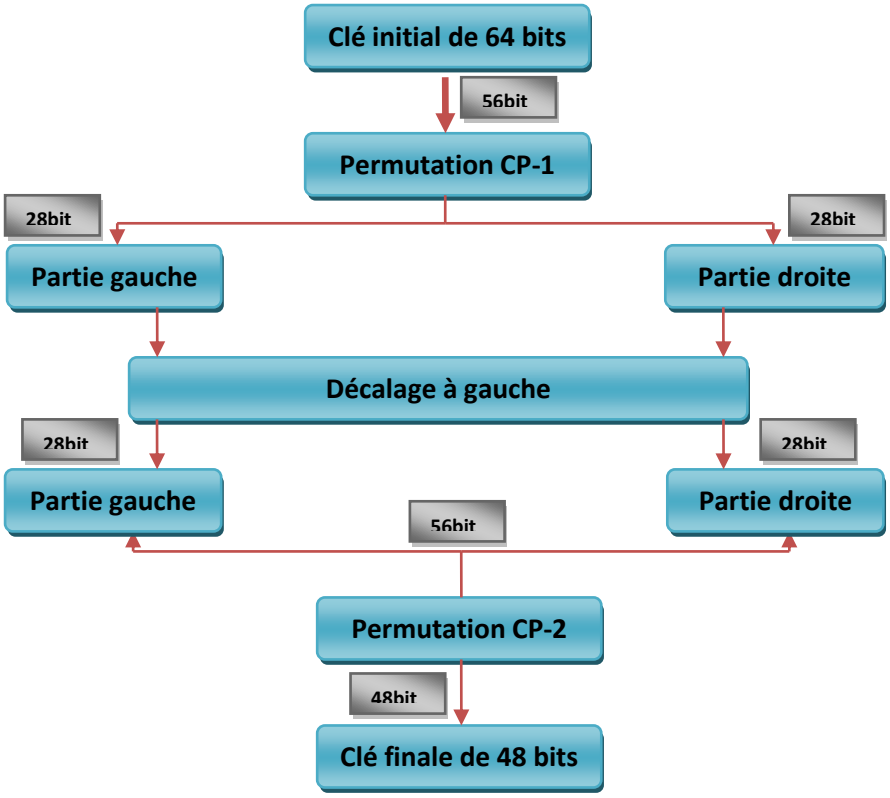

Fig. II.6 : Schéma de génération de clé.

Dans un premier temps les bits de parité de la clé sont éliminés afin d'obtenir une clé d'une longueur utile de 56 bit La première étape consiste en une permutation notée CP1dont la matrice est présentée Ci-dessous:

Tableau. II.8 : La matrice de CP1.

CP1	**57**	**49**	**41**	**33**	**25**	**17**	**9**	**1**	**58**	**50**	**42**	**34**	**26**	**18**
	10	**2**	**59**	**51**	**43**	**35**	**27**	**19**	**11**	**3**	**60**	**52**	**44**	**36**
	63	**55**	**47**	**39**	**31**	**23**	**15**	**7**	**62**	**54**	**46**	**38**	**30**	**22**
	14	**6**	**61**	**53**	**45**	**37**	**29**	**21**	**13**	**5**	**28**	**20**	**12**	**4**

Cette matrice peut en fait s'écrire sous la forme de deux matrices G_i et D_i (pour gauche et droite) composées chacune de 28 bits.

Tableau. II.9 : La matrice G_i.

G_i	57	49	41	33	25	17	9
	1	58	50	42	34	26	18
	10	2	59	51	43	35	27
	19	11	3	60	52	44	36

Tableau. II.10 : La matrice D_i.

D_i	63	55	47	39	31	23	15
	7	62	54	46	38	30	22
	14	6	61	53	45	37	29
	21	13	5	28	20	12	4

On note G_0 et D_0 le résultat de cette première permutation.

Ces deux blocs subissent ensuite une rotation à gauche, de telles façons que les bits en seconde position prennent la première position, ceux en troisième position prennent la seconde, ...

Les bits en première position passent en dernière position. Les 2 blocs de 28 bits sont· ensuite regroupés en un bloc de 56 bits. Celui-ci passe par une permutation, notée CP2, fournissant en sortie un bloc de 48 bits, représentant la clé K.

Tableau. II.11 : La matrice de CP2.

CP2	14	17	11	24	1	5	3	28	15	6	21	10
	23	19	12	4	26	8	16	7	27	20	13	2
	41	52	31	37	47	55	30	40	51	45	33	48
	44	49	39	56	34	53	46	42	50	36	29	32

Des itérations de l'algorithme permettent de donner les 16 clés K1 à K16 utilisées dans l'algorithme du DES.

Tableau. II.12 : Les numéros de décalage bit dans la clé par ronde.

Ronde	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
N°	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

II.3. Algorithme TDES (3 DES)

En 1990 Eli Biham et Adi Shamir ont mis au point la cryptanalyse différentielle qui recherche des paires de texte en clair et des paires de texte chiffrées. Cette méthode marche jusqu'à un nombre de rondes inférieur à 15, or un nombre de 16 rondes sont présentes dans l'algorithme présenté ci-dessus.

D'autre part, même si une clé de 56 bits donne un nombre énorme de possibilités, de nombreux processeurs permettent de calculer plus de 10 clés par seconde, voir la **Fig. II.7)** ainsi, utilisés parallèlement sur un très grand nombre de machines, il devient possible pour un grand organisme (un Etat par exemple) de trouver la bonne clé...Une solution à court terme consiste à chaîner trois chiffrement DES à l'aide de deux clés de 56 bits (ce qui esquivait à une clé de 112 bits). Ce procédé est appelé **Triple DES**, noté TDES (parfois 3DES ou 3-DES). [23]

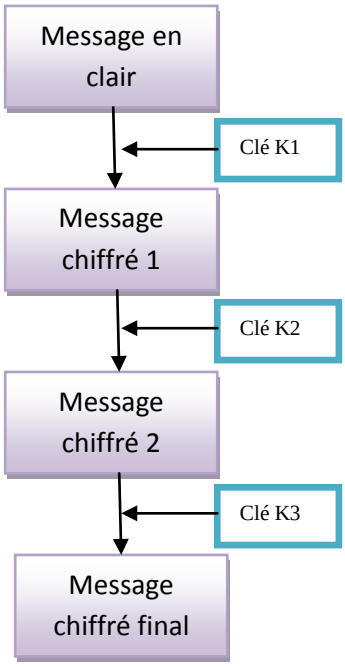

Fig. II.7 : Schéma de TDES.

Le **TDES** permet d'augmenter significativement la sécurité du DES, toutefois il a l'inconvénient majeur de demander également plus de ressources pour le chiffrement et le déchiffrement.

On distingue habituellement plusieurs types de chiffrement triple DES :

- DES-EEE3 : 3 chiffrements DES avec 3 clés différentes.
- DES-EDE3 : une clé différente pour chacune des 3 opérations DES (chiffrement, déchiffrement, chiffrement).

- DES-EEE2 et DES-EDE2 : une clé différente pour la seconde opération (déchiffrement).

II.4. Le déchiffrement

Nous possédons maintenant toutes les informations nécessaires pour chiffrer des données avec le DES. Il nous faut maintenant découvrir comment récupérer ces données chiffrées sous leur forme claire.

Le DES étant un algorithme symétrique, c'est la même clé qui sera utilisée pour déchiffrer les données. De plus, moyennant quelques modifications, la structure de chiffrement étudiée ci-dessus sera celle utilisée pour le déchiffrement. La configuration et l'ordonnancement des Encrypte rounds ne seront pas modifiées. Par contre, les Key rounds changeront légèrement. Les rotations de bits vers la gauche seront simplement remplacées par des rotations vers la droite, le reste demeurant inchangé. La structure de déchiffrement du DES pourra donc être vue comme celle représentée à la figure 6, à condition de remplacer les rotations vers la gauche des Key rounds en rotations vers la droite. Le texte chiffré sera introduit dans la boîte de permutation IP et le texte en clair sera produit en sortie de la permutation IP^{-1}.

Cette propriété du DES lui accorde encore un nouvel avantage : la facilité avec laquelle on peut passer d'une unité de chiffrement à une unité de déchiffrement. Il n'y a donc pas besoin de construire deux outils différents pour le chiffrement et le déchiffrement. En général, la même unité pourra opérer dans les deux sens, et elle sera conçue sans nécessiter l'ajout substantiel de ressources. [23]

II.5. Conclusion

Dans le deuxième chapitre, nous avons donnés une présentation le principe de fonctionnement l'algorithme DES.

Le DES présente donc une facilité de conception, mais aussi d'exécution. Il n'y a en outre pas de différence entre le chiffrement et le déchiffrement (seul l'ordre dans lequel les bits de la clé sont présentés aux fonctions f diffère). C'est pourquoi c'est un standard adopté par un grand nombre d'institutions.

Chapitre 03

IMPLEMENTATION SUR FPGA DE L'ARCHITECTURE DES

Chapitre3. Implémentation sur FPGA de l'architecture DES

III.1 Introduction

Le but de ce projet est d'optimiser une implémentation VHDL de l'algorithme cryptographique DES sur une plate-forme reconfigurable basée sur FPGA. Deux architectures sont étudiées pour ce projet : la première est le plus rapide possible et la deuxième consomme moins d'espace que la première architecture sur le FPGA. Le sens de la vitesse pour ce projet est le débit (nombre de bits traités par seconde) et la signification de la surface est le nombre de CLB (Bloque logique configurable).

DES (Data Encryption Standard) peut être considéré comme une technologie de codage de données standard qui est largement utilisée dans le domaine du cryptage des données avec de bonnes performances de sécurité. L'algorithme DES est largement utilisé dans certaines régions telles que POS (point de vente), guichet automatique (caisse automatique), cartes magnétiques et cartes à puce, Stations de péage routier, etc., afin de protéger l'information contre toute partie indésirable.

L'avantage du cryptage et décryptage DES est la haute vitesse et facilité de réalisation des algorithmes. Généralement, il est très utilisé pour le cryptage des signaux sonores. Parmi ces inconvénients on peut citer sa longueur de clé courte qui affecte sa robustesse face à la cryptanalyse.

III.2. Les circuits FPGAs (Field Programmable Gâte Array)

Un FPGA est un composant électronique constitué de million voir milliers de transistors connectés ensembles pour réaliser des fonctions logiques. Des fonctions logiques simples peuvent être réalisées telles que des additions ou soustractions tout comme des fonctions complexes peuvent être réalisées telles que le filtrage numérique du signal ou détection d'erreur et correction. Aérospatial, aviation, automobile, radar, missile, ordinateur… ne sont que quelque exemples des domaines ayant recours aux FPGA. [24]

La plupart des grands FPGA modernes sont fondés sur des cellules SRAM aussi bien pour le routage du circuit que pour les blocs logiques à interconnecter.

Un bloc logique est de manière générale constitué d'une table de correspondance (LUT ou Look-Up-Table) et d'une bascule (Flip-Flop en anglais). La LUT sert à implémenter des équations logiques ayant généralement 4 à 6 entrées et une sortie [25]. Elle peut toutefois être considérée comme une petite mémoire, un multiplexeur ou un registre à décalage. Le registre permet de mémoriser un état (machine séquentielle) ou de synchroniser un signal (pipeline).

III.2.1. Avantages de l'utilisation des FPGA

Les FPGAs ont connu d'importantes évolutions architecturales. Résultant essentiellement de l'augmentation des capacités d'intégration. Les principaux atouts de la technologie FPGA sont :

- ❖ Exécution très rapide.
- ❖ Facilité et flexibilité d'utilisation.
- ❖ Adapté au prototypage rapide.
- ❖ Fabrication économique et maintenance à long terme.

III.2.2. Architecture des FPGAs

C'est une famille de puces électroniques introduite par la société Xilinx. Les FPGAs sont les premières architectures reconfigurables à avoir été proposées. Ils ont l'avantage d'être reconfigurables à souhait. Ils comprennent des blocs configurables qui permettent de générer des fonctions logiques combinatoires ou séquentielles sur 1 bit. Ainsi, en associant plusieurs éléments configurables de 1 bit par le biais d'interconnexions tout aussi programmables **(Fig. III.1)**, l'utilisateur est capable de générer n'importe quelle fonction logique.

Si le nombre d'unités configurables est suffisamment important, il est possible, en associant ces blocs, de recréer un système entier. C'est pour cette raison que ces composants sont utilisés pour faire des prototypes de circuits, avant de les envoyer chez le fondeur de silicium. Son développement s'est accéléré sous la double pression du «Time-To-Market» et de «First Time-Right» (minimiser d'une part le temps de développement et de mise sur le marché des circuits digitaux et d'autre part éviter toute défaillance dans leur conception).

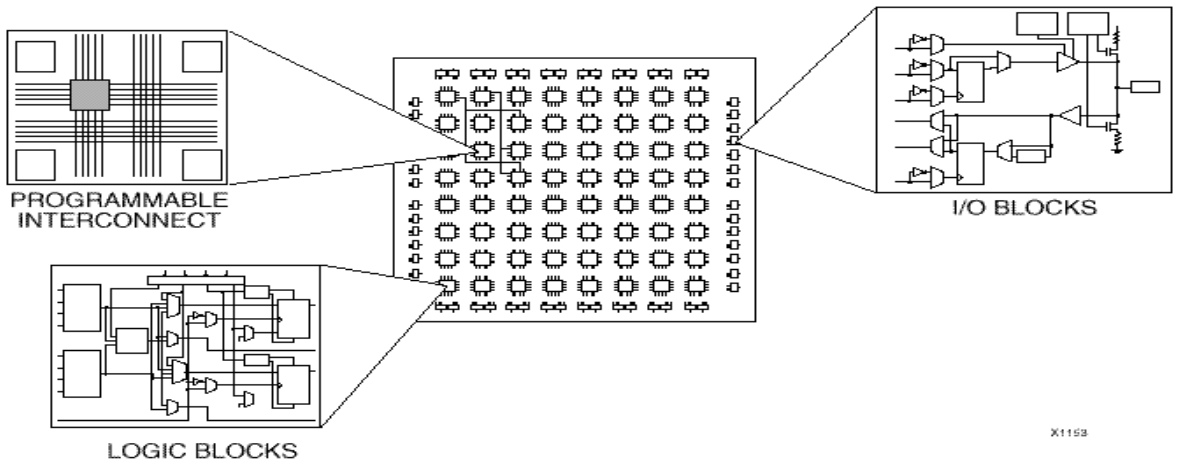

Fig. III.1 : Les différents éléments d'un FPGA. [26]

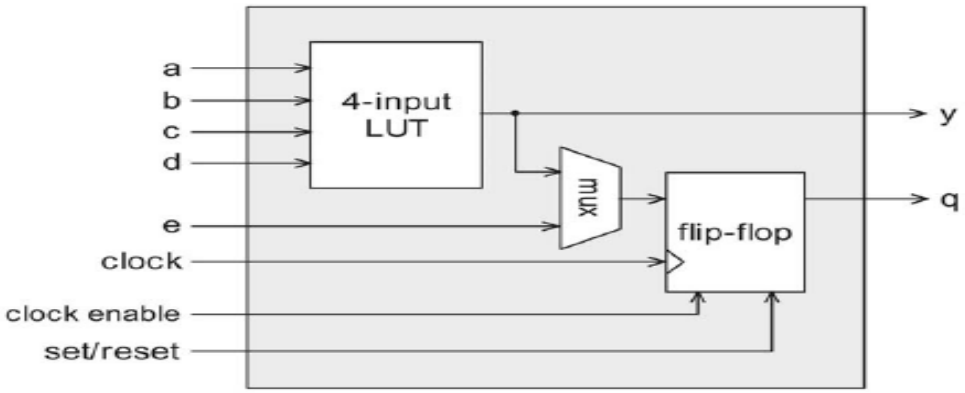

Fig. III.2 : Un LUT (Look-Up-Table).

Les circuits FPGA du fabricant Xilinx utilisent deux types de cellules de base :

- Les cellules d'entrées/sorties appelés 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.

III.3. Avantage de l'implémentation matérielle

Dans un modèle d'exécution logicielle, le traitement est séquentiel et exécuté par un processeur. En effet, un CPU (Central Processing Unit) exécute une tâche (ou une opération) à la fois. Toute application est découpée en tâches unitaires exécutées les unes à la suite des autres. Le passage d'une tâche à l'autre nécessite une sauvegarde de contexte (Context Switch) qui permet de conserver la cohérence globale de l'application, et donner une apparence d'exécution parallèle des tâches. Mais ce modèle nécessite des processeurs de plus en plus rapides pour répondre à la complexité algorithmique des applications. Or les fréquences de fonctionnement des processeurs ne sauraient être augmentées indéfiniment à cause d'une part des limites technologiques, et d'autre part d'une augmentation de la consommation qui n'est pas souhaitable dans les systèmes embarqués.

Par contre dans un ASIC ou un FPGA, les applications (algorithmes, fonctions, etc...) décrites dans un langage de description de circuits (VHDL, Verilog, etc...) sont implémentées matériellement; ceci apporte un gain de performance supérieur à celui des processeurs grâce à l'implémentation spatiale (parallélisme) de la tâche [27]. En effet dans ce type d'implémentation matérielle, seul le temps de propagation des signaux de l'entrée à la sortie d'un système fixe sa limite maximum en fréquence.

En outre, la reconfigurabilité dynamique de certains FPGAs permet de faire évoluer l'architecture pour s'adapter au traitement, contrairement aux ASICs. Il est aujourd'hui envisageable d'implémenter et d'exécuter séquentiellement sur ce type de composants

reconfigurables plusieurs algorithmes, et de profiter à la fois de leur reconfigurabilité (flexibilité) et de leur performance. Pour cela une architecture reconfigurable inclut généralement deux principales parties, une partie matérielle et une partie logicielle. La partie logicielle comprend généralement un processeur (à usage générique - GPP ou orienté traitement du signal - DSP*)* chargé du contrôle et de la gestion de la reconfiguration de la partie matérielle, ainsi que de l'exécution des tâches encore dévolues au logiciel. Elle utilise la partie matérielle (un ou plusieurs FPGAs) comme accélérateur matériel, en y implémentant les parties les plus critiques des traitements.

Le tableau comparatif ci-dessous (**Tableau. III.1**) **:** Permet de situer les FPGAs parmi les principaux types d'architectures et leurs caractéristiques. [28]

Tableau. III.1 : Les différents types d'architecture et leur caractéristique.

	Performance	**Cout**	**Consommation**	**Flexibilité**	**Programmabilité**
GPP	faible	faible	moyen	élevé	élevé
DSP	moyen	moyen	moyen	moyen	élevé
ASIC	élevé	élevé	faible	faible	faible
FPGA	moyen	moyen	élevé	élevé	moyen

III.4. Optimisation logicielle de DES

L'algorithme DES peut être modifié dans le nombre de voies qui n'auront aucune influence sur la fonction calculée selon le programme. Dans chaque cryptage DES il y a 16 paires produit transformation/ bloc-transformation. La transformation de bloc est simplement l'échange de R32 bits avec L32 bits. Pour éviter la transformation de bloc à la fin de chaque transformation de produit, il y a deux transformations de produit différentes: l'une fonctionne de la manière habituelle et l'autre applique les opérations sur L comme si elle était R et verse-versa [29]. L'utilisation de ces deux transformations de produit élimine alternativement le besoin d'une transformation de bloc entre des transformations de produit. Le seul problème avec cet arrangement est qu'après la dernière transformation de produit, les blocs de L et R sont échangés .Ce renversement est incorporé dans la finale IP-1 parce qu'un échange et une permutation sont juste une permutation [30]. Soit € représentant l'expansion E, σ représente la fonction S et π représente la permutation P. K_i étant la nième sous-clé, Ri étant la nième valeur de droite, et Li est la nième valeur de gauche.

Par définition :

$$L_i=R_i-1$$

$$R_i=L_i-1+f\ R_i-1$$

$$fi\ R_i-1=\pi\sigma\ (K_i+\epsilon R_i-1)$$

Où fi est la transformation *f* qui utilise la sous-clé Ki.

Après deux rondes:

$$L_i+1=L_i-1+fi\ R_i-1$$

$$R_i+1=R_i-1+fi+1(L_i-1+fi\ R_i-1)$$

Puisqu'il y a 16 rondes dans DES, 8 double-rondes de la forme suivante peuvent être utilisées au lieu de cela:

$$L_i+1=L_i-1+fi\ R_i-1$$

$$R_i+1=R_i-1+fi+1\ L_i+1$$

Remarquons qu'il n'existe aucune valeur intermédiaire entre Ri et Li.

Après 1 double-ronde:

$$L_i+1=L_i-1+fi\ R_i-1$$

$$R_i+1=R_i-1+fi+1(L_i-1+\ fi\ R_i-1)$$

Qui est le même comme auparavant.

En réalité, l'échange a été construit dans l'itération, parce que R et L peuvent jouer des rôles réciproques.

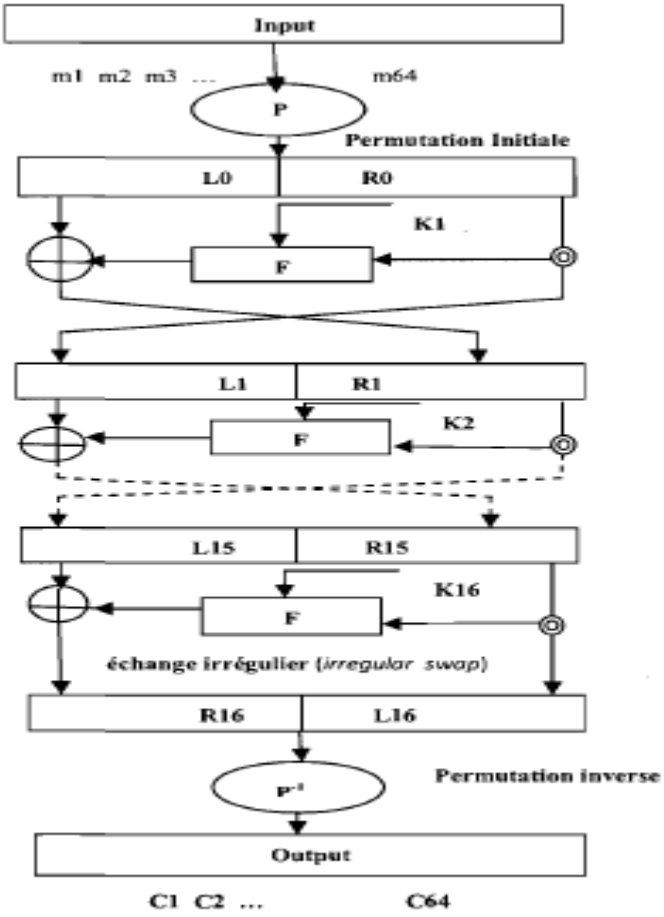

Fig. III.3 : DES avec l'échange irrégulier *(irregular swap).*

III.5. Implémentation de DES

L'implémentation matérielle de l'algorithme DES (Data Encryption Standard) sur une plate-forme reconfigurable basée sur FPGA (Field Programmable Gâte Arrays) Spartan-3EXC3S100E package TQ144, utilisant Xilinx ISE comme outil de synthèse, ModelSim pour la simulation, et Xilinx Chip Scope Pro comme analyseur logique pour le débogage.

L'architecture du design a été décrite dans le langage VHDL. Dans le but de réduire le chemin critique du design associé, nous avons utilisé une approche pipeline qui nous permet d'obtenir des performances compétitives. Cette approche sera étudiée avec son efficacité sur le DES en mettant l'accent sur les S-Boxes à haut débit et reconfigurables.

III.5.1 Les outils

Il est essentiel de détailler les différents outils (et leur version) utilisés dans ce projet parce que chacun a ses caractéristiques (fonctions) spécifiques et d'autres flux.

1. Software

A. ModelSim :

ModelSim 6.5 est un outil pour simuler un fichier VHDL et est utilisé pour vérifier le comportement du code. Un fichier banc de test (test bench) est très utile pour vérifier immédiatement et automatiquement si la conception fonctionne comme prévu.

B. Xilinx ISE :

ISE 14.5 *(Integrated Software Environment)* est un panel d'outils développé et commercialisé par la société Xilinx [31]. Ce panel permet de réaliser toutes les phases du flot de conception d'applications sur des composants reconfigurables (FPGA) de la société Xilinx. Il est constitué d'une dizaine d'outils, exemples: *Project navigator*: outil de description du projet (spécification du système à concevoir en VDHL, schématique, machines d'états), synthèse, placement, routage. *FloorPlanner*: outil de visualisation et de localisation des éléments utilisés du composant, ainsi que des communications réalisées. *FPGA editor* : outil de visualisation du placement et routage, disposant d'un placeur routeur intégré. *IMPACT:* utilisé pour télécharger directement le fichier. *Bitstream*au FPGA.

La dernière étape est de placer et d'acheminer le code synthétisé avec Xilinx ISE. ISE signifie environnement de logiciel intégré. Il fournit beaucoup d'outils à accomplir Chaque étape du processus de conception à partir de l'entrée de conception pour télécharger la conception au circuit FPGA. Un synthétiseur (XST) fait partie de ces outils. L'un des autres outils d'ISE s'appelle Impact est utilisé pour télécharger le bit Généré directement dans le FPGA.

2. Hardware

A. Spartan 3E :

Le FPGA que nous avons utilisé dans notre projet est un paquet Spartan3E XC3S100E PackageTQ144

Voici les caractéristiques principales de notre FPGA :

- Nombre de tranches: 960.
- Nombre de Flip Flop: 1920.
- Nombre de LUT 4 entrées: 1920.
- Nombre d'IO: 108.
- Nombre de GCLK: 24.

III.5.2 Flux de conception

Pour la conception de matériel complexe, avec DES qui est sans aucun doute, le développement, la simulation, la synthèse et des outils de vérification sont nécessaires. Le flot de conception générale de cette implémentation de DES sur FPGA se base généralement sur les éléments sont résumées dans le diagramme de conception ci-dessous, en utilisant les outils que nous avons mentionné ci-dessus [32] :

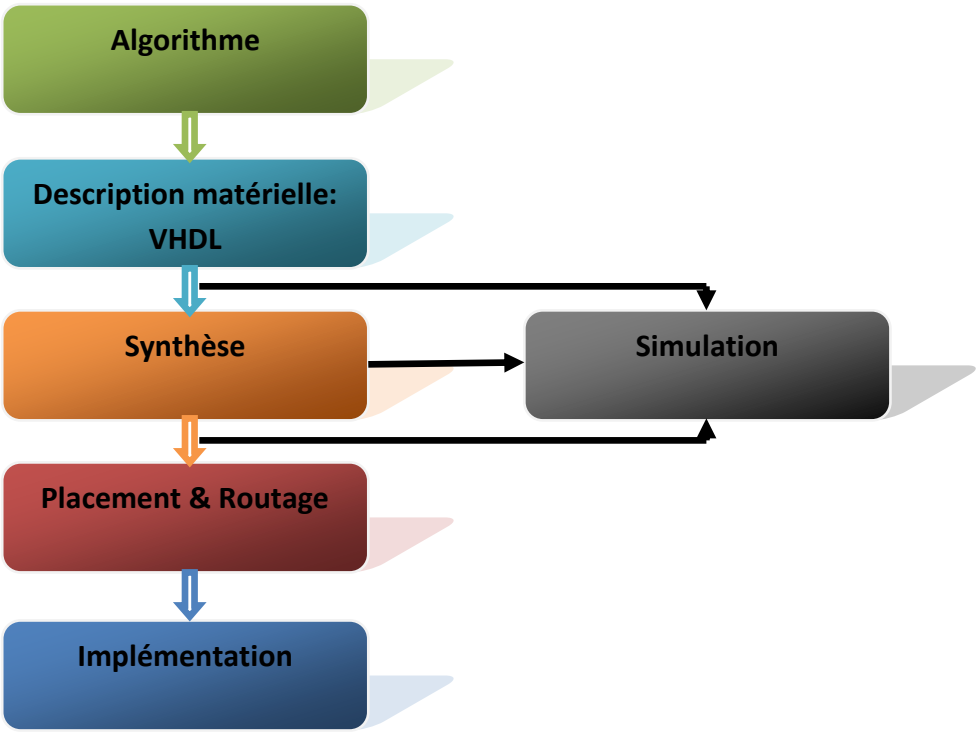

Fig. III.4 : Diagramme de conception.

L'objectif principale de ce projet est d'optimiser de 2 façons: zone (moins de portes possible) ou vitesse (débit maximum). Ensuite, nous aurons deux conceptions: l'un appelé conception rapide et un autre appelé petit conception.

Mais avant de commencer à travailler sur ces modèles, nous devrions comprendre le code qui sera utilisé pour les deux modèles.

III.5.3 Comprendre le code

Avant de passer par le flux de conception, une vérification sur le code fourni est nécessaire. Il y avait 2 fichiers donnés pour ce projet. Le premier, appelé "pipelined-des.vhdl", Contient l'implémentation du processeur DES. L'autre, appelé "Testbench1.vhdl", contient un repère qui sera utilisé dans la simulation processus.

Comme le dit le nom, "pipelined-des.vhdl" est une version en pipeline de 16 étapes d'un DES processeur. C'est un code structurel qui utilise 16 tours. En lisant et en analysant le fichier, nous pouvons écrire un premier schéma sur la façon dont le code est fonctionné.

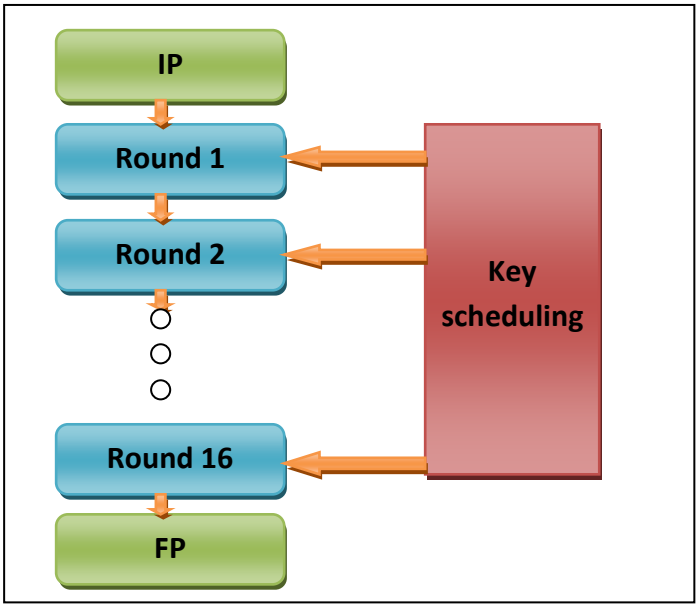

Fig. III.5 : Structure de code.

Ce chiffre correspond à ce que nous avons appris dans la description de l'algorithme DES. Les composants au début et à la fin sont respectivement la permutation initiale et la permutation finale (comme nous l'avons vu dans l'aperçu de DES).

Si nous compilons le code immédiatement tel quel, nous constatons qu'il existe plusieurs erreurs. Mettre tous les composants dans un fichier n'était pas une bonne idée et Model Sim refuse de Compilez le code comme celui-ci. Donc, la première action était de diviser le code fourni dans les différents composants. Nous avons aussi a dû faire quelques modifications (ajouter la bibliothèque et la déclaration des composants utilisés) pour faire fonctionner le code.

En outre, le fichier n'a pas été écrit en VHDL standard. Par exemple, l'opérateur "Rol" utilisé dans l'ordonnancement des clés ne fonctionne pas avec« std_logic_vector » et il y avait certaines ambiguïtés dans les fichiers S-boxes.

Maintenant, nous pouvons décrire plus facilement la fonction de chaque composant.

III.6 Composants du code

a- Desenc :

C'est le sommet du design. Il comprend 4 composants:

- Keysched (key scheduling).
- Ip (initial permutation).
- Roundfunc (round function).
- Fp (final permutation).

Nous connaîtrons le détail de la fonction de ces composants (et de leurs sous-composants).

b- Keyshed :

Il s'agit de composant génération de la clé. Il comprend 2 composants:

- pc1 (permuted choice 1).
- pc2 (permuted choice 2).

PC1 et PC2 tous les deux permutent des composants de bits. PC1 supprime 8 bits de la clé (Qui est à l'origine 64 bits de large). En pratique, ces 8 bits permettent de vérifier si la clé n'a pas été modifiée (avec un contrôle de parité).

PC2 supprime également certains bits pour réduire le nombre de bits de 56 à 48.Le composant keysched applique le premier PC1 puis déplace une ou deux fois une sous-clé après une sous-clé, ensuite, PC2 est appliqué à la fin. Cela conduit à cette observation: toutes les sous-clés sont créées en une seule ligne.

Le composant clé (et ses sous-composants) utilise uniquement des ressources de câblage, Parce qu'il est fait de permutions et d'opérations de déplacement seulement. Donc, cette partie sera exécutée très rapidement et aucune optimisation n'a eu d'effet.

c- IP :

La permutation initiale est seulement une question d'échange des bits de l'entrée (le texte en clair). Nous pouvons donc affirmer que ce composant ne nécessitera pas de logique ressource (uniquement le câblage).

d- Round func :

C'est la fonction ronde, répétée 16 fois comme indiqué dans la conception supérieure. C'est en fait une conception structurelle qui relie ensemble les composants suivants:

- Xp (expansion).

- desxor1 (xor 1).
- s1, s2, s3, s4, s5, s6, s7, s8 (S-boxes).
- pp (p-permutation).
- desxor2 (xor 2).

Xp : signifie expansion, car son comportement consiste à étendre le nombre de bits de 32à 48 bits (ce qui est implémenté avec seulement des ressources de câblage). Desxor1 est un géant de la 48 bits du porte xor qui exoré la sous-clé et l'entrée élargie de la fonction ronde.

Ensuite, nous voyons les 8 S-boxes qui sont en fait des tables de consultation. Les registres nécessaires Pour le pipeline sont également intégrés dans les boîtes S. PP est P permutation, donc échange de bits. Enfin, un autre xor (desxor2) est responsable d'exoré le résultat de la permutation P avec la partie gauche du tour précédent.

e- FP :

La permutation finale est l'inverse de la permutation initiale.

Nous pouvons donc conclure que beaucoup de composants sont uniquement constitués de ressources de câblage.

Il n'y aura donc aucun effort pour optimiser la logique sur ces composants. Les seuls composants qui utiliseront les ressources logiques:

- desxor1, desxor2 et S-boxes.

III.7. Conception rapide (Fast Design)

Nous avons étudié tous les composants afin que nous puissions construire d'abord notre conception rapide. Mais avant cela, nous allons ajouter des optimisations au code VHDL pour atteindre une vitesse plus élevée.

III.7.1. Approche pipeline [33]

Desxor1 et desxor2 ne peuvent pas être logiquement optimisés car ils utilisent la base de la fonction xor. Nous allons mettre l'accent sur les S-boxes, qui correspond à un grand cas en VHDL, et qui seront implémentées dans des LUTs de FPGA.

Nous pouvons utiliser la figure suivante pour expliquer comment fonctionne l'architecture pipeline :

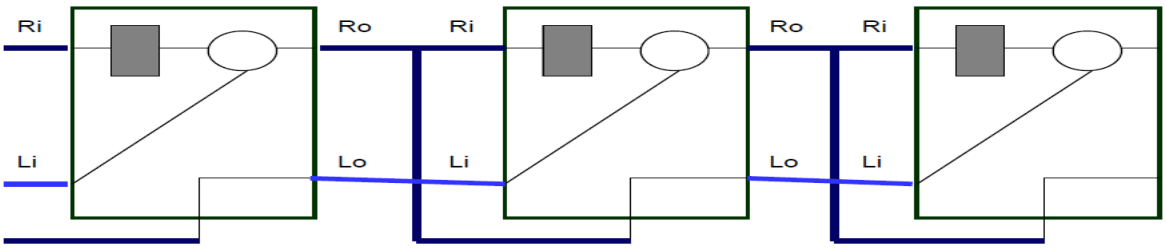

Fig. III.6 : Architecture d'un pipeline original.

Les boîtes S qui contiennent les registres sont surlignées en gris sur la figure ci-dessus, étant donné que les registres ont une largeur de 4 bits (taille de la sortie à sauvegarder) et que nous avons 8 boîtes en S, nous obtenons 32 bits mémorisés dans chaque étape (une étape = un tour). Il peut sembler étrange parce que nous avons 64 bits à transmettre entre chaque étape.

Mais si l'on considère l'algorithme DES, la partie gauche du prochain tour est la partie droite du précédent. Donc seulement la moitié de l'information doit être stockée. En faisant de cette façon, L'auteur du code a privilégié une architecture optimisée pour la surface, car nous ne stockons que 32 bits au lieu de 64 bits. Mais cette approche conduit à une conception moins efficace en termes de la vitesse. Pour prouver cette affirmation, nous devons examiner le chemin critique**,** qui détermine en réalité la fréquence d'horloge [34]. Le chemin critique dans une conception pipeline est le chemin le plus long entre deux étapes (c'est-à-dire, entre deux registres) [35]. Nous allons le caractériser en utilisant la figure ci-dessus. Par la suite nous avons deux chemins à considérer :

1- Commencez par le registre, passez par la fonction f, passez de r_0 à r_i et arrivez à le prochain registre.

2- Commencez par le registre, passez par la fonction f, passez de r_0 à li, passez à l_0, puis a l_i du prochain tour, passez par la fonction f, passez de r_0 à r_i et enfin arriver au prochain registre.

Nous concluons que le deuxième chemin est le plus long et peut être réduit beaucoup. En effet, le deuxième chemin à la distance la plus longue à parcourir, mais surtout, il va deux fois par la fonction f (et le xor qui a été fusionné dans la fonction f dans le figure ci-dessus à des fins de simplicité). Nous présentons ici un deuxième registre qui réduira le chemin critique vers un seul stade (C'est-à-dire un tour). Nous mettrons les deux registres à la fin du tour. Donc nous espérons avoir un grand coup de vitesse.

Une modification utile à la conception est d'ajouter une broche de réinitialisation. C'est vraiment facile à mettre en œuvre et cela n'aura pas d'impact important sur les performances finales.

Un autre point qui doit être étudié est le problème des E / S. Comme nous l'avons vu avant, le FPGA Spartan-3EXC3S100E package TQ144 n'a que 108 E / S. Et nous avons :

- 64 bits pour le texte clair.
- 64 bits pour la clé.
- 1 bit pour la réinitialisation.
- 1 bit pour l'horloge.
- 64 bits pour le chiffre.

La somme est de 194 E / S! Nous devons donc réduire le nombre d'E / S. Une façon de faites ceci pour utiliser un convertisseur qui chargera une partie des données pendant une certaine horloge cycles et ensuite livrer les données complètes (et ainsi de suite à plusieurs reprises). Nous pouvons ainsi diviser les entrées (clé et texte clair) comme ceci :

- 16 bits pour le texte clair.
- 16 bits pour la clé.
- 1 bit pour la réinitialisation.
- 1 bit pour l'horloge.
- 64 bits pour le chiffre.

La somme est maintenant de 98 afin que notre conception s'insère dans le FPGA (pour les E / S).Par conséquent, deux convertisseurs seront ajoutés à la conception qui prendra 4 cycles d'horloge à rassemblez les 64 bits nécessaires pour le texte clair et la clé. Le comportement d'un convertisseur est très similaire à un convertisseur série à un convertisseur parallèle, sauf que dans notre cas, nous convertir un bloc de plusieurs bits en bloc plus grand.

En fait, seuls 3 cycles d'horloge sont nécessaires pour cette opération, car au troisième cycle d'horloge, les données entrantes sont concaténées avec les données stockées.

Un signal ajouté à la conception appelé "load data" informe les convertisseurs qu'une nouvelle clé ou un nouveau texte clair est transmis aux entrées de la conception. Un autre point à explorer serait de mettre en place les S-boxes avec des éléments ROM.

La FPGA dispose de certaine surface réservée à ces éléments, comme on le voit sur la figure suivante :

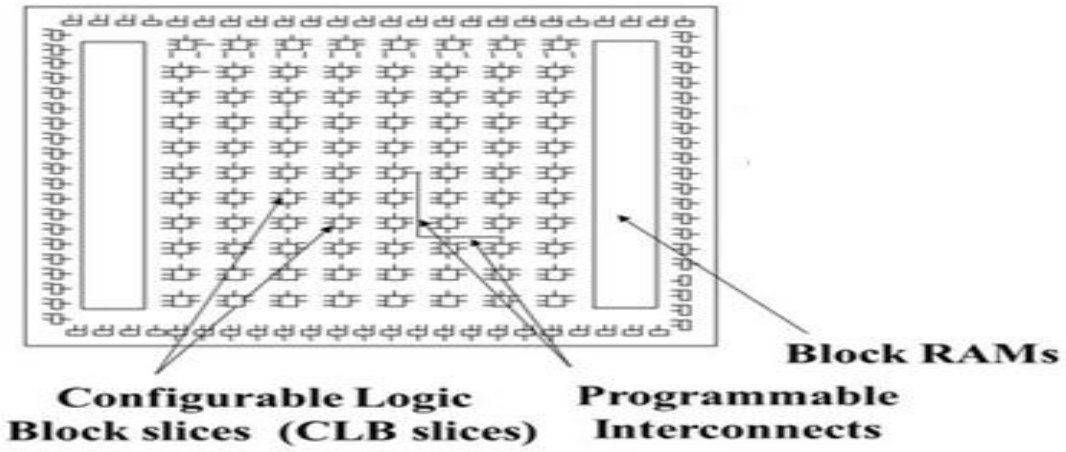

Fig. III.7 : Block RAM sur le Spartan3E-100.

ROM serait mieux implémentée dans le Block RAM au lieu d'utiliser les CLB, mais dans ce projet, nous n'allons pas loin. Ce serait intéressant d'utiliser cette technique si nous veulent une conception entièrement optimisée. Cela réduirait bien sûr le nombre de CLB utilisés mais cela peut aussi accélérer l'implémentation. Donc, pour ce projet, nous aurons confiance au synthétiseur pour avoir la meilleure implémentation des S-boxes. Nous sommes maintenant prêts à tester notre conception optimisée.

III.7.2 Processus de simulation (Fast Design)

Pour faciliter la simulation de la conception, un fichier a été fourni: testbench1.vhd.Le principe de ce banc d'essai est une boîte noire qui utilise le design supérieur (desenc), il donne des vecteurs de simulation prédéfinis (le texte clair, la clé et l'horloge) et vérifient si la sortie (le chiffre) correspond au chiffre valide. Le banc d'essai donne en fait un nouveau texte clair et une nouvelle clé tous les 16 cycles d'horloge et comparez la sortie à la fin avec le chiffre associé.

Notre architecture de pipeline donne un bloc de chiffrement (64 bits) à chaque cycle d'horloge, sauf au début lorsque le pipeline doit être rempli (pendant 16 cycles d'horloge parce que nous avons 16 étapes), il y a donc une latence de 16 cycles d'horloge.

Nous pourrions donc écrire un autre banc d'essai qui donne un texte clair pendant 16 cycle d'horloge et puis un texte clair chaque cycle d'horloge, mais ce ne serait pas pratique, effectivement avec le banc d'essai fourni, nous pouvons comparer immédiatement la clé et le

texte clair qui ont toujours la même valeur avec le chiffrement calculé (comme nous le verrons dans la prochaine figures).

Après cela, nous obtenons cette forme d'onde :

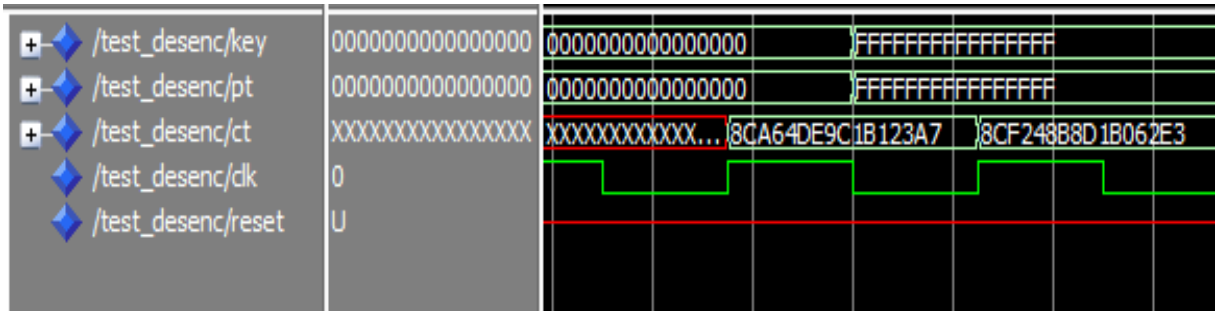

Fig. III.8 : Résultat de simulation testbench1.

La première ligne de la figure est la clé, la seconde est le texte en clair, la troisième est le texte chiffré, le quatrième est l'horloge et le dernier est la réinitialisation. Le chiffre est celui que nous attendions. On peut voir que pendant les prochains cycles après le premier chiffre, d'autres chiffres sont présents, mais ils ne signifient rien. Normalement, nous allons donnez un chiffrement à chaque cycle d'horloge, donc il n'y aurait pas ces "chiffrages parasites". Compte tenu de la version avec les convertisseurs, nous devons écrire un autre banc d'essai qui donne 16 bits pour la clé et le texte clair et qui gèrent le signal 'Load data'.
Ce nouveau banc d'essai donne ces résultats :

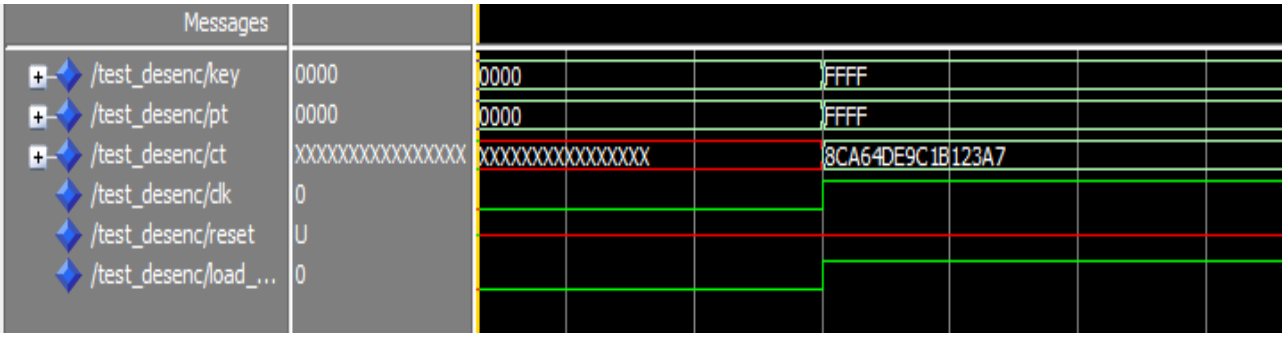

Fig. III.9 : Résultats de simulation de la conception rapide avec le convertisseur.

Une autre ligne (la dernière) a été ajoutée qui représente le signal 'load data'. Le chiffre donné est correct. Puisque nous avons vérifié le fonctionnement de notre conception, nous pouvons maintenant passer par la synthèse processus.

III.7.3 Résultats de synthèse (Fast Design)

1- L'estimation des ressources pour cette conception à partir d'ISE est la suivante :

Tableau. III.2 : Résumé de l'utilisation des composants (valeurs estimées).

Utilisation logique	Utilisé	Disponible	Utilisation
Nombre de tranche	1911	960	199%
Nombre de bascules en tranche	1024	1920	53%
Nombre de LUT 4 entrées	3718	1920	193%
Nombre d'IO Bliés	186	108	172%
Nombre de GCLKs	1	24	4%

2- Le schéma RTL de la conception rapide (fast design) Spartan 3E XC3S100E est :

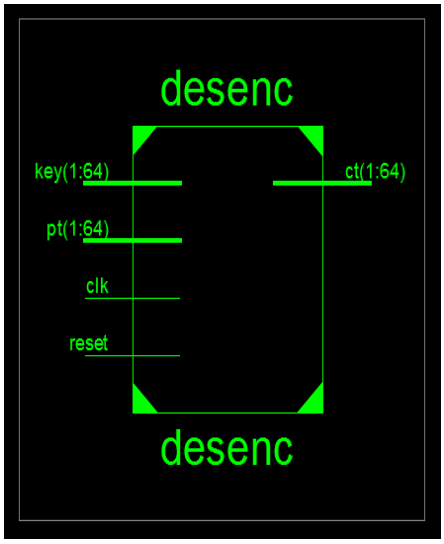

Fig. III.10 : Schéma bloc générale de circuit DES Pipeline.

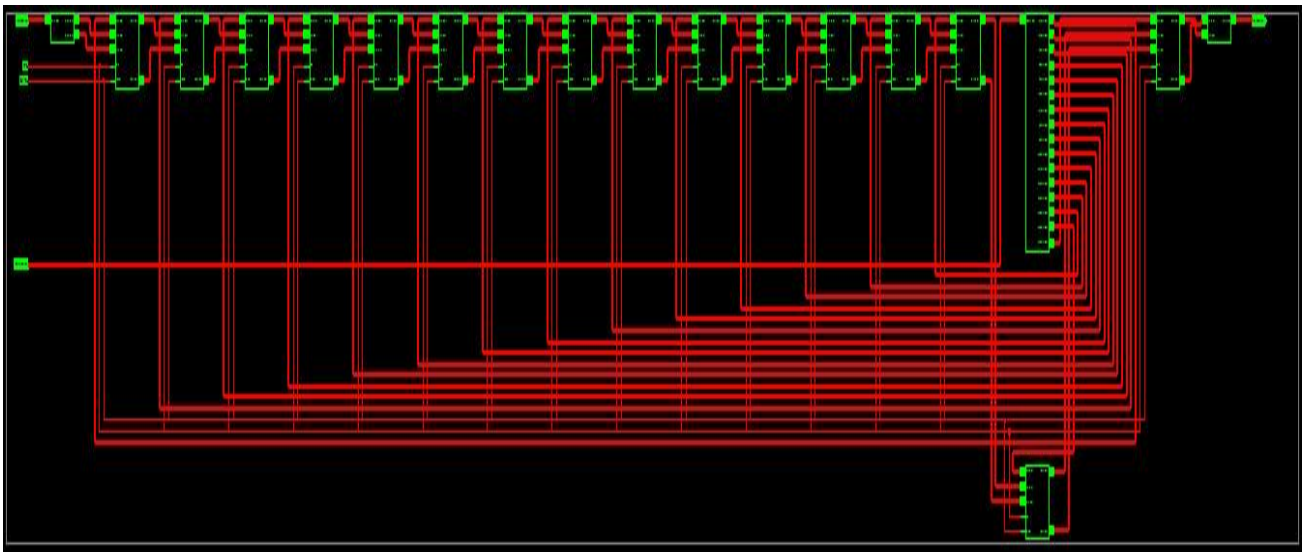

Fig. III.11 : Aperçue du schéma RTL de la conception rapide.

3- La fréquence maximale estimée de cette conception est: **181.653MHz**.

4- On peut calculer le débit de cette conception à partir de la fréquence maximale, En général, le débit est calculé comme suit:

Le débit=fréquence×64 /16 = 181.653×64/16=726.612Mbits/s

5- L'estimation des ressources pour cette conception à partir d'ISE (fast design) avec convertisseur est la suivante :

Tableau. III.3 : Résumé de l'utilisation des composants (valeurs estimées) avec convertisseur.

Utilisation logique	Utilisé	Disponible	Utilisation
Nombre de tranche	1964	960	204%
Nombre de bascules en tranche	1238	1920	64%
Nombre de LUT 4 entrées	3808	1920	198%
Nombre d'IO Bliés	97	108	89%
Nombre de GCLKs	2	24	8%

6- Le schéma RTL de cette conception avec convertisseur Spartan 3E XC3S100E est :

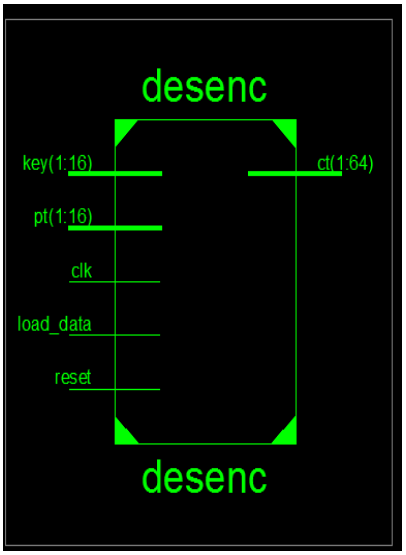

Fig. III.12 : Schéma RTL de DES avec convertisseur.

III.8. La conception réduite (Small design)

Afin d'obtenir un design avec la plus petite surface possible, nous devons changer notre architecture. Nous avions une architecture pipeline qui est très efficace pour atteindre une vitesse élevée. Mais maintenant nous voulons utiliser la surface la plus basse possible.

De toute évidence, l'architecture pipeline gaspille beaucoup de surface sur le FPGA, car elle utilise 16 fois le même tour. Le défi est donc de trouver un moyen d'utiliser un seul tour et de faire boucle de données 16 fois. Cette idée peut être implémentée avec une machine d'état. Cela conduit à ce qui suit la figure suivante :

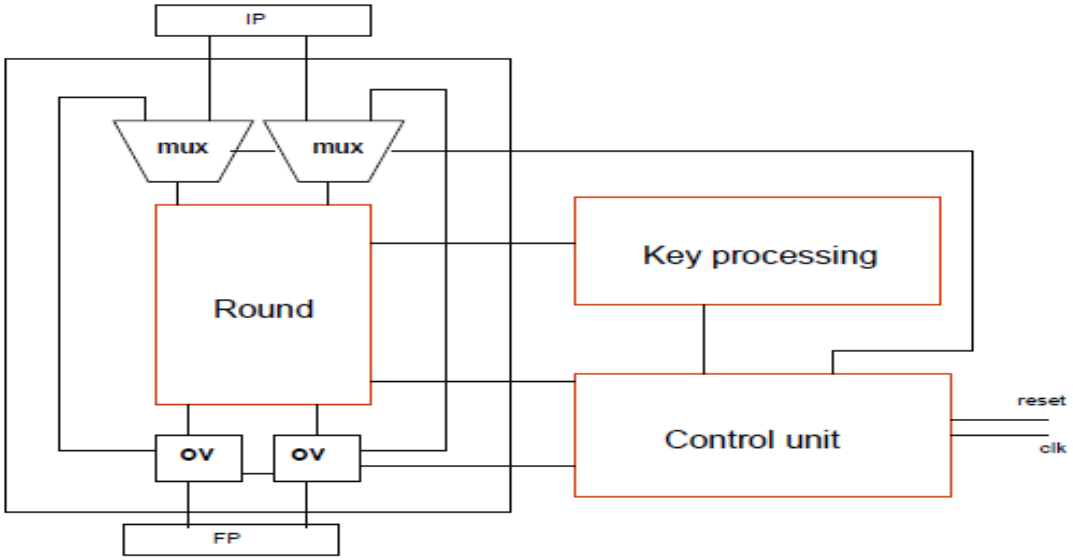

Fig. III.13 : La machine d'état.

En faisant cela, nous pouvons réutiliser les mêmes composants ronds que nous avions avec le pipeline architecture. La programmation des clés doit toutefois être réécrite dans une unité de traitement de clé (Nous appelons ce processeur clé, ou unité de traitement clé).

Naturellement, une unité de contrôle doit être incluse dans la conception. Tous ces composants supplémentaires ne coûteront pas beaucoup de surfaces, mais nous devons également inclure deux multiplexeurs aux entrées du cycle. Comme les entre sont 64 bits de largeur, nous aurons deux multiplexeurs 32 bits, qui peuvent coûter beaucoup de surfaces. Ces deux multiplexeurs laissent passer l'entrée de la conception à travers le tour (charge nouvelles données), ou prenez la sortie du tour pour les réinsérer dans le tour (boucle).

Juste après la ronde et avant la permutation finale, nous avons deux validateurs de sortie (ov sur la figure ci-dessus). Ces composants garderont la même sortie entre chaque nouvelle donnée / clé (tous les 16 cycles d'horloge). Nous pourrions utiliser différentes méthodes comme la fréquence mais la sortie valider est probablement la solution la plus simple avec un faible coût de côté de la surface.

Nous pouvons imaginer que nous avons une machine d'état de 16 états, mais en réalité, nous aurons17 (car nous avons besoin d'un état pour charger la clé). Nous verrons cela en détail plus tard. Avec cette nouvelle architecture, nous avons un débit divisé par 17 (un chiffre tous les 17cycles d'horloge au lieu d'un chiffre de chaque cycle d'horloge avec l'architecture pipeline). Il signifie que la performance de cette architecture sera très faible (mais ce n'est pas important puisque nous privilégions la surface).Passons maintenant à la description de ce nouveau design.

III.8.1. Description de nouvelle conception (Small design)

1. Unité de contrôle

Comme mentionné précédemment, nous avons une machine d'état de 16 états. L'unité de contrôle est responsable de ces tâches:

- Demander une nouvelle sous-clé à chaque cycle d'horloge (changement de signal).
- Indiquer aux multiplexeurs s'ils doivent charger de nouvelles données ou prendre les sorties du tour (Signal load new pt).
- Dites aux validateurs de sortie d'activer leurs sorties aux sorties courantes du cycle tous les 16 cycles d'horloge (signal output ok).

En réalité, nous devons fournir deux informations sur le processeur clé: si nous avons besoin une nouvelle clé ou si nous devons changer une fois, deux fois ou rien. Tout peut être codé dans un seul vecteur de 3 bits (appelé décalage) qui peut être réalisé comme indiqué dans ce tableau ci-dessous :

Tableau. III.4 : Le vecteur d'état.

Valeur	Action
000	Sans changement, pas de nouvelle clé
001	Pas de changement, nouvelle clé
010	Décalage une fois, pas de nouvelle clé
011	Décalage une fois, nouvelle clé
100	Décaler deux fois, pas de nouvelle clé
101	décalage deux fois, nouvelle clé
Autre	Erreur => sans changement, pas de nouvelle clé

Le bit le moins significatif (bit situé à l'extrême droite) indique si une nouvelle clé est nécessaire, Tandis que le bit du milieu indique si nous voulons changer une fois (1 = oui, 0 = non) et enfin le bit le plus significatif dit de changer deux fois. Des valeurs comme 111, 110 sont impossibles (il n'y a pas d'états codés avec 111 ou 110).

Le décalage du signal sera facilement décodé à l'aide d'une déclaration de cas dans le VHDL.

Voici donc une description approximative du comportement du composant de contrôle :

Init: Charger une nouvelle clé, décalé une fois.

Key loading: Charger une nouvelle clé, décaler une fois, donner la sortie (ct).

State 1: Décaler une fois.

State 2: Décaler deux fois.

State 3: Décaler deux fois

State 4: Décaler deux fois

State 5: Décaler deux fois

State 6: Décaler deux fois

State 7: Décaler deux fois

State 8: Décaler une fois

State 9: Décaler deux fois

State 10: Décaler deux fois

State 11: Décaler deux fois

State 12: Décaler deux fois

State 13: Décaler deux fois

State 14: Décaler deux fois

State 15: Décaler une fois

State 16: aucun

Toutes les instructions relatives à l'état sont exécutées à l'état suivant (état futur).L'état "chargement des clés" est nécessaire, comme le dit le nom, pour charger la clé avant début de la boucle. Il est possible de le supprimer mais nous avons dû modifier la planification de la clé, de sorte qu'il utilisera plus de ressources logiques. Comme l'objectif est de minimiser la quantité d'espace, nous ne supprimerons pas cet état.

Le comportement de l'unité de contrôle a été expliqué et nous allons maintenant poursuivre unité de traitement clé.

2. Unité de traitement de clé :

Nous nous souvenons que, dans l'architecture pipeline, nous avions une planification de clé implémentée avec un seul câblage. Ce ne serait pas le cas avec notre petit design.

En fait, l'unité de traitement de clé donne une nouvelle sous-clé à chaque cycle d'horloge, donc elle a besoin certains registres.

Cependant, les composants PC1 et PC2 restent uniquement des ressources de câblage. Donc un nouveau composant apparaît: shifter. Le décalage change simplement une ou deux fois le stockage sous clé. Par conséquent, nous aurons un registre de 56 bits pour stocker cette clé.

Le shifter est basé sur les composants logiques suivant:

- Multiplexeurs.
- Un décodeur (pour décoder le signal de changement de vitesse).
- Registre pour stocker la clé.

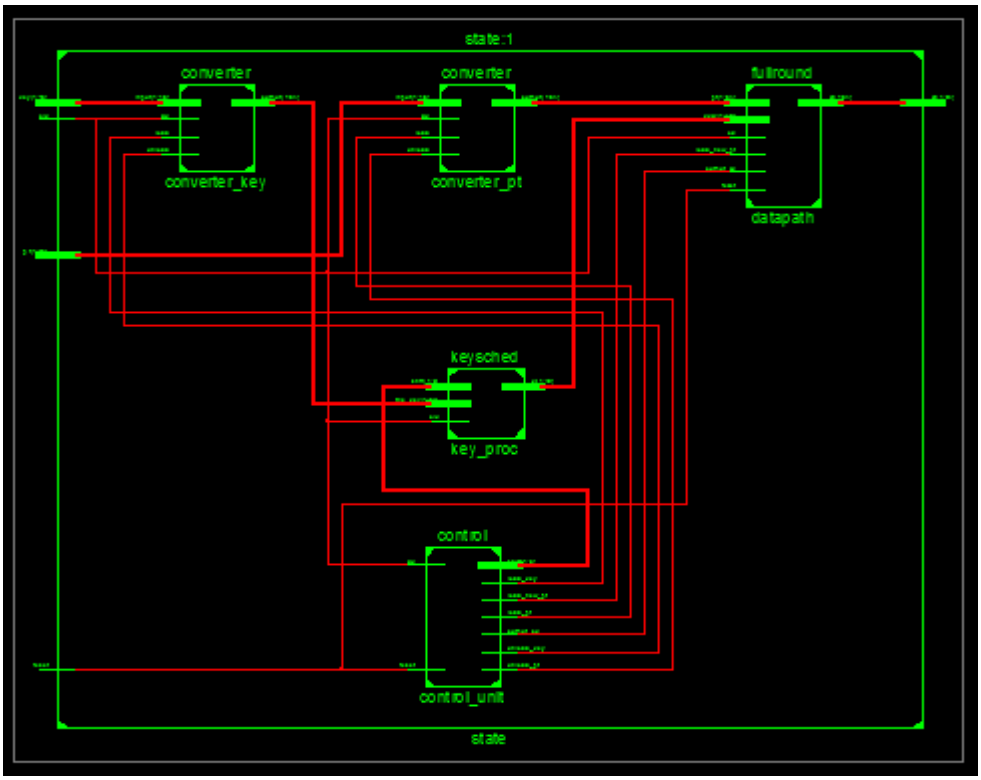

Fig. III.14: Aperçue du schéma RTL de la conception réduite

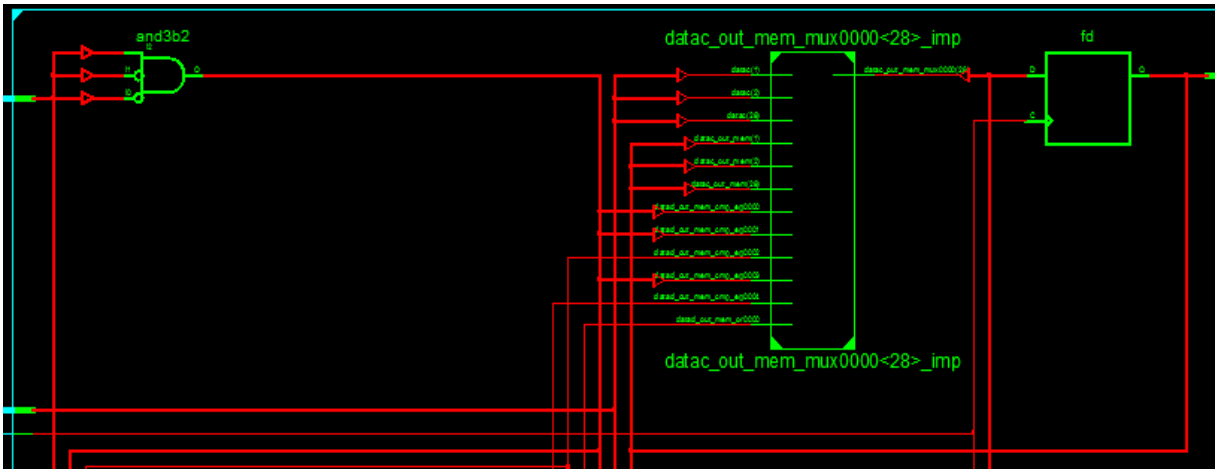

Fig. III.15: Une section de schéma RTL d'un shifter.

3. Unité complète (full round unit) :

Le tour complet est fondamentalement le nom du composant qui contient le tour composants (le même que nous avions dans le design rapide) plus les deux validateurs de sortie. Comme nous l'avons déjà décrit, la composante ronde dans le design rapide, aucune explication sont nécessaires. Les validateurs de sortie (ov32) sont réalisés uniquement à l'aide de registres.

La conception de notre nouvelle architecture est maintenant prête à être testée.

III.8.2. Processus de simulation (Small design)

La simulation est aujourd'hui devenue de rigueur et permet un gain de temps élevé par rapport à une mise au point totalement exécutée sur le matériel. Elle s'exécute sur la machine de développement [36].

Nous avons vu dans la conception rapide que la quantité de broches sur notre FPGA nous oblige à réduire le nombre d'entrées / sorties de la conception. La même remarque s'applique ici et nous doit réutiliser le convertisseur pour réduire le pt et la clé à 16 bits chacun.

Après l'intégration des convertisseurs, nous pouvons ré-simuler notre conception modifiée. Normalement, Nous avons besoin des 17 cycles d'horloge nécessaires, mais nous avons besoin de 4 cycles d'horloge supplémentaires

Mais nous traitons cette situation différemment avec le banc d'essai. Bien sûr, les premières données aura besoin de 4 cycles d'horloge à charger, plus 1 pour l'état INIT, plus 17 pour la conception à travail. Mais nous pouvons charger les données suivantes alors que la conception est en train de calculer le chiffre. Alors on peut éliminer ces 4 cycles d'horloge et on obtient seulement 17 cycles d'horloge au lieu de 21.Après avoir réécrit le banc d'essai avec ces nouvelles considérations, nous les obtenons résultats :

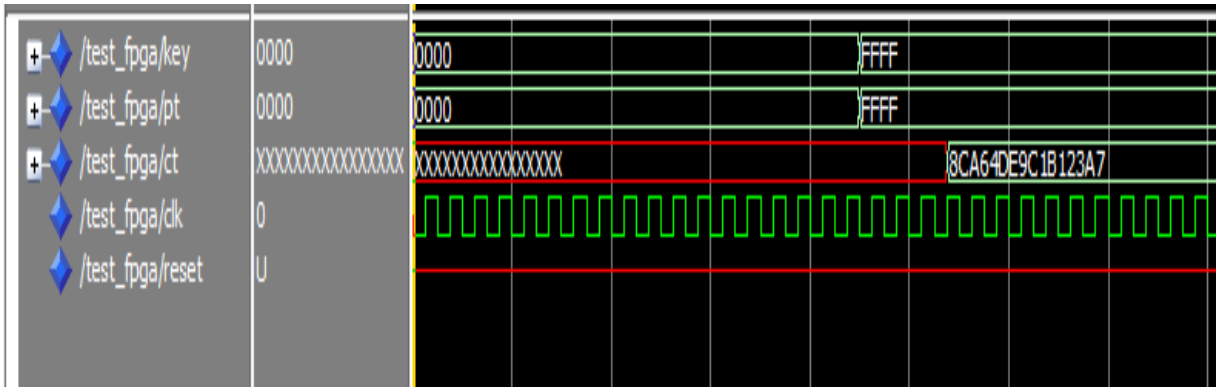

Fig. III.16 : Le résultat de simulation de la conception réduite.

Nous voyons clairement que le nouveau chargement de données 4 cycles d'horloge avant la sortie (ct) est donné. Alors que le design rapide souffre des 4 cycles supplémentaires d'horloge pour charger Les données, la petite conception peut éliminer ce problème.

Maintenant que la conception fonctionne, nous sommes prêts à synthétiser.

III.8.3. Processus de synthèse (Small design)

1. L'estimation des ressources pour cette conception à partir d'ISE est la suivante :

Tableau. III.5 : Résumé de l'utilisation des composants (Valeurs estimées).

Utilisation logique	Utilisé	Disponible	Utilisation
Nombre de tranche	238	960	24%
Nombre de bascules en tranche	367	1920	19%
Nombre de LUT 4 entrées	245	1920	12%
Nombre d'IOBliés	98	108	90%
Nombre de CLICKs	1	24	4%

2. La fréquence maximale estimée de cette conception est:**171.102MHz.**
3. On peut calculer le débit de cette conception à partir de la fréquence maximale, En général, le débit est calculé comme suit:

Débit = fréquence × 64/17 = 644.1Mbits/s

Nous savons déjà comment la fonction ronde sera implémentée car nous avons la même chose que le design rapide. Cependant, nous devons vérifier si l'unité de traitement de clé, l'unité de control et les sorties validés (intégrés intégralement) sont implémentées efficacement.

Comme nous le constatons dans la figure (**Fig. III.15)**, le shifter (qui est le seul composant dans la planification de clé à l'aide d'éléments logiques) est composé de registres (appelés FDE) et des tables de consultation (LUT).

4. **Le schéma RTL de cette conception (Small design)** Spartan 3E XC3S100E est **:**

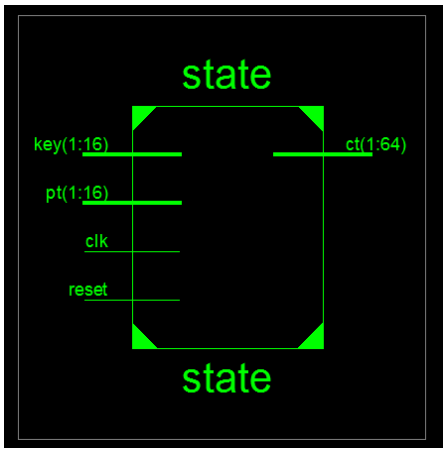

Fig. III.17 : Schéma RTL de la conception réduite.

III.9. Conclusion

Dans ce chapitre, nous avons optimisé l'implémentation des deux conceptions de l'algorithme DES sur la plateforme FPGA xilinx spartan3E XC3S100E et nous avons sauvegardé tous les résultats obtenus à l'aide des logiciels qui nous avons utilisés (ISE14.5 MODELSIM 6.5), on constate qu'il y a une grande différence entre les deux conceptions.

La grande différence réside dans le fait que la conception rapide consomme plus de ressources de notre FPGA en pourcentage 199% par contre la petite conception (Small design) consomme 24% soit moins de 8 fois, cette estimation implique que la conception réduite (Small design) consomme moins d'espace sur le circuit FPGA que la conception rapide. Une autre comparaison concerne le nombre de bascule en tranche, le fast design consomme 53% par contre le Small design ne consomme que 19%. Un autre point très important est le nombre d'entrées sortie lier la première conception qui consomme 172% ce qui ne permet d'implanter la conception rapide sur Spartan3E XC3S100E. Mais avec la deuxième conception ce problème est résolu parce que la conception Small ne consomme que 90% des E/S. Alors on conclut que le Small design est le mieux adapté pour une implémentation sur ce type de FPGA Spartan 3E XC3S100E.

Conclusion général

Les avantages de l'implémentation matérielle sont, par nature, plus sécurisée physiquement, plus rapide (haut débit), plus fiable. Les circuits FPGAs sont des circuits logiques programmables dont la fonction implémentée n'est pas figée, et ils peuvent être programmés plusieurs fois sur site (in situ). C'est donc une alternative prometteuse pour l'implémentation de chiffrement à bloc. Nous avons donc présenté une implémentation matérielle optimisée de l'algorithme cryptographique DES *(Data Encryption Standard)* sur une plate-forme reconfigurable basée sur FPGA *(Field Programmable Gâte Arrays).* Nos conceptions ont été effectuées et simulées sur un circuit FPGA cible*Spartan3E XC3S100E.*

Notre approche pipeline met l'accent sur le parallélisme des S-boxes de l'algorithme DES et qui seront implémentées dans les circuits FPGA, ce qui permet d'atteindre des résultats, tout à fait, compétitifs par rapport à d'autres implémentations matérielles de DES.

Deux architectures sont étudiées pour ce projet : la première est la plus rapide possible et la deuxième consomme moins de ressources matérielles que la première architecture sur le FPGA. Le sens de la vitesse pour ce projet est le débit (nombre de bits traités par seconde) et la signification de ressources matérielles est le nombre de CLB (Bloque Logique Configurable).

La conception rapide (Fast Design) est une architecture très rapide mais qui consomme beaucoup de ressource sur le FPGA et notamment le nombre d'entrées sorties plus que la quantité disponible sur le circuit choisi Spartan 3E XC3S100E.
Cette conception est utilisée lorsque le circuit FPGA possède suffisamment de ressources.

L'utilisation d'un convertisseur dans la conception rapide pour but de réduire seulement le nombre d'entrées sorties.

La conception (Small design) méthode idéale pour minimiser l'architecture de l'implémentation lorsqu'on a une FPGA à faible nombre de ressource comme notre cas le Spartan 3E XC3S100E.

Le cryptage par DES est caractérisé par sa facilite d'implémentation. Il n'y a en outre pas de différence entre le chiffrement et le déchiffrement (seul l'ordre dans lequel les bits de la clé sont présentés aux fonctions *f* diffère). C'est pourquoi c'est un standard adopté par un grand nombre d'institutions. Cependant, sa clé de petite taille incite ses utilisateurs a le remplacer par l'algorithme AES. Nous proposons en guise de perspectives d'implanter une version optimisée de l'algorithme AES dans un circuit FPGA approprié dans un projet future.

Références bibliographiques

[1] : A. Canteaut et F. Levy-Dit-Vehel, « La Cryptologie Modern », V. Ensta, Paris 15.2001.

[2] : Un cours tutorial de centre canadien télédétection, « notion fondamentaux de la télédétection », ressources naturelles, canada, 2011.

[3] : B. Schneier, « Cryptographie Appliquée », 2ème édition ITP, 1997.

[4] : http://laurent.flaum.free.fr/pgpintrofr.htm.

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

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

[7] : F. Arnault, « Théorie des nombres & cryptographie », cour2 D.E.A, Université de Limoges, 12 mai 2003.

[8] :http://math.univ-lyon1.fr/~roblot/masterpro.html/chapitre1, [En ligne], [Citation : 06 - 11- 2012].

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

[10] : M, Blaze, W. Diffie, R. L. Rivest, B. Schnieir, T. Shimomura, E. Thomson et M. Wiener, « Longueur Minimale des clefs de chiffrement symétriques afin désassurer une sécurité commerciale suffisante », un rapport par un groupe ad hoc de cryptographes et de scientifiques informatiques, janvier1996.

[11] : file:///I:/Principe%20de%20Kerckhoffs%20%E2%80%94%20Wikip%C3%A9dia.htm.

[12]: R. Stinson, « Cryptography theory and practice, (discrete mathematics and its applications) », chapman & hall / crc press, New York, November 2005.

[13] : A. Lan et B. Vandervelde, «Panorama des algorithmes de Cryptographie », 13 mars 2011.

[14] : M. B. M. Belkaid, « Application des techniques de cryptage pour la transmission sécurisé d'image MSG », Mémoire de Magister en électronique, Mouloud Mammeri, Tizi Ouzou, 2015-2016.

[15] : Mr. FILALI Mohamed Amine, « Etude et Implémentation Pipeline sur FPGA de l'algorithme de Chiffrement AES», Mémoire de Magister, Université de Mohamed Boudiaf, Oran, 2014-2015

[16] : A. Wurcker, « Etude de la sécurité d'algorithmes de cryptographie embarquée vis-à-vis des attaques par analyse de la consommation de courant», Thèse de Doctorat en Informatique, Université de Limoges, France, 15/10/2015.

[17] : J. Emonet, « Algorithmes de chiffrement », 22 juin 2005.

[18]: W. C. Barker, «Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher», Revised 19 May 2008, NIST Special Publication 800-67, Version 1.1.

[19] : F. Germain, « Sécurité cryptographique par la conception spécifique de circuits intégrés », Thèse Ph. D, Ecole Polytechnique, Paris VI, Juin 2006.

[20]: O. Mencer, M. MORF ET MJ Flynn, « Hardware Software tri-design of

Encryption for mobile Communication Units », In Proceeding, volume 5, 1998.

[21]: C. Liua et Jin-shui Ji et Zi-long Liu, « Implementation of DES Encryption Arithmetic based on FPGA », 2013 AASRI Conference on Parallel and Distributed Computing Systems, Northwest University for nationalities Lanzhou, Gansu, China.

[22]: https://fr.wikipedia.org/wiki/Constantes_du_DES.

[23]: M. Sébastien, « Implémentation du DES et cryptanalyse sur FPGA », mémoire d'ingénieur civil électricien, Université Catholique de Louvain, Septembre 1999.

[24] : N. Toufik, « Implémentation d'une instrumentation sur un FPGA », Mémoire de Magister en électronique, Mouloud Mammeri, Tizi Ouzou, 2011-2012.

[25] : https://fr.wikipedia.org/wiki/Circuit_logique_programmable#FPGA.

[26] : National instruments: http://www.ni.com.

[27] : S. Pillement, O. Sentieys et R. David, « Architectures Reconfigurables : un survol. », LASTI - ENSSAT, 26/04/2002.

[28] : J-P. Delahaye,« Systèmes Radio Dynamiquement Reconfigurables sur des Architectures Hétérogènes », Mémoire DEA SETI, Université Paris XI, septembre 2003.

[29]: D.C. Feldmeier, «A high-Speed Software DES Implementation », Computer Communication Research Group, Bellcore, Morristown, NJ, Juin 1989.

[30]: S. Landau, «Standing the Test of Time: The Data Encryption Standard», NOTICESOF THE AMS, Sun Microsystems Inc, VOLUME 47, NUMBER 3, 2000 Mars.

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

[32] : S. Le beux, «Un flot de conception pour applications de traitement du signal systématique implémentées sur FPGA à base d'Ingénierie Dirigée par les Modèles», Thèse de doctorat, UNIVERSITÉ DES SCIENCES ET TECHNOLOGIES DE LILLE, France, Décembre 2007.

[33]: C. Patterson, «High Performance DES Encryption in Virtex FPGAs using Jbits»,

In: Field-programmable custom computing machines, FCCM' 00, Napa Valley, CA, USA, IEEE Comput. Soc., CA, USA, (2000) 113-121,2000.

[34] : G. Rouvroy, FX. Standaert, 11. Quisquater et JD. Legat, « Efficient Uses of FPGA's for implementations of DES and ifs experimental linear cryptanalysis», IEEE Transactions on Computers, Special CHES Edition, pages 473-482, April 2003.

[35] : E. Bennour et E. M. Abdülhamid, « Les problèmes d'ordonnancement cycliques dans la synthèse des systèmes numériques», Université de Montréal, Publication 996, Oct. 1995.

[36] : R. Ben Atitallah, «Modèles et simulation de systèmes sur puce multiprocesseurs Estimation des performances et de la consommation d'énergie», Thèse de doctorat, Laboratoire d'informatique fondamentale de Lille, Université des sciences et technologies de Lille, France, Décembre 2007.

Résumé :

Dans ce mémoire, nous présentons l'implémentation de l'algorithme cryptographique DES *(Data Encryption Standard)* sur une plate-forme reconfigurable basée sur FPGA *(Field Programmable Gâte Arrays).* Notre conception a été implémentée sur une *Spartan 3E XC3S100E de package TQ144.* L'architecture du design a été décrite dans le langage VHDL, deux architectures sont étudiées pour ce projet : la première est le plus rapide possible, dans ce conception nous avons utilisé une *approche pipeline* avec l'accent sur la non-linéarité des S-boxes de DES et la deuxième architecture réduite consomme moins de surface que la première architecture sur le FPGA. Le sens de la vitesse pour ce projet est le débit (nombre de bits traités par seconde) et la signification de la surface est le nombre de CLB (*Configurable Logic Block*). Pour but de réduire la consommation de la surface sur le circuit FPGA parce qu'on a un type de FPGA de faible ressources donc nous allons favoriser une architecture réduite (Small design).

Mots clés: Architecture rapide, Architecture réduite, Cryptographie, DES, FPGA, Implémentation, Pipeline, VHDL.

Abstract:

In this paper, we present the hardware implementation of the Data Encryption Standard cryptographic algorithm on a reconfigurable platform based on FPGA (*Field Programmable Gate Arrays*). Our design was implemented on a Spartan 3E XC3S100E device package TQ144. The architecture of the design has been described in the VHDL language. Two architectures are studied for this project: one which is the fastest possible in this design we used a *pipeline approach* with emphasis on the non-linearity of des-S boxes, and another one which results in the less area than the first architecture on the FPGA. The meaning of speed for this project is the throughput (number of bits processed per second) and the meaning of area is number of CLB's (*Configurable Logic Block*). In order to reduce the consumption of the area on the FPGA circuit because we have a type of FPGA of low resources so we will favor a Small design.

Key words: Fast design, Small design, Cryptography, DES, FPGA, implementation, Pipeline, VHDL.

الملـــــــخص:

في هذه المذكرة نقدم دراسة لعملية زرعجهازيلخوارزمية التشفير، تشفير البيانات القياسية DES في دارة مبرمجة FPGA من نوع Spartan 3E XC3S100E , TQ144، تم استعمال تصميمين لعملية الزرع الاول يدعى التصميم السريع والذي استعملنا فيه نظرية التوازي مع التنبيه على عدم خطية المصفوفاتS-BOX، والتصميم الاخر هو التصميم المصغر و الذي يستهلك اقل قدر من مساحة الدارة المبرمجة، والمقصود بالسرعة في هذا العمل هو التدفق (عدد البيانات المعالجة في الثانية) والمقصود بالمساحة هي عدد CLB. بغرض الحد من استهلاك مساحة الدارة المبرمجة نظرا لاستعمالنا لدارة ضعيفة من حيث المسالك فان التصميم الثاني هو الأنسب لنا.

الكلمات المفتاحية: التصميم السريع، التصميم المصغر، التشفير، DES، FPGA، زرع ، المتوازية، VHDL.