République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

ECOLE NATIONALE POLYTECHNIQUE D'ALGER
Département d'Electronique

Mémoire de fin d'études

En vue de l'obtention du diplôme
d'ingénieur d'état en Electronique

THEME

CRYPTOGRAPHIE RSA

SUR FPGA

Présenté par :

Mr Abdelkader HALLAL
Mr Khalil EL ABED

Proposé par :

Mr R.SADOUN

Promotion: Juin 2008

Ecole Nationale Polytechnique
10, Avenue Hacen Badi, El-Harrach, Alger.

مـلـخـص

عملنا يتضمن اختزال خوارزمية نظام التشفير RSA على شريحة FPGA ، و تهيائها للحل Soc. اختزال لخوارزمية يتم بعدة طرق التسلسلي , المتوازي و السيستولي. اخترنا الاختزال السيستولي لانه يقدم افضل اتفاق بين المساحة و وقت التنفيد. الاختزال تم باستعمال بيئة التطوير ISE لـ Xilinx .

الكلمات المفتاحية : علم التشفير، RSA ، Montgomery ، السيستولية ، Soc ، FPGA ، خلية الملكية الفكرية.

Résumé

Notre projet a pour objet de l'implémentation de l'algorithme de cryptographie RSA sur FPGA et son adaptation à une solution SoC. L'implémentation de l'algorithme peut prendre diverses formes. Nous citons les implémentations sérielle, parallèle et systolique. Nous avons opté pour l'implémentation systolique car elle apporte un meilleur compromis entre la surface et le temps de traitement. L'implémentation a été réalisée sur l'environnement ISE/EDK de Xilinx.

Mots clefs : Cryptographie – RSA – Montgomery – systolique – SoC – FPGA – IP-Core.

Abstract

Our project consists on the implementation of the RSA cryptosystem on FPGA and it's adaptation to a SoC. The algorithm implementation can take several forms. We quote serial, parallel and systolic implementation. We chose systolic implementation because they it ensures the best compromise between the area and processing time. The implementation was done with ISE/EDK environment of Xilinx.

Key words: Cryptography – RSA – Montgomery – systolic – SoC – FPGA – IP-Core.

Remerciements

Nous tenons, avant tout, à remercier DIEU, tout clément, tout puissant, de nous avoir donné la force de réaliser notre travail.

Nos remerciements vont exceptionnellement à Monsieur R.SADOUN, chargé de cours à l'Ecole Nationale Polytechnique d'Alger, pour son aide, son suivi, ses conseils et directives et pour son dévouement.

Nous tenons à remercier Monsieur D.BERKANI, Professeur à l'Ecole Nationale Polytechnique d'Alger, d'avoir accepté de présider le jury.

Nos remerciements vont aussi à Mademoiselle A.MOUSSAOUI, d'avoir bien voulue accepter d'examiner notre travail.

Nous remercions tous les enseignants de l'Ecole Nationale Polytechnique d'Alger, spécialement ceux des département des Sciences Fondamentales et d'Electronique, pour leur apport en savoir.

Nos remerciements, vont au personnel de l'Ecole et à toute personne dévouée au service de l'Ecole Nationale Polytechnique.

Enfin, nos remerciements vont à toute personne ayant contribuée, de prés ou de loin, à réaliser ce travail.

Dédicaces

Je dédie ce modeste travail à ma mère, ma mère, ma mère..., à mon père et à ma chère sœur qui m'ont soutenu sans relâche, dans toutes les circonstances, tout au long de mon parcours d'études.

Je dédie, aussi, ce travail à mes grands parents, et à toute ma famille pour leur soutien.

Je dédie ce travail à tous mes ami(e)s pour leur soutien tout au long de mon parcours.

Enfin je dédie ce travail à ma chère et très spéciale promotion.

Khalil

Je dédie ce travail à mes parents, mes frères : Mohamed, Abdelhakim, Anis et ma sœur Hidaya ; ainsi qu'à ma grande mère et toute ma famille grands et petits.

Je dédie aussi ce travail à tous mes amis Khaled, Hakim ..., et à toute ma promotion

Abdelkader

Sommaire

Chapitre II : Cryptographie à clef publique (cas du RSA)

Chapitre III : Algorithme d'implémentation

Chapitre IV : Mise en œuvre

Liste des figures

Liste des tableaux

Glossaire

A

AES : Advanced Encryption Standard

ASIC : Application Specific Integrated Circuit

ANSI : Agence de Normalisation et de standardisation internationale

B

BSB : Base System builder

BRAM : block of random access memory

C

Cell : Cellule

CPLD : Complex Programmable Logic Device

CLB : Configurable Logic Bloc

D

DES : Data Encryption Standard

DSS : Digital Signature Standard

DLMB : Data Local Memory Bus

DOPB : Data On-Chip Peripheral Bus

DH : Protocole Diffie-Hellman

E

EDK : Embedded Development Kit

e-Cell : Exponentiator cellule

e-PE : Exponentiator Processor Element

EPLD : Erasable Programmable Logic Device

elf file : Executable and Linkable Format file

F

FPGA : Field Programmable Gate Array

FSL : Fast Link Simplex

FIFO : First In, First Out

G

GUI : Graphical User Interface

H

HW : Hard Ware

HA : Half Additionner

I

IOB : Input/Output Bloc

ISE : Integrated Software Environment

IP-Core : Intellectual Property Core

ISO : International Standards Organisation

ILMB : Instruction Local Memory Bus

IOPB : Instruction On-chip Peripheral Bus

J

JTAG : Joint Test Action Group

L

LSI : Last signifiant bit

LED : Light-Emitting Diode

LSB : Least significant bit

M

µP : Micro Processeur

µC : Micro Contrôleur

mod : Modulo

Mux : Multiplexer

mss file : Microprocessor Software Specification file

mhs file : Microprocessor Hardware Specification file

N

NIST : National Institute of Standards and Technology

O

OPB : On-chip Peripheral Bus

P

PE : Processor element

PLD : Programmable Logic Device

ParExp : Parallel Exponentiation

PC : Personnel Computer

R

RSA : Rivest, Shamir and Adleman

RS232 : Recommended Standard 232

RISC : Reduced Instruction-Set Computer

S

SAMMM : Systolic Architecture of Montgomery Modular Multiplication

SRAM : Synchrone Random Access Memory

SoC : System on Chip

SLI : System-Level Integration

SW : Soft Ware

SeqExp :Sequential Exponentiation

SysExp : Systolic Exponentiation

SDK : Software Development Kit

V

VHDL : Very high speed integrated circuit Hardware Description Language

VGA : Video Graphics Array

X

XCL : Xilinx CacheLink

XPS : Xilinx Platform Studio

Introduction générale

La cryptographie est un moyen de dissimuler l'information. Depuis les temps préhistoriques, elle était utilisée surtout pour des fins militaires et diplomatiques. Aujourd'hui, le champ d'application de la cryptographie s'est élargi et a trouvé un regain d'actualité avec les problèmes posés par la sécurité des transactions bancaires, la transmission de fichiers et de bases de données sous forme électronique.

Les systèmes de cryptographie, quant à eux, ont beaucoup évolué depuis leur apparition. En effet, on est passé des systèmes cryptographiques les plus simples tels que le chiffrement de César, de Vigenère... etc. à des systèmes, certes, plus complexes, mais surtout plus sûrs et ce en passant par la machine du Ché et ENIGMA. De nos jours, on parle de systèmes cryptographiques à clefs publiques et à clefs secrètes. Jusqu'à très récemment, les systèmes dits à clefs secrètes, réputés pour être extrêmement difficile à casser, étaient les plus utilisés, on cite le système triple DES et le système AES, néanmoins, la communication de la clef secrète posait problème. Dés lors, sont apparus les systèmes dits asymétriques (ou à clef publique).

Le système RSA, inventé en 1978, est de loin le système cryptographique asymétrique le plus courant. Son concept pour le cryptage/décryptage des messages se base sur un calcul d'exponentiation modulaire de très grands nombres. Cette opération est réputée pour être très coûteuse en temps; ce qui est un grand inconvénient pour une application en temps réel. Dans ce cas, les algorithmes de calcul rapide de l'exponentiation modulaire deviennent impératifs. L'algorithme de Montgomery, utilisant l'architecture systolique, s'impose comme une solution intéressante. Son implémentation matérielle devient alors une nécessité dès lors que des temps de calcul faibles sont recherchés. Son intégration dans un système sur puce (ou Soc) devient naturelle et même un atout. Notre travail s'inscrit dans cette dernière thématique. Le développement de ce dernier a suivi le cheminement suivant.

Le premier chapitre donne une introduction à la cryptographie, en partant de son historique, de ses techniques ainsi que des standards de chiffrement. Nous aborderons le cas particulier de la primalité dans la cryptographie.

Dans le second chapitre, on abordera la cryptographie asymétrique. On exposera les systèmes à clef publique existants ; nous nous intéresserons au cas spécifique du chiffrement RSA.

On abordera dans le troisième chapitre les implémentations possibles de l'algorithme RSA tenant compte des diverses architectures possibles.

Le dernier chapitre traitera du développement de notre implémentation. On présentera pour cela le circuit systolique implémenté et son intégration dans une architecture SoC.

CHAPITRE I

Introduction à la cryptographie

Chapitre I : Introduction sur la cryptographie

I.1 Introduction

Le passage de la préhistoire à l'Histoire s'est fait dès l'apparition de l'écriture. En effet, cette dernière a débuté, à partir du moment où l'on a pu conserver les grands faits de l'homme. Dès lors, la notion de confidentialité portant sur l'écriture a, vite, pris de l'élan, sans l'écriture la cryptologie n'aurait jamais existé.

Dès que la culture d'une société atteigne un certain niveau, mesuré par sa littérature, la cryptographie apparaît spontanément (comme ses parents l'écriture et le langage, l'ont probablement fait). Les besoins multiples de l'homme demandant de la confidentialité entre deux ou plusieurs personnes, au milieu de la société, conduit inévitablement à la cryptographie.

La cryptologie est apparue dans de nombreux domaines tels que l'armée, le commerce, la religion… Elle a donc énormément influencé le cours de l'histoire même si elle est restée dans l'ombre. Dans ce qui suit, on abordera quelques moments marquent l'histoire de la cryptographie, et on développera plus ses fondements.

I.2 Définition de la cryptographie :

La cryptographie, est la science qui emploie des méthodes mathématiques afin de dissimuler ses intentions ou ses instructions à ses ennemis et/ou de les transmettre à ses amis au moyen d'un texte chiffré. En face, chez l'adversaire, il s'agit de briser le code, de trouver le système qui préside à son élaboration : c'est la cryptanalyse.

Un algorithme cryptographique est une fonction utilisée lors du chiffrement et du déchiffrement, cet algorithme fonctionne avec une clef qui sera utilisée pour chiffrer le texte clair en un texte chiffré.

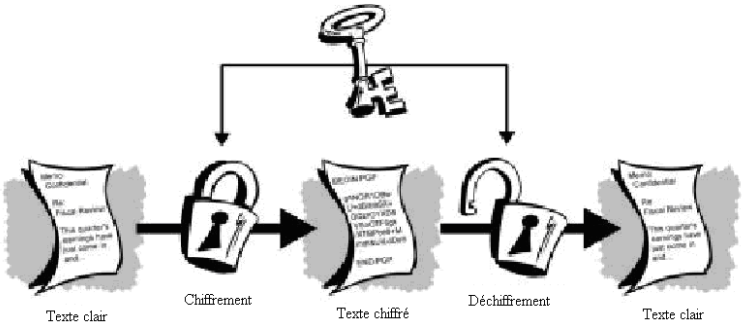

Figure I-1 : Système cryptographique

Un système cryptographique est un quintuplet $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ tel que :

- $\mathcal{P}$ est un ensemble fini de blocs de textes clairs possibles.
- $\mathcal{C}$ est un ensemble fini de blocs de textes chiffrés possibles.
- $\mathcal{K}$ est un ensemble fini de clefs possibles.
- $\mathcal{E}$ est un ensemble de transformations de chiffrement (chaque transformation étant indexée par une clef).
- $\mathcal{D}$ est un ensemble de transformations de déchiffrement.

Il existe deux sortes de cryptographie : cryptographie faible et cryptographie forte, la force de la cryptographie est mesurée par le temps et les moyens nécessaires pour trouver le texte clair.

I.3 Historique et origines de la cryptographie

La cryptographie a vu le jour pour la première fois il y a environ 4000 ans, au bord du Nil, où était utilisée par les scribes égyptiens pour raconter l'histoire de leurs maîtres, cependant elle était loin d'être utilisée pour des fins militaires.

On notera un évènement très marquant où on conçut pour la première fois, à Sparte (Grèce) un procédé de chiffrement militaire, des historiens grecs mentionnent l'utilisation de ce procédé par les Spartiates vers 475 avant J.C. Ultérieurement, les Grecs mettaient au point plusieurs précédés stéganographiques, ils sont aussi à l'origine du premier procédé de substitution, mis par l'écrivain grec Polybe, mais ce dernier n'a vu son utilisation que dans les opérations militaires menées par J.César.

Au moyen âge, la cryptologie évolue très faiblement car elle représentait peu d'importances, cependant, elle fut utilisée par les arabes. En 1467, le savant italien Leon Batista Alberti inventa la substitution polyalphabétique et mis en place le cadran chiffrant d'Alberti, il va plus loin et mis en place, après, un répertoire de 336 groupes de mots représentés par toutes les combinaisons allant de 11 à 4444. Ces découvertes ne seront utilisées que 400 ans plus tard. Depuis ce temps là, de nombreuses découvertes apportées par des savants européens ont enrichis la cryptographie, notamment la découverte de la notion de clef littérale par Giovanni Batista Belaso, qu'il appela "mot de passe", ou la découverte du "carré de Vigenère" qui pouvait dissimuler un message dans l'image d'un champ d'étoiles.

A l'aube du 20ème siècle, le savoir en cryptographie et cryptanalyse est important. C'est dans le domaine militaire que l'on verra le plus cette science des écritures secrètes. Durant la Seconde Guerre Mondiale, la cryptographie connût un développement considérable notamment avec l'utilisation de la machine ENIGMA.

La machine ENIGMA

L'histoire débute, en 1923, lorsque la Cipher Machine Corporation montre pour la première fois au Congrès Postal International à Bern en Suisse, la machine de codage ENIGMA, modèle A. Le modèle est lourd et volumineux, un clavier de machine à écrire (de type QWERTY) est utilisé pour la saisie des messages. Dans les faits, la machine pouvait être utilisée comme une machine à écrire standard et cela même en plein milieu de l'encodage d'un texte. ENIGMA A ne connue pas un très grand succès malgré la publicité faite à cette époque. Par la suite, trois autres modèles apparurent, soit les modèles B, C et D. Le modèle B est similaire au modèle A à l'exception des rotors qui ont maintenu 26 contacts au lieu de 28 pour le modèle A. Les modèles C et D étaient portables et cryptographiquement différents des modèles précédents. Ces derniers fonctionnent selon des principes identiques à ceux des machines de Hebern, mais avec néanmoins quelques différences importantes.

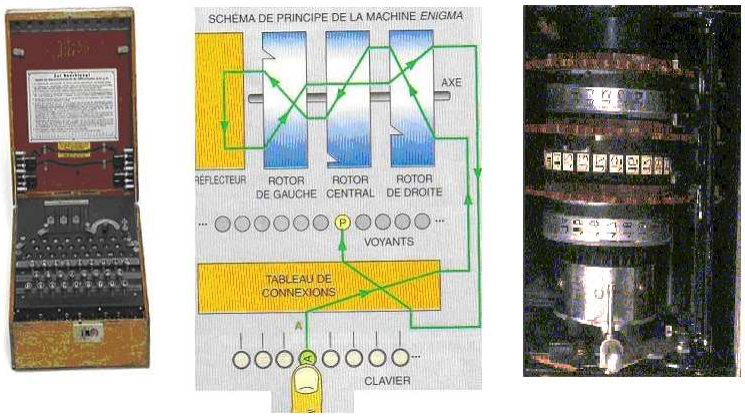

Figure I-2 : La machine ENIGMA

ENIGMA ressemble à une machine à écrire. Le principe de chiffrement qu'utilise ENIGMA est à la fois simple et astucieux. Simple, car il ne s'agit d'une simple substitution de lettres : par exemple, A devient Q, P devient N, etc. Et astucieux, parce que la substitution change d'une lettre à une autre : si la lettre A correspond à Q la première fois qu'on la saisit, elle pourrait correspondre à M, K, H, ou tout autre lettre différente de Q à la fois suivante (ce

principe est possible grâce au système de rotors). De plus, un autre avantage non négligeable que possède ENIGMA est la réversibilité : si on tape le message clair, on obtient le message code, et avec le message codé, on obtient le message clair.

Le chiffre du Che

Le révolutionnaire marxiste d'Amérique latine Ernesto Rafael Guevara de la Serna alias "Che Guevara" fut exécuté par l'armée bolivienne, le 8 octobre 1967, on retrouva sur son corps un papier indiquant le procédé qu'il utilisait pour échanger des messages avec Fidel Castro.

Le procédé passait par 4 étapes :

Dans la première, on utilisait une substitution de lettres par des chiffres telle que l'est montrée dans le tableau **I-1**, dans la seconde, on découpait les chiffres du message en blocs de 5 chiffres. Après, dans la troisième étape, on additionnait en modulo 10 le message obtenu avec un nombre de 5 chiffres connu de lui et de Fidel Castro seulement. Et en dernier lieu, on envoyait tous les blocs cryptés et détruisait le brouillon ayant servi à faire les calculs.

Clair	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Code	6	38	32	4	8	30	36	34	39	31	78	72	70	76	9	79	71	58	2	0	52	50	56	54	1	59

Tableau I-1 : Table de Che

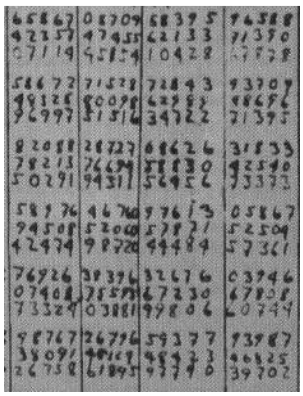

Figure I-3 : Brouillon utilisé par Che Guevara

Ces vingt années ont été riches en découvertes, en innovations et en progrès techniques, les uns décisifs, les autres d'importance plus relative. Parmi les faits les plus marquants de la courte histoire de la cryptologie moderne, on cite la mise au point du premier système à clef publique, par Rivest, Shamir et Adleman.

De tout temps, la cryptologie fut utilisée que par les gouvernements, les services secrets et les armées. Ainsi, très peu de gens, voire personne n'utilisait la cryptographie à des fins personnelles, ce qui a fait de la cryptologie une science secrète.

De nos jours, la cryptologie est de plus en plus utilisée sur le réseau mondial Internet. Avec l'apparition du commerce en ligne, c'est-à-dire la possibilité de commander des produits directement sur Internet, la cryptographie est devenue nécessaire. D'autre part, on remarque de plus en plus d'entreprises qui s'échangent des informations confidentielles, en tête les banques, c'est lors que la cryptologie entre en jeu.

I.4 Techniques cryptographiques [1] [2]

I.4.1 Chiffrement par décalage

On peut utiliser le chiffrement par décalage pour chiffrer un texte ordinaire en décidant d'une correspondance entre les caractères alphabétiques et les résidus modulo 26 comme ceci :

A	B	C	D	E	F	G	H	I	J	K	…	S	T	U	V	W	X	Y	Z
0	1	2	3	4	5	6	7	8	9	10	…	18	19	20	21	22	23	24	25

Tableau I-2 : Table Du chiffrement par décalage

On ajoute à chaque valeur la valeur de la clef (en modulo 26) et on convertit, à l'aide de la table du chiffrement par décalage les nombres, en caractères alphabétiques. Le chiffrement par décalage n'est pas sûr, car il peut être cryptanalysé par la méthode de recherche exhaustive, comme il n'y a que 26 clefs possibles, essayer le déchiffrement avec toutes les clefs jusqu'à trouver un texte clair compréhensible est aisé.

Remarque : pour la clef particulière K=3, le système cryptographique est appelé chiffrement de César, car il était utilisé par Jules César.

I.4.2 Chiffrement par substitution

Ce procédé a été utilisé pendant des centaines d'années. Ici, on fait correspondre chaque caractère de l'alphabet avec un autre, dans ce cas, une recherche exhaustive est impossible car le nombre de clefs cette fois est 26 !, cependant ce chiffrement est facile à cryptanalyser par une autre méthode.

I.4.3 Chiffrement de Vigenère

L'idée de Vigenère est d'utiliser un chiffre de César, mais où le décalage utilisé change de lettres en lettres. Pour cela, on utilise une table composée de 26 alphabets, écrits dans l'ordre, mais décalés de ligne en ligne d'un caractère. On écrit encore en haut un alphabet complet, pour la clef, et à gauche, verticalement, un alphabet complet, pour le texte à coder :

Figure I-4 : Table de Vigenère

Pour coder un message, on choisit une clef qui sera un mot de longueur arbitraire. On écrit ensuite cette clef sous le message à coder, en la répétant aussi souvent que nécessaire pour que, sous chaque lettre du message à coder, on trouve une lettre de la clef. Pour coder, on regarde, dans le tableau, l'intersection de la ligne de la lettre à coder avec la colonne de la lettre de la clef.

I.4.4 Masque jetable

Identique au chiffrement de Vigenère mais on n'utilise la clef qu'une seule fois, ce qui contraint à choisir une clef aussi longue que le message à transmettre.

I.4.5 Chiffrement de Hill

Inventé en 1929 par Lester S. Hill. L'idée consiste à transformer m caractères d'un bloc du texte clair en m caractères d'un bloc du texte chiffré par des combinaisons linéaires. Si m=2, pour un bloc de texte clair : $x = (x_1, x_2)$, on obtient un bloc de texte chiffré : $y = (y_1, y_2)$, où y_1 et y_2 sont obtenus par combinaisons linéaires de x_1 et x_2.

I.4.6 Chiffrement par permutation

L'idée est de conserver les mêmes caractères en les réordonnant. Appelé aussi chiffrement par transposition, il a été utilisé pendant des centaines d'années.

I.4.7 Chiffrement en chaine

Dans les systèmes cryptographiques précédents, les éléments du texte clair sont chiffrés de la même manière à partir de la clef K. En effet, la chaine du texte chiffré y est obtenue : $y = y_1 y_2 \ldots = e_K(x_1)e_K(x_2) \ldots$, les procédés de ce type sont appelés chiffrement par bloc.
Une autre approche consiste à utiliser la notion de chiffrement en chaine. L'idée de base consiste à engendrer une séquence de clefs : $z = z_1 z_2 \ldots$, et à l'utiliser pour chiffrer la chaine $x = x_1 x_2 \ldots$ suivant la règle : $y = y_1 y_2 \ldots = e_{z_1}(x_1)e_{z_2}(x_2) \ldots$

I.4.8 Surchiffrement

C'est la composition de deux méthodes de chiffrement. Le message chiffré par la première méthode est utilisé comme 'message' clair pour la deuxième méthode.

I.5 Standards de chiffrement [1] [2]

Un algorithme de chiffrement transforme un message, appelé texte clair, en un texte chiffré qui ne sera lisible que par son destinataire légitime. Cette transformation est effectuée par une fonction de chiffrement paramétrée par une clef de chiffrement. Une personne connaissant la clef de déchiffrement peut, en utilisant la fonction de déchiffrement, déchiffrer le texte chiffré. En conséquence, la sécurité d'un algorithme de chiffrement doit uniquement reposer sur le secret de la clef de déchiffrement. Par ailleurs, le fait de rendre publiques les méthodes de chiffrement et de déchiffrement offre une certaine garantie sur la sécurité d'un système, dans la mesure où tout nouvel algorithme cryptographique est immédiatement confronté à la sagacité de la communauté scientifique.

Un système de chiffrement est dit par blocs s'il divise le texte clair en blocs de taille fixe et chiffre un bloc à la fois. La taille des blocs est généralement de 64 ou de 128 bits. On distingue deux grands types d'algorithmes de chiffrement, les algorithmes à clef secrète et les algorithmes à clef publique. Chacune de ces deux classes possède ses propres avantages et inconvénients. Les systèmes à clef secrète nécessitent le partage d'un secret entre les interlocuteurs. La découverte

en 1976 des systèmes à clef publique a permis de s'affranchir de cette contrainte, mais elle n'a pas pour autant apporté de solution parfaite, dans la mesure où tous les algorithmes de chiffrement à clef publique, de par leur lenteur, ne permettent pas le chiffrement en ligne. Dans la plupart des applications actuelles, la meilleure solution consiste à utiliser un système hybride, qui combine les deux types d'algorithmes.

I.5.1 Chiffrement à clef secrète

Les algorithmes à clef privée sont aussi appelés algorithmes symétriques. En effet, lorsque l'on crypte une information à l'aide d'un algorithme symétrique avec une clef secrète, le destinataire utilisera la même clef secrète pour décrypter. Il est donc nécessaire que les deux interlocuteurs se soient mis d'accord sur une clef privée auparavant, par courrier, par téléphone ou lors d'un entretien privé.

I.5.1.1 Le DES

Abréviation de **D**ata **E**ncryption **S**tandard, ce système de chiffrement à clef secrète est, jusqu'à très récemment, le plus utilisé, adopté comme standard américain en 1977 pour les communications commerciales, puis par l'ANSI en 1991.

Le DES utilise une clef sur 56 bits, pour chiffrer des blocs de 64 bits, les blocs chiffrés ayant aussi 64 bits. L'algorithme se déroule en trois étapes :

1. Une chaine de bits x_0 est construite à partir du texte clair x, en changeant l'ordre des bits de x suivant une permutation initiale IP fixée. On écrit $x_0 = IP(x) = L_0R_0$. Où L_0 contient les 32 premiers bits de la chaine x_0 et R_0 contient les 32 restants.
2. 16 itérations d'une certaine fonction sont effectuées. On calcule L_iR_i, suivant la règle

   $$L_i = R_{i-1} \text{ et } R_i = L_{i-1} \oplus f(R_{i-1}, K_i)$$

   Où $\oplus$ représente le ou-exclusif bit-à-bit de deux chaines. f est une fonction décrite plus loin, et $K_1, K_2, K_3 \ldots K_{16}$sont obtenus par diversification de la clef.
3. La permutation inverse IP^{-1} est appliquée à $L_{16}R_{16}$ pour obtenir le bloc de texte chiffré $y = IP^{-1}(R_{16}L_{16})$.

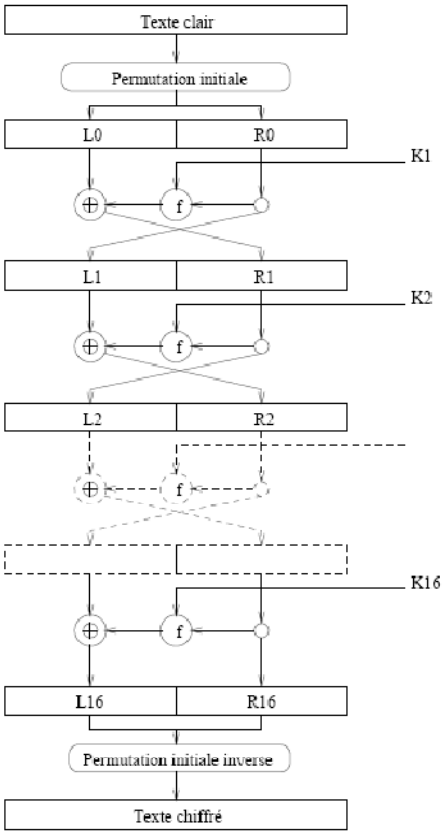

Figure I-5 : Schéma général du DES

Le DES est malheureusement vulnérable aux attaques exhaustives. C'est pourquoi la plupart des applications l'utilisent maintenant sous la forme d'un triple DES à deux clefs. Cette technique permet de doubler la taille de la clef secrète (112 bits). On effectue d'abord un chiffrement DES paramétré par une première clef de 56 bits, puis un chiffrement DES paramétré par une seconde clef, et à nouveau un chiffrement DES avec la première clef. Le triple DES à deux clefs a notamment été adopté dans les standards ANSI X9.17 et ISO 8732. Il est extrêmement utilisé pour les applications bancaires.

I.5.1.2 L'AES

L'AES, abréviation de **A**dvanced **E**ncryption **S**tandard, encore appelé RIJNDAEL, est le nouveau standard de chiffrement à clef secrète conçu par deux cryptographes belges, V.Rijmen et J.Daemen. 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).

L'algorithme opère sur des blocs de texte de 128 bits, trois tailles de clefs sont possibles 128, 192 et 256 bits. Pour une clef de 128 bits, L'AES effectue 10 itérations de la fonction

montrée sur la Figure **I-6**, la première itération est précédée par un ou-exclusif entre le texte clair et la sous-clef numéro 0.

Une fonction itérée se compose de trois étapes :

1. Etape de confusion : on applique à chaque octet du message une permutation dite S, cette dernière est la fonction inverse dans un corps fini à 2^8 éléments, elle assure la résistance de l'algorithme aux attaques classiques.
2. Etape de diffusion : on effectue une permutation des bits par une fonction P qui est composée de fonctions simples dans le corps fini à 2^8 éléments.
3. On effectue un ou-exclusif bit-à-bit entre le message résultant et la sous-clef de l'itération.

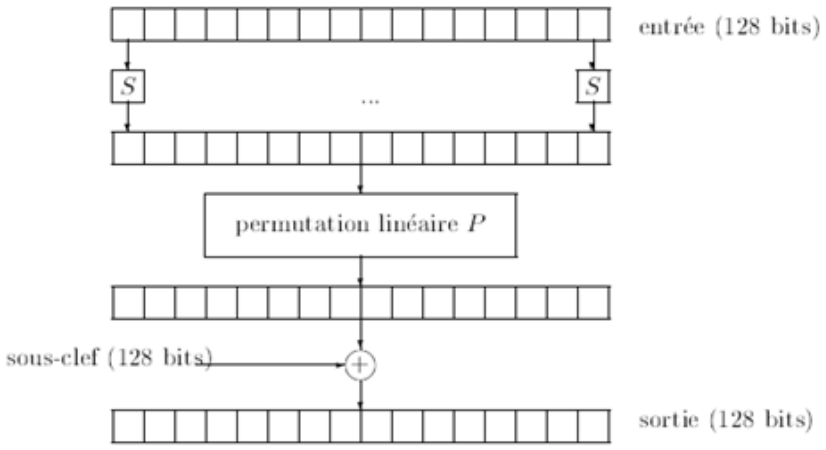

Figure I-6 : Une itération de l'AES

Les sous-clefs de 128 bits sont obtenues de la clef secrète de la manière suivante : la sous-clef numéro 0 est elle même la clef secrète, la sous-clef numéro i est obtenue de la clef secrète en utilisant l'algorithme décrit sur la figure **I-7.**

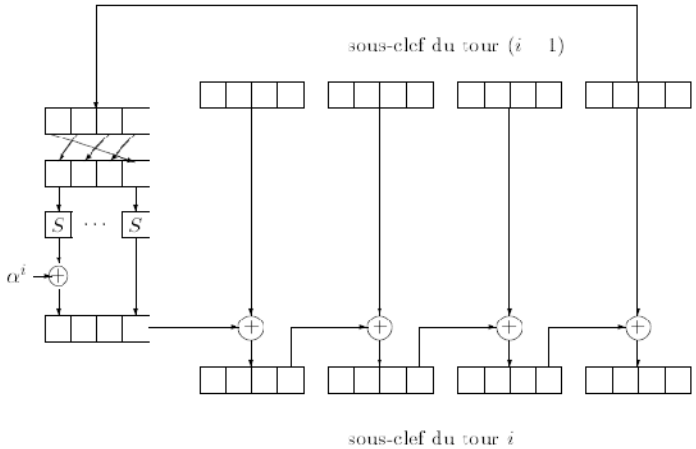

Figure I-7 : Algorithme de cadencement de clef de l'AES

L'AES, composé d'opérations simples sur les octets, est réputé pour être extrêmement rapide, à la fois pour les processeurs 8 bits utilisés dans les cartes à puce et pour les implémentations logicielles. Il atteint par exemple un débit de chiffrement de 70 Mbits/s pour une implémentation en C++ sur un Pentium à 200 MHz.

I.5.2 Chiffrement à clef publique

L'idée de système à clef publique, qui date de 1976, est due à Diffie et Hellaman. La première réalisation d'un système à clef publique fut publiée en 1977 par Rivest, Shamir et Adleman : c'est le chiffrement RSA, que l'on va étudier dans le prochain chapitre.

I.6 Cryptographie et primalité [1]

Dans la cryptographie, 3 problèmes se posent quant à la primalité d'un chiffre :

- Prouver que N est premier
- Prouver que N est décomposable (non premier)
- Factoriser N s'il est décomposable.

Alors l'algorithme basique le plus utilisé est de diviser N successivement sur 1,2,3..$\sqrt{N}$, si on ne trouver aucun diviseur, on aurait démontré que N était premier, dans le cas contraire, on dira que N n'est pas premier (décomposable) et en répétant la même opération sur le diviseur on aurait factorisé le nombre N. Le problème qui se pose est la complexité pour des nombres N très grands. Pour ce faire, il existe plusieurs algorithmes :

I.6.1 Test de Miller-Rabin

Ce test est utilisé pour vérifier la non primalité d'un nombre N. Le test s'énonce comme suit : On choisit 20 nombres au hasard. Si pour tout a, l'algorithme répond "N est premier" alors la probabilité d'erreur est de $1/4^{20}$ ($\simeq 10^{-12}$). Cette marge d'erreur (quasi nulle, voire nulle) fait que l'on appelle de tels N des premiers industriels. D'autre part, s'il existe un a tel que l'algorithme général réponde "N n'est pas premier", alors N est composable.
Ce test ne donne donc absolument pas de preuve que N est premier.

I.6.2 Théorème de Pocklington-Lehmer

Cet algorithme démontre qu'un nombre N est premier.
Supposons que pour tout diviseur premier l de N-1, on puisse trouver des entiers a_p tels que :

$a_p^{N-1} \equiv 1\ mod\ N$ et $\left(a_p^{\frac{N-1}{p}} - 1\right) \wedge N = 1$, Alors N est premier.

I.6.3 Méthode ρ de Pollard

Cette méthode est utilisée pour la factorisation d'un nombre N.

Données : N ≥ 2, N prouvé composé par Miller-Rabin
Résultat : donne un diviseur de N
$x \leftarrow 1$
$y \leftarrow 1$
$d \leftarrow 1$
Tant que $d \leq 1$ faire
$x \leftarrow (x^2 + 1) mod\ N$
$y \leftarrow (y^4 + 2 * y^2 + 1) mod\ N$
$d \leftarrow (x - y) \wedge N$
Fin tant que
Retourner d

Figure I-8 : Algorithme ρ de Pollard

I.6.4 Méthode p-1 de Pollard

Cette méthode date depuis 1974, c'est une méthode de factorisation, l'énoncé de l'algorithme est le suivant :

Données : n, B et g
$a \leftarrow g$
Pour $j = 2$ jusqu'à B faire
$a \leftarrow a^j mod\ n$
$d \leftarrow pgcd\ (a - 1, n)$
Si $1 < d < n$ alors
D est un facteur de n (succés)
Sinon
Aucun facteur trouvé (échec)

Figure I-9 : Algorithme ρ-1 de Pollard

I.7 Conclusion

Les techniques cryptographiques, vues dans ce chapitre, paraissent traditionnelles, ceci est dû à la médiocre importance qu'on lui a donné nos ancêtres. Ce n'est qu'à l'aube du 20ème siècle, lors de la première guerre mondiale, où la cryptographie a pris son essor avec l'apparition de la machine ENIGMA. Depuis le temps, la recherche n'a jamais cessé pour mettre en œuvre des systèmes cryptographiques de plus en plus surs, cette recherche a aboutit aux systèmes cryptographiques à clef secrète, on cite parmi eux DES, AES...etc. et à clef publique, cette dernière qu'on détaillera plus dans le prochain chapitre.

CHAPITRE II

Cryptographie à clef publique

(Cas du RSA)

Chapitre II : Cryptographie à clef asymétrique (cas du RSA)

II.1 Introduction

Considéré comme difficile à percer, le DES (standard de chiffrement de données à clef secrète) a été adopté par le ministère de la défense des états unis, mais aussi, plus tard par plusieurs établissements bancaires. Cependant, son inconvénient majeur était le problème d'échange de clef secrète au préalable. La cryptographie à clef publique, quant à elle, a été inventée par Whitfield Diffie et Martin Hetlman en 1976 pour résoudre ce problème.

Dans ce chapitre, Nous allons présenter cette notion de cryptographie à clef publique, citer les systèmes à clef publique les plus connus, et enfin procéder à l'étude du système à clef publique RSA, le plus répondu à l'heure actuelle.

II.2 Définition du chiffrement à clef asymétrique

Le système de chiffrement à clef publique, où encore appelé chiffrement asymétrique, date de 1976, il est du à Whitfield Diffie et Martin Hellman. Dans ce système, la clef existe par paire, une clef publique pour le chiffrement et une autre secrète pour le déchiffrement.

Lorsqu'un utilisateur désire envoyer un message à un autre, il lui suffit de chiffrer le message à envoyer au moyen de la clef publique du destinataire (qu'il trouvera par exemple dans un serveur de clefs). Ce dernier sera en mesure de déchiffrer le message à l'aide de sa clef privée (qu'il est seul à connaitre).

Une notion importante dans le chiffrement à clef publique est celle de fonction à sens unique avec trappe. Une fonction est appelée à sens unique si elle est facile à calculer mais impossible à inverser. Une telle fonction est dite à trappe si le calcul de l'inverse devient facile dès que l'on possède une information supplémentaire (la trappe).

Il est très simple de construire un système de chiffrement à clef publique à partir d'une fonction à sens unique avec trappe. La procédure de chiffrement consiste simplement à appliquer la fonction au message clair. La fonction étant à sens unique, il est très difficile de l'inverser, c'est-à-dire de déterminer le message clair à partir du message chiffré, sauf si on connait la trappe, qui correspond à la clef secrète du destinataire. Toute la difficulté réside donc dans la

recherche de ces fonctions très particulières. Leur construction s'appuie généralement sur des problèmes mathématiques réputés difficiles.

II.3 Les systèmes à clef asymétrique [1] [13]

II.3.1 Le protocole de Diffie-Hellman-Merkle

Durant les années 1970, Ralph Merkle, étudiant à Berkley aux états unis, présente ses idées concernant la communication de la clef secrète à travers une ligne publique à Martin Hellman de Standford, ce dernier fut rejoint par Whitfield Diffie. Après deux ans de recherche concentrée sur l'arithmétique modulaire et les fonctions univoques, le trio mit en œuvre le premier système de cryptographie à clef publique (plus connu sous le nom de Diffie-Hellman ou DH). Cette méthode fut employée dans les protocoles Internet tels que le Secure Socket Layer et Internet Protocol Security.

L'algorithme s'étale sur 4 étapes :

Première étape : on crée 3 nombres :

- Un grand nombre premier (P) d'une longueur de plus de 1024 bits appelé le module.
- Un nombre (g) inférieur au module appelé la base.
- Un nombre aléatoire (x_1) d'environ 160 bits appelé exposant privé.

Ensuite, on calcule le 4ème ($\mathbf{y_1}$) appelé la valeur publique en effectuant : $y_1 = g^{x_1} \ mod \ P$. On communique au destinataire les trois nombres P, g et y_1, et garde pour lui x_1.

Deuxième étape : Le destinataire choisit au hasard un exposant aléatoire privé ($\mathbf{x_2}$) et calcule une valeur publique$y_2 = g^{x_2} \ mod \ P$.

Troisième étape : Le destinataire calcule la clef secrète $S = {y_1}^{x_2} \ mod \ P$ et nous transmet sa valeur publique y_2.

Quatrième étape : On calcule la clef secrète $S = {y_2}^{x_1} \ mod \ P$ (qui sera la même que celle que le destinataire a calculé).

Toute personne espionnant l'opération (c'est-à-dire possédant les quatre paramètres communiqués : y_1, y_2, g et P) ne pourra calculer la clef secrète S. Cependant un inconvénient de la méthode DH est qu'il ne permet pas de signer des documents. C'est pour cette raison que Diffie-Hellman est souvent associé à DSS (Digital Signature Standard, un autre algorithme). DSS permet de signer les documents.

II.3.2 Le système d'EL Gamal

EL Gamal a proposé un système cryptographique basé sur le problème du logarithme discret. Le problème du logarithme discret est l'objet de nombreuses études. Le problème est réputé difficiles si p est convenablement choisi. En fait, on ne connait aucun algorithme polynomial pour le résoudre. Pour éviter les attaques connues, p doit avoir au moins 150 chiffres, et $p-1$ doit avoir au moins un grand facteur premier. L'utilité du problème du logarithme discret en cryptographie provient du fait que calculer des logarithmes discrets est difficile, tandis que calculer l'opération inverse d'exponentiation peut se faire efficacement avec l'algorithme d'exponentiation modulaire. En d'autres termes, l'exponentiation modulo p est une fonction à sens unique pour des nombres premiers p convenables.

On suppose que p est un nombre premier et que α est une racine primitive modulo p. p et α étant fixés, le problème du logarithme discret se réduit à trouver l'unique exposant a, $0 \leq a \leq p-2$ tel que $\alpha^a = \beta\ (mod\ p)$. Pour ce faire, plusieurs algorithmes existent (algorithme de Shanks, algorithme de Pohlig-Hellman…).

Le chiffrement d'EL Gamal est non déterministe, car l'opération de chiffrement dépend du message à crypter et d'une valeur aléatoire choisie au hasard (k). Il y a donc plusieurs textes chiffrés qui correspondent à un même texte clair.

Au début on choisit au hasard k, et on calcule $y_1 = \alpha^k mod\ p$ et $y_2 = x\beta^k mod\ p$. On transmet le texte chiffré $y = (y_1, y_2)$, à la réception, on tire le texte clair par :
$x = y_2 ({y_1}^{\alpha})^{-1} mod\ p$.

II.3.3 Corps finis et courbes elliptiques

Cette méthode s'agit d'une méthode d'échange de clef secrète à travers un canal publique d'une manière semblable à celle de Diffie-Hellman, ici, on procède au choix d'une courbe elliptique $E(a, b, K)$ publiquement, c'est-à-dire un corps fini K et d'une courbe elliptique : $y^2 = x^3 + ax^2 + b$ (tel que $(4a^3 + 27 \neq 0)$, et d'un point de la courbe noté P. Les deux parties choisissent chacun un nombre secret respectivement K_a et K_b, puis se communiquent entre eux les nombres qu'ils ont calculé auparavant, respectivement $P.K_a$ et $P.K_b$, puis chacun calcule le produit du nombre qu'il l'a choisit secrètement et le nombre communiqué par l'autre personne. Toute personne espionnant la communication, connaît $E(a, b, K)\ P, K_a\ et\ K_b$, mais pour calculer la clef secrète, l'espion devra calculer K_a depuis P et $P.K_a$, c'est ce que l'on appelle résoudre le logarithme discret sur la courbe elliptique. Le logarithme discret est déjà difficile à résoudre dans les groupes bien connus(Z/PZ). Pour les courbes elliptiques, c'est encore plus difficile.

Pour s'échanger un message, les deux personnes doivent se mettre d'accord sur la façon de représenter un texte en points sur la courbe elliptique. Pour transmettre un point M de la courbe elliptique, la personne doit choisir un nombre secret l et transmettre à l'autre le couple $(lP, M + lK_bP)$, le destinataire lui multiplie lP par $K_b$et retranche le résultat de $M + lK_bP$, et retrouve M. Tout espion ne connaissant pas K_b, ne pourra pas retrouver M .

Cette méthode présente autant d'avantages que d'inconvénients, certes, une clef de 200 bits pour les courbes elliptiques est plus sure qu'une clef de 1024 bits pour le RSA, d'autre part un appareil de signature doit stocker la publique, une clef publique RSA occupe 256 octets, cependant, son homologue occupe à peine 20 octets, cette différence de taille peut être un facteur important. Pour ce qui concerne ces inconvénients, la signature avec les courbes elliptiques est moins rapide que la signature RSA sauf si l'on utilise des valeurs calculées au préalable, donc on doit bénéficier de tableaux de valeurs calculées dans les appareils de signatures, qui arrivent à occuper jusqu'à 20 000 octets. Aussi un autre inconvénient, est que la technologie de cryptographie par courbes elliptiques a fait l'objet du dépôt de nombreux de brevets, cela rendrait son utilisation vite coûteuse. [5]

II.3.4 Le chiffrement de Rabin

C'est un système de cryptage à clef publique inventé par Michael Rabin en 1979, qui s'appuie sur le même principe que RSA, c'est-à-dire factoriser un très grand nombre en deux nombres premiers. Pour la génération de la clef, on procède au choix de deux nombres premiers très grands P et Q, et qui soient congrus à 3 modulo 4, on calcule la clef publique $n = P.Q$, puis on choisit un nombre au hasard compris entre 0 et $n - 1$, le texte chiffré y s'obtient du texte clair **x** comme suit : $y = x(x + B)\ mod\ n$**,** lors du déchiffrement, le texte clair est obtenu par $x = \sqrt{\frac{B^2}{4} + y} - \frac{B}{2}$. Trouver la racine carrée $\sqrt{\frac{B^2}{4} + y}$ modulo n revient, comme le dit le théorème des restes chinois, au même que trouver la racine carrée $\sqrt{\frac{B^2}{4} + y}$ modulo P et la racine carrée $\sqrt{\frac{B^2}{4} + y}$ modulo Q. C'est ici qu'intervient le fait que P et Q doivent être congrus à 3 modulo 4. En effet, dans ce cas, il existe une formule simple:

Si P mod 4=3, alors la solution de $C^{1/2}\ mod\ P$ est$\mp C^{(P+1)/4}\ mod\ P$.

Bien que le chiffrement de Rabin dispose d'une preuve de difficulté aussi grande que la factorisation d'entiers, preuve qui n'existe pas encore pour RSA, il présente néanmoins l'inconvénient qu'il existe 4 textes clairs pouvant se chiffrer en un même texte chiffré.

II.4 Le système RSA

Le RSA est un algorithme de cryptographie à clef publique qui a été inventé en 1977 par Ron Rivest, Adi Shamir et Len Adleman, d'où le sigle RSA. La découverte de RSA a été fortuite, au départ Rivest, Shamir et Adleman voulaient prouver que chaque système à clef publique possédait une faille. A présent, le brevet de cet algorithme appartient à la société américaine RSA Data Dynamics et aux Public Key Parteners qui détiennent les droits en général sur les algorithmes à clef publique.

II.4.1 Génération des clefs avec RSA [1] [2]

L'algorithme RSA se base essentiellement sur la difficulté de factoriser un très grand nombre en deux nombres premiers. Donc partons du fait qu'on va choisir deux grands nombres premiers (de l'ordre de 200 chiffre) de manière totalement aléatoire et appelons les P et Q, pour ce faire il existe des algorithmes de génération aléatoire de nombres premiers. L'étape suivante est de calculer le produit des deux nombres : $n = P.Q$, puis chercher le nombre e de telle manière à ce que $2 < e < \varphi(n)$. $\varphi(n)$ est la fonction indicatrice d'Euler, elle représente le nombres d'entiers inférieurs à n et qui sont premiers avec lui, on choisira e de telle sorte qu'il soit lui-même premier avec $\varphi(n)$. Il est facile à démontrer que $\varphi(n) = (P-1)(Q-1)$ puisque P et Q sont premiers. Le couple (n, e) forme la clef publique de l'algorithme RSA. Pour ce qui est de la clef secrète d, la théorie de RSA dit que d se calcule de manière à ce que $(e.d-1)$ soit divisible par $\varphi(n)$, d'après le théorème de Bezout, il existe (d, k) dans Z tel que $e.d + k.\varphi(n) = 1$. La solution par l'algorithme d'Euclide nous donne un ensemble de solution d, $d = r.\varphi(n) + d_0, r$ entier, cet ensemble est une classe de modulo $\varphi(n)$, et par conséquent, il existe une seule solution d comprise entre 2 et $\varphi(n)$, $d = d_0$.

Désormais le couple (n, e) représente la clef publique, et (n, d) la clef secrète. Pour les nombres P, Qet $\varphi(n)$, ils ne servent à présent à rien, il est recommandé de les conserver secrets, soit les détruire.

II.4.2 Algorithme de cryptage et de décryptage avec RSA

Pour crypter un document, on doit déjà le mettre sous la forme d'un nombre m (m étant le texte clair désormais) inférieur à n (la clef publique), le texte chiffré C sera calculé à partir du texte clair et de la clef publique, $C = m^e \ mod \ n$.

Une fois reçue, le texte chiffré C subit l'opération inverse que pour le chiffrement, le texte clair m est calculé depuis le texte chiffré et la clef secrète, $m = C^d \ mod \ n$. On trouvera bien que le texte obtenu est identique au texte clair au départ.

II.4.3 Authentification d'un document avec RSA [1]

L'authentification d'un document est l'opération qui consiste à donner au document une emprunte de son auteur, cette opération peut s'avérer très utile en cas de litige. Pour un document numérique, l'authentification consiste à apposer au document une signature digitale que l'on cryptera par exemple avec RSA pour empêcher sa falsification.

Malgré son apparente simplicité, Le système RSA reste l'un des algorithmes les plus surs, Néanmoins, il présente deux principaux inconvénients : le problème principal est sa lenteur par rapport à DES. La transmission de gros fichiers est donc particulièrement lente. Le second problème est qu'il n'est pas aussi sûr que DES, les algorithmes de factorisation sont de plus en plus performants et il est nécessaire d'employer des nombres premiers de plus en plus longs. Cependant plus la clef est longue, plus le chiffrement et le déchiffrement sont longs, ce qui fait que l'on a intérêt à chiffrer des messages assez courts.

II.5 Les attaques sur RSA

La résistance d'un document crypté avec RSA s'appuie sur le fait qu'il est très difficile de factoriser un très grand nombre en deux nombres premiers, l'attaque consiste donc à trouver des algorithmes de factorisation performant et en même temps rapide. Cela fait 25 ans qu'on essaye de casser RSA. En 2005, le plus grand nombre factoriser était long de 663 chiffres. Pour une taille de 512 bits, et depuis 1999, il faut faire travailler plusieurs centaines d'ordinateurs en parallèle. Cependant, les clefs utilisées habituellement pour RSA sont sur 1024 voire 2048 bits.

II.5.1 Attaque de Wiener

L'attaque de Wiener élaborée en 1989, elle n'est exploitable que si l'exposant secret (la clef secrète) d est inférieur à $n^{1/4}$, dans ce cas, l'exposant secret peut être retrouvé à l'aide du développement en fractions continues d'e/n.

II.5.2 Attaque de Hastad

L'attaque de Hastad est parmi les premières attaques qui furent élaborées, elle date de 1985, cette attaque peut repose sur la possibilité que l'exposant publique (la clef publique) e soit assez petit. Dans ce cas, en interceptant le même message chiffré envoyé à plusieurs destinataires, il est possible de retrouver le message clair à l'aide du théorème des restes chinois.

II.5.3 Attaque par chronométrage

L'Attaque par chronométrage décrite pour la première fois en 1995 par Kocher, elle suppose qu'en connaissant le matériel du destinataire du message, on peut déduire la clef secrète en mesurant les temps de déchiffrement de plusieurs documents chiffrés. Une solution pour contrecarrer cette attaque, est la technique d'aveuglement cryptographique (blinding). L'aveuglement se sert de la propriété multiplicative de RSA, en insérant dans le calcul une valeur choisie au hasard que l'on peut annuler par la suite. Cette valeur est différente chaque fois, de telle manière, le temps de déchiffrement ne soit pas corrélé aux données chiffrées.

II.5.4 Attaque par chiffrement choisis

En 1998, Daniel Bleichenbacher décrit la première attaque pratique de type 'chiffré choisi adaptable', contre des messages RSA en raison de défauts dans le schéma de remplissage, dés lors, RSA doit toujours être utilisé par une table de remplissage afin d'éviter qu'une valeur du message, une fois chiffrée, ne donne un résultat peu sur.

II.6 Conclusion

Le chiffrement RSA, présenté dans ce chapitre, est fondé sur l'hypothèse qu'il est très difficile de factoriser un nombre très grand en deux nombres premiers, ceci le met en tète des chiffrements à clef publique. Cependant, étant donné sa lenteur, due au calcul d'exponentiation modulaire, il est plus judicieux de l'implémenter en hardware pour compenser cet inconvénient. Ceci est l'objet des deux prochains chapitres.

CHAPITRE III

Algorithme d'implémentation

Chapitre III : Algorithme d'implémentation

III.1 Introduction

La performance des systèmes cryptographiques et en particulier RSA dépend essentiellement de l'implémentation d'algorithme d'exponentiation modulaire efficace. Comme les opérandes (le texte clair/crypté et clefs) sont souvent grands (sur 1024 bits ou plus), on cherche des algorithmes qui offrent notamment une architecture rapide, flexible et peu coûteuse en ressources matérielles.

Toutes les techniques existantes pour le calcul de l'opération $C = m^e \ mod \ n$ réduisent l'opération de l'exponentiation modulaire en une série d'opérations de multiplication modulaire.

Dans ce chapitre, on citera quelques algorithmes existants, et on développera l'algorithme le plus utilisé en pratique à savoir l'algorithme de Montgomery.

En général, les algorithmes classiques de multiplication modulaire procèdent sur deux étapes : la première consiste à générer le produit et la deuxième à calculer le modulo, la méthode la plus directe pour implémenter une multiplication est de se baser sur un additionneur-accumulateur itératif, cependant, cette solution se révèle lente puisque le résultat ne sera disponible qu'après n itérations, où n est la taille de l'opérande en binaire.

Il existe plusieurs algorithmes pour la multiplication modulaire, on cite la méthode de Bouth-Barrett, l'algorithme de Karatsuba, Brickell. Considéré comme le plus efficace et par conséquent le plus populaire [10], on adoptera pour notre travail l'algorithme de Montgomery.

III.2 Algorithme de Montgomery [3],[5],[6],[7],[8]

En 1985, P.L.Montgomery mit en œuvre un algorithme efficace pour le calcul de la multiplication modulaire de la forme $R = A.B \ mod \ M$ où A, B et M représentent respectivement le multiplicande, le multiplicateur et le modulo écrits en leur représentation binaire comme suit :

$$A = \sum_0^{n-1} a_i . 2^i \ , \quad \text{B}= \sum_0^{n-1} b_i . 2^i \quad \text{et} \quad \text{M}= \sum_0^{n-1} m_i . 2^i$$

Néanmoins, l'application de l'algorithme de Montgomery est soumise à deux conditions :

- Le modulo M doit être premier avec la base (dans notre cas la base est égale à 2, donc M doit être un nombre impair), condition qui est toujours vérifiée puisque dans le système RSA, le modulo M n'a que deux diviseurs P et Q qui sont premiers.

- Le multiplicande et le multiplicateur doivent être inférieurs au modulo M.

Une propriété principale de l'algorithme de Montgomery est que le calcul du multiple du modulo M qu'on va soustraire est basée sur le LSB de l'accumulation du produit partiel. Si R est le produit partiel courant, alors q est choisi de telle sorte que R+q.M soit un multiple de la base 2, après cela, R est décalé à droite par 2 positions, i.e. divisé par 2 pour l'utiliser à l'itération suivante. Après n itérations, le résultat obtenu est $R = A.B.2^{-n}\ mod\ M$.

Pour retrouver le juste résultat, on doit multiplier le résultat de cet algorithme par $2^n\ mod\ M$. Toute fois, étant donné que l'objectif de l'algorithme de multiplication modulaire de Montgomery est de calculer des exponentiations, il est préférable de pré-multiplier l'opérande par 2^{2n}et ensuite de multiplier le résultat par 1 pour éliminer le facteur 2^{-n}.
L'algorithme de Montgomery est représenté sur la figure **III.1**.

```
Algorithm Montgomery (A,B,M)
    Int R=0;
    1: for i=0 to n-1
    2:     R = R + a_i.B;
    3:     if r_0 = 0 then
    4:         R = R div 2;
    5:     else
    6:         R = (R + M) div 2;
    Return R;
end.
```

Figure III-1 : Algorithme de Montgomery

Une version modifiée de l'algorithme de Montgomery consiste en une simplification de l'algorithme, en effet, le LSB de $R + a_i.B$ est égal au LSB de la somme du LSB de R et du LSB de B si $a_i = 1$, et est égal au LSB de R autrement. En outre, la nouvelle valeur de R est l'ancienne valeur sommée soit avec $a_i.B$ où avec $a_i.B + q.M$, tout dépend la valeur de q.

L'algorithme de Montgomery modifié est représenté sur la figure **III.2**.

Algorithm ModifiedMontgomery (A,B,M)

Int R=0;

1: **for** i=0 to n-1

2: $q_i = (r_0 + a_i.b_0)\ mod\ 2$;

3: $R = (R + a_i.B + q_i.M)\ div\ 2$;

Return R;

end.

Figure III-2 : Algorithme de Montgomery modifié

Etant donné l'expression $R = R + a_i.B + q_i.M$ de la deuxième ligne de l'algorithme représenté sur la figure **III.2**, on peut calculer la valeur de R dépendamment des valeurs des bits a_i et q_i, comme c'est montré dans le tableau **III.1**.

a_i	q_i	$R + a_i.B + q_i.M$
1	1	$R + MB$
1	0	$R + B$
0	1	$R + M$
0	0	R

Tableau III-1 : Calcul de $R + a_i.B + q_i.M$

Où MB est le résultat de la somme $B + M$.

III.2.1 Architecture systolique du bloc de multiplication [3],[4],[5],[10],[13]

L'architecture systolique, introduite en 1978 par Kung et Leiserson, s'est révélé être un outil puissant pour la conception de processeurs intégrés spécialisés. L'architecture systolique est présentée comme étant un réseau de cellules élémentaires identiques et localement interconnectées (voir la figure **III.3**). Chaque cellule reçoit des données en provenance des cellules voisines, effectue un calcul simple, puis transmet les résultats, toujours aux cellules voisines, un temps de cycle plus tard. Seules les cellules situées à la frontière du réseau communiquent avec le monde extérieur.

La dénomination systolique provient d'une analogie entre la circulation des flots de données dans le réseau et celle du sang humain.

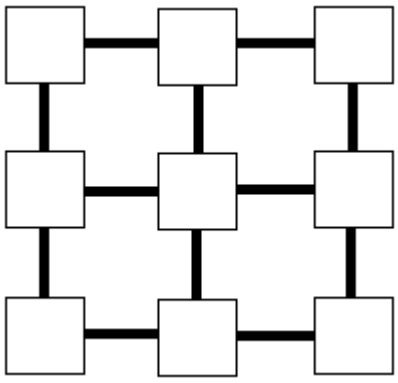

Figure III-3 : Réseau systolique

En s'appuyant sur les travaux de Montgomery, Mc Canny et Mc Whirter, Colin.D Walter mettent en place une architecture systolique de la multiplication modulaire de Montgomery (voir figure **III.4**) destinée à une implémentation hardware.

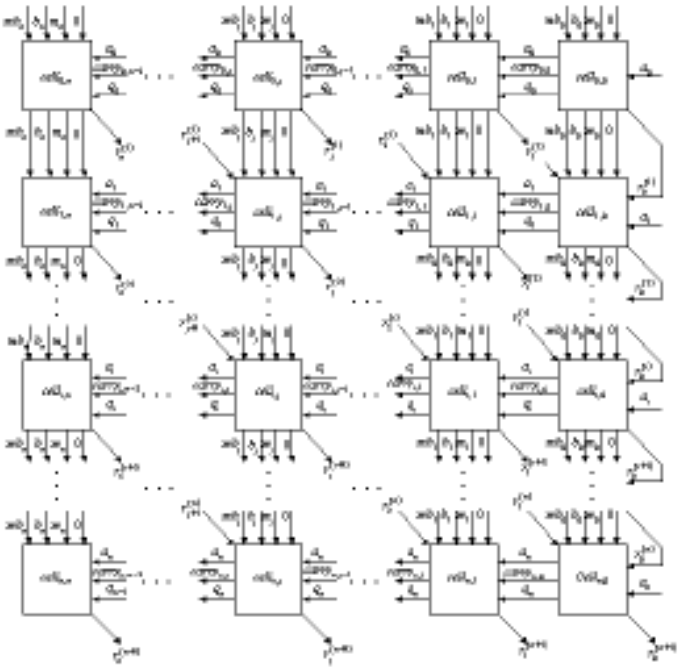

Figure III-4 : Architecture systolique du bloc de multiplication de Montgomery

Le principal élément calculateur (PE) calcule le bit r_j du résidu R. Le bloc de multiplication est constitué de PE's dupliqués en cascade (n+1 fois), chacun d'eux calcule l'élément r_j, $0 \leq j \leq n$, de R. la duplication de toute la chaine verticalement constitue le calcul itératif du bit r_j. L'architecture de base de ces éléments est illustrée par la figure **III.5**.

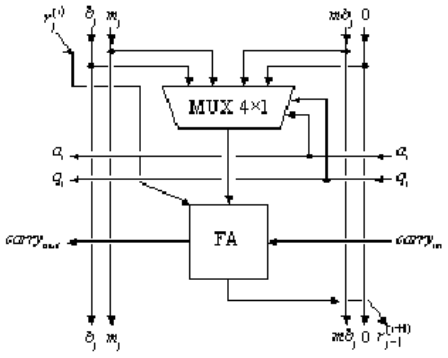

Figure III-5 : Architecture de base des PE

Les éléments PE situés à la périphérie présentent des architectures peu différentes de celles des éléments de l'intérieur du bloc, ceci est dû aux spécificités des entrées de ces éléments, en effet, l'élément situé en haut à droite, présente des particularités, du fait que $r_0^{(0)}$et $carry$ sont nuls, ceux situés au long de la colonne à droite, présentent la particularité que le carry soit nul et enfin, ceux d'en haut que $r_0^{(0)}$soit nul. Pour voir cette différence, les éléments d'en haut, ceux de la colonne la plus à droite et celui en haut à droite sont représentés respectivement de gauche à droite sur la figures **III.6**.

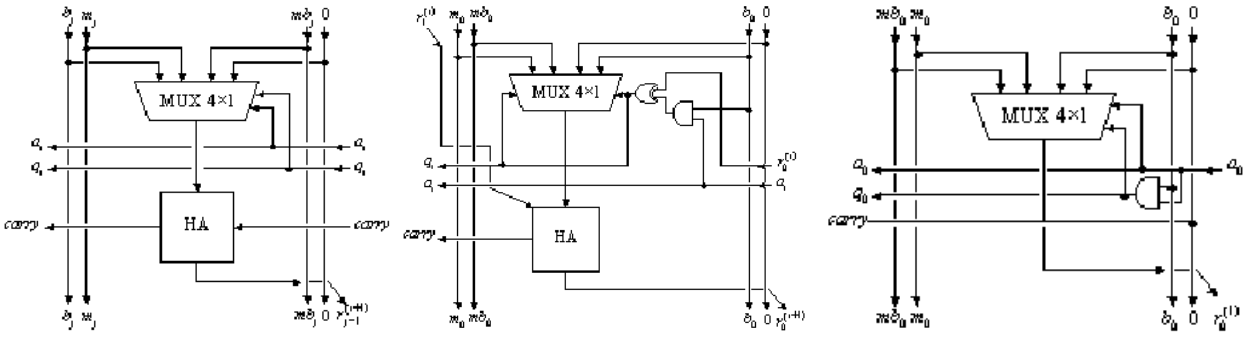

Figure III-6 : Architecture des éléments d'en haut, de droite et celui d'en haut à droite

III.3 Exponentiation modulaire séquentielle [5]

La version séquentielle de l'algorithme de Montgomery se base sur un calcul de multiplications successives, d'où la séquentialité.

La mise en œuvre hardware de cette architecture emploie deux blocs de multiplication modulaire à base de l'algorithme de Montgomery systolique, quatre registres pour chacune des variables T, M, RxR, RxT, et enfin un dernier registre à décalage pour le stockage de E, il

représente lui même le contrôleur, en se basant sur la longueur de E, il contrôle le nombre d'itération nécessaire pour le déroulement de l'algorithme.

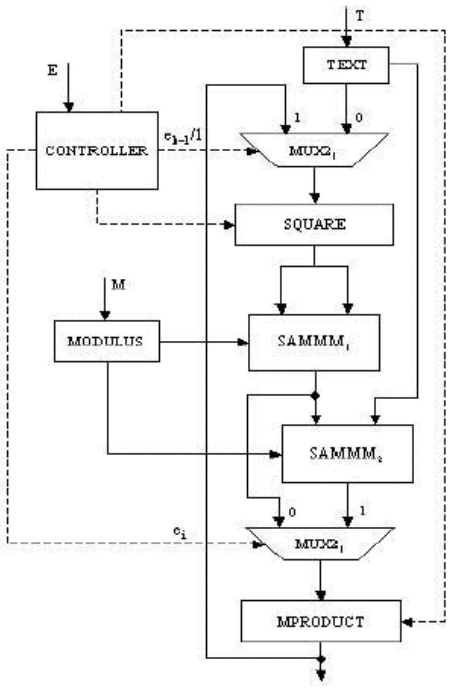

Figure III-7 : Architecture séquentielle du bloc d'exponentiation modulaire

III.4 Exponentiation modulaire parallèle [5]

Contrairement à l'architecture séquentielle, l'architecture parallèle de l'exponentiation modulaire se base sur un calcul de deux opérations de multiplication distinctes, d'où la possibilité d'exécuter ces deux opérations en parallèle, ceci revient à réduire le temps d'exécution de l'algorithme de moitié, cependant il requière une circuiterie deux fois plus importante que celle nécessaire pour l'implémentation de l'algorithme séquentiel.

L'implémentation hardware de cette architecture, comme la montre la figure **III-8**, nécessite l'utilisation de deux blocs de multiplication modulaire à base de l'algorithme de Montgomery systolique, 6 registres pour stocker chacune des variables T, M,$P^{(i)}$, $P^{(i+1)}$, $R^{(i)}$, $R^{(i+1)}$, et enfin un contrôleur (dernier registre à décalage) pour stocker E et contrôler le nombre d'itération nécessaire.

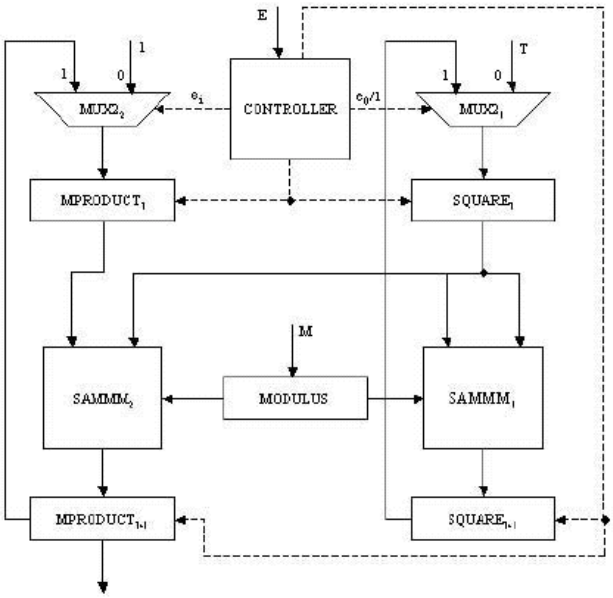

Figure III-8 : Architecture parallèle du bloc d'exponentiation modulaire

III.5 Exponentiation modulaire systolique [5]

L'architecture systolique du bloc d'exponentiation modulaire de gauche vers droite (illustrée dans la figure **III-9**) emploie m-1 éléments e-PE montés en cascade, chacun d'eux effectue les opérations d'une itération de l'architecture parallèle, pour ce faire, il utilise, en outre que le multiplexeur, deux blocs de multiplication suivant l'architecture systolique.

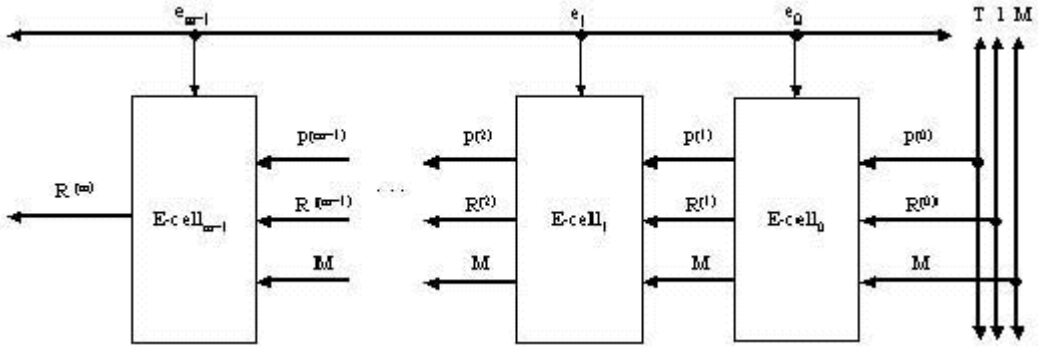

Figure III-9 : Architecture systolique du bloc d'exponentiation modulaire

L'architecture de chaque bloc e-PE est illustrée sur la figure **III.10**.

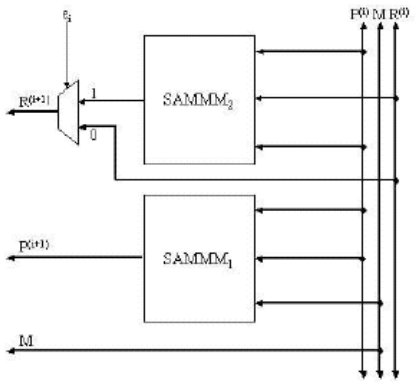

Figure III-10 : Architecture d'un e-PE

III.6 Architecture finale du bloc d'exponentiation modulaire [5],[9]

Comme on l'a déjà évoqué auparavant, l'algorithme de Montgomery décrit auparavant, calcule le produit : $A.B.2^{-n}\ mod\ M$, pour un résultat juste de l'exponentiation modulaire, il est impératif de multiplier, avant, les opérandes par $2^{2n} mod\ M$, et puis après, multiplier le résultat de l'exponentiation modulaire par 1, ceci éliminera la constante 2^{-n}.

La figure **III-11** montre le schéma final du bloc d'exponentiation modulaire, le bloc le plus à droite est celui qui effectue la multiplication par $2^{2n} mod\ M$, le plus à gauche effectue quant à lui la multiplication par 1. Le bloc d'exponentiation (du milieu) pourrait être l'un des trois architectures de l'exponentiation modulaire à savoir : séquentielle, parallèle ou systolique.

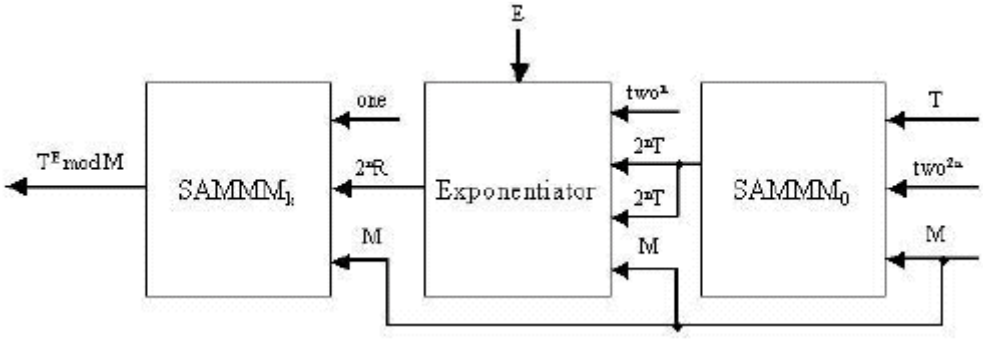

Figure III-11 : Architecture finale du bloc d'exponentiation modulaire

III.7 Conclusion

Le test des trois architectures du bloc d'exponentiation, réalisé sur une carte FPGA (VIRTEX-E) [12], a permis d'estimer la surface hardware requise, le temps d'exécution et enfin le produit surface×temps. La figure **III-12** montre la comparaison entre les trois architectures du produit surface×temps.

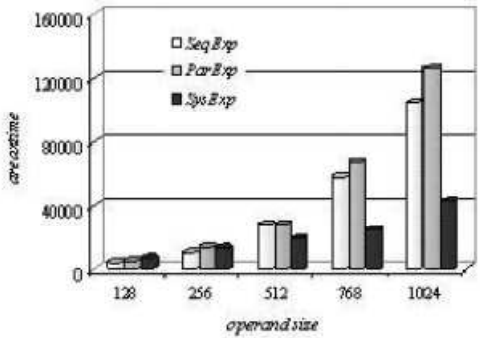

Figure III-12 : Comparaison du produit surface×temps chez les trois architectures

Se basant sur cette comparaison, le choix de l'algorithme d'implémentation s'est porté sur l'architecture systolique puisqu'elle offre le meilleur compromis entre le temps de propagation et la surface hardware requise, surtout pour une application comme la notre, où on essaye de réduire au minimum le temps de cryptage/décryptage des données.

CHAPITRE IV

Mise en œuvre

Chapitre IV : Mise en œuvre

IV.1 Introduction

De nos jours, les systèmes électroniques sont de plus en plus complexes, par conséquent, la nécessité d'une plus grande intégration est impérative. Dans notre application, on a adopté la solution SoC, cette dernière permet d'intégrer tout un système sur une seule puce.

Les circuits programmables se révèlent convenables pour de telles applications grâce à leur grande capacité d'intégration et facilité de programmation. Les circuits FPGA viennent en tête des circuits programmables du fait qu'ils offrent des coûts de réalisation plus faibles et une grande flexibilité devant les technologies concurrentes : les microcontrôleurs et les ASIC.

Notre travail consiste en la conception d'un IP-core en utilisant l'environnement ISE de Xilinx, dédié au calcul de la multiplication modulaire selon l'algorithme de Montgomery. Cet IP-core sera par la suite intégré dans un SoC en utilisant l'environnement de développement EDK fournit par Xilinx. Dans ce chapitre, on va détailler nos outils de développement et la démarche suivie pour la réalisation de notre application.

IV.2 Les systèmes sur puce

IV.2.1. Définition

Un SoC est un circuit intégré complexe qui intègre la majorité des fonctionnalités d'un produit final, le tout, sur une seule puce. En général un SoC incorpore un (ou plusieurs) processeur, de la mémoire sur puce, des fonctions d'accélération, mais aussi des interfaces avec des composants périphériques. La conception du SoC utilise l'approche du Co-design, il permet de combiner, à la fois, des composants logiciels (SW) et matériels (HW). Parce que la conception SoC peut être interfacée avec le monde réel, le SoC peut souvent incorporer des composants analogiques, et peut-être dans le futur, des systèmes opto/microélectroniques mécaniques.

L'élaboration d'un système SoC peut passer par l'utilisation d'un circuit FPGA. Cependant, l'utilisation d'un circuit ASIC pour la conception d'un SoC présente un inconvénient dû au manque de flexibilité. En effet, les circuits ASIC réputés pour être des circuits figés, ne permettent pas de faire des modifications sur l'algorithme implémenté ou même sur ses

paramètres. Les FPGA, quant à eux, certes, ne sont pas aussi rapides que les ASIC, mais offrent une très grande flexibilité qui les privilégie pour des applications d'implémentation d'algorithmes. C'est le cas que nous aurons à traiter.

IV.2.2 Configuration des FPGA

Contrairement aux ordinateurs par exemple, les FPGA ne possèdent pas un programme résident, il faudrait les configurer à chaque mise sous tension, cette configuration peut se faire par la méthode Master/Slave (le FPGA est connecté à un ordinateur), où depuis une ROM que l'on pourrait lui connecter. Cette configuration se ramène au chargement d'un fichier appelé le bitstream, ce dernier décrit les interconnexions entres les constituants du FPGA (CLB et IOB). Ce fichier peut être créé par deux moyens :

- L'utilisation d'un langage de description matérielle (VHDL), ce langage permet de synthétiser comment un matériel doit être implémenté, Dans ce cas, on est amené à utiliser l'outil de développement ISE de Xilinx.
- L'utilisation d'un processeur soft, qui offre plus de flexibilité et de rapidité de reconfiguration que les processeurs hard. Ce qui nous a conduit à l'utilisation de l'outil EDK de Xilinx.

IV.2.3 L'environnement ISE

Xilinx a mis au point un environnement flexible de conception haut niveau. Il permet le développement d'une application Soc sur circuit programmable de sa définition jusqu'à sa synthèse finale.

La version utilisée dans notre travail est la version ISE 9.2i parue en juin 2007. Pour la conception d'un projet, Xilinx ISE met à la disposition de l'utilisateur pour la conception d'un projet, quatre outils essentiels :

- Les outils d'édition (texte (HDL) ou graphique (schématique, machine d'état));
- Les outils de simulation
- Les outils de synthèse
- Les outils de mise au point

IV.2.4 L'environnement EDK

EDK est un environnement de développement fourni par Xilinx destiné au développement d'une application complète embarquée et à son intégration sur un FPGA. EDK donne accès à tous les dimensionnements nécessaires à l'application que l'on souhaite créer.

L'environnement EDK fait appel au concept du Co-design. Il permet le développement hardware en parallèle à un développement software

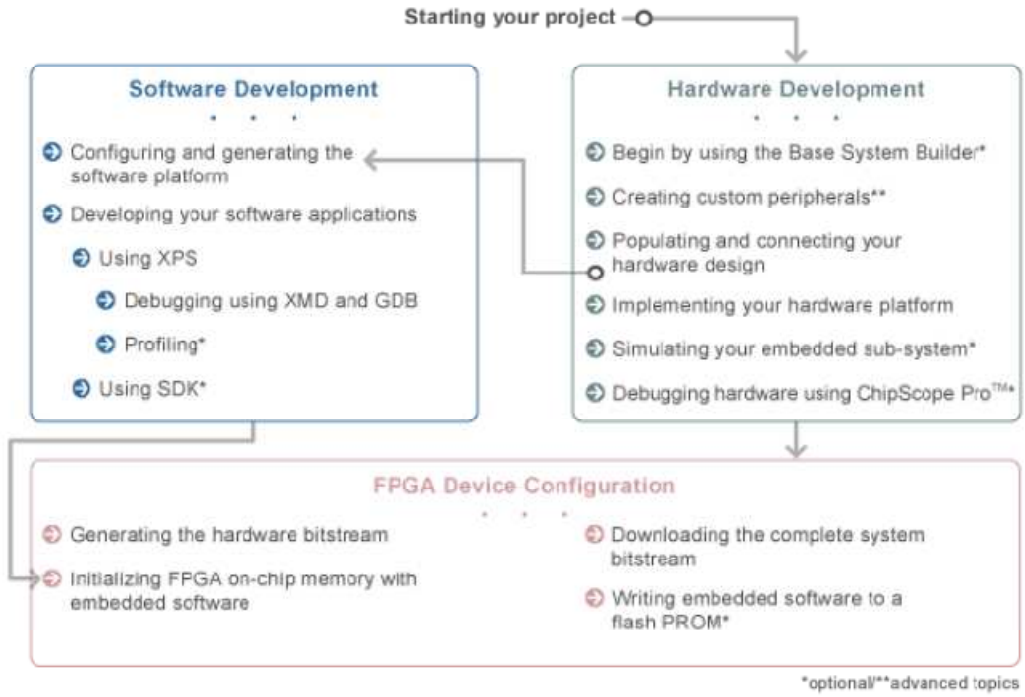

Figure IV-1 : Approche de développement sur EDK

IV.3 Architecture du SoC

Notre réalisation se fait autour d'un processeur virtuel appelé Microblaze, ce dernier est entouré de tout un environnement constitué de BRAM (bloc de RAM), de bus, de périphériques d'entrées/sorties et enfin d'un IP-Core que l'on détaillera plus loin. L'architecture de ce système est illustrée par la figure **IV.2.**

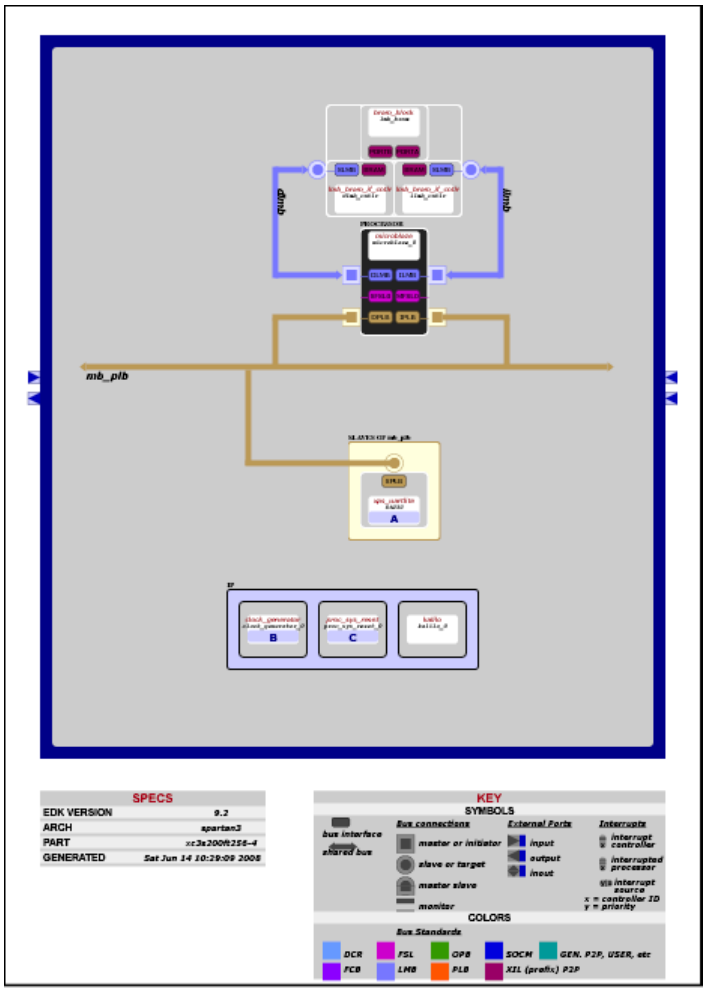

Figure IV-2 : Architecture du SoC

Le Microblaze est un soft processeur de 32 bits ayant une architecture Harvard à jeu d'instruction réduit (RISC), c'est-à-dire qu'il inclut des bus d'instructions et de données séparés, de 32 bits chacun fonctionnant à la même vitesse.

Le bus OPB permet de lier plusieurs maîtres à plusieurs esclaves. Il autorise un maximum de 16 maîtres et un nombre d'esclaves illimité selon les ressources disponibles. Xilinx conseille néanmoins un maximum de 16 esclaves. Comme ce bus est multi maîtres, il a donc une politique d'arbitrage paramétrable. Ce bus permet donc d'ajouter des périphériques au MicroBlaze dont les besoins en communications seront faibles.

Le bus LMB est un bus synchrone utilisé principalement pour accéder aux blocks RAM inclus sur le FPGA. Il utilise un minimum de signaux de contrôle et protocole simple pour s'assurer d'accéder à la mémoire rapidement (un front d'horloge).

FSL est un bus de communication unidirectionnel utilisé pour assurer des communications rapides entre deux éléments (IP-Cores) implémentés sur FPGA. Le MicroBlaze comporte 8 interfaces FSL de type Master/Slave (entrées/sorties). Chaque interface FSL est unidirectionnelle (simplex) et met en œuvre une FIFO (pour stocker les données) et des signaux de contrôle (FULL, EMPTY, WRITE, READ,…). Il met aussi à la disposition de l'utilisateur plusieurs fonctions intéressantes, les plus utilisées sont : "microblaze_bwrite_datafsl" et "microblaze_bread_datafsl". Ces deux fonctions permettent d'échanger des données entre différents MicroBlazes, par exemple, en utilisant la FIFO déjà intégrée dans le bus FSL. Les deux fonctions bwrite et bread sont bloquantes, bwrite se bloque lorsque la FIFO du bus FSL est saturée et bread se bloque lorsque la FIFO est vide. Il doit y avoir un bwrite pour débloquer la lecture. Les communications sur les liens FSL se font simplement grâce à des instructions prédéfinies. Elles peuvent atteindre les 300 Mo/s pour une fréquence d'horloge de 150 Mhz.

IV.4 Réalisation de l'IP-Core

L'IP-Core est un bloc logique utilisé dans la conception d'application sur FPGA. Il a pour tâche d'accélérer l'exécution d'une opération qui consomme beaucoup de temps en software. Dans notre travail, le SoC qu'on va réaliser, aura pour tâche de calculer et accélérer l'opération de la multiplication modulaire.

L'IP-Core qu'on va réaliser comporte trois principaux blocs :

- Le bloc de calcul de la multiplication modulaire réalisé à partir de l'architecture systolique ;
- Le bloc de commande ;
- Un bloc de communication.

La réalisation de l'IP-Core s'est faite en utilisant l'environnement de développement ISE de Xilinx, La figure ci-suit montre l'architecture de notre IP-Core.

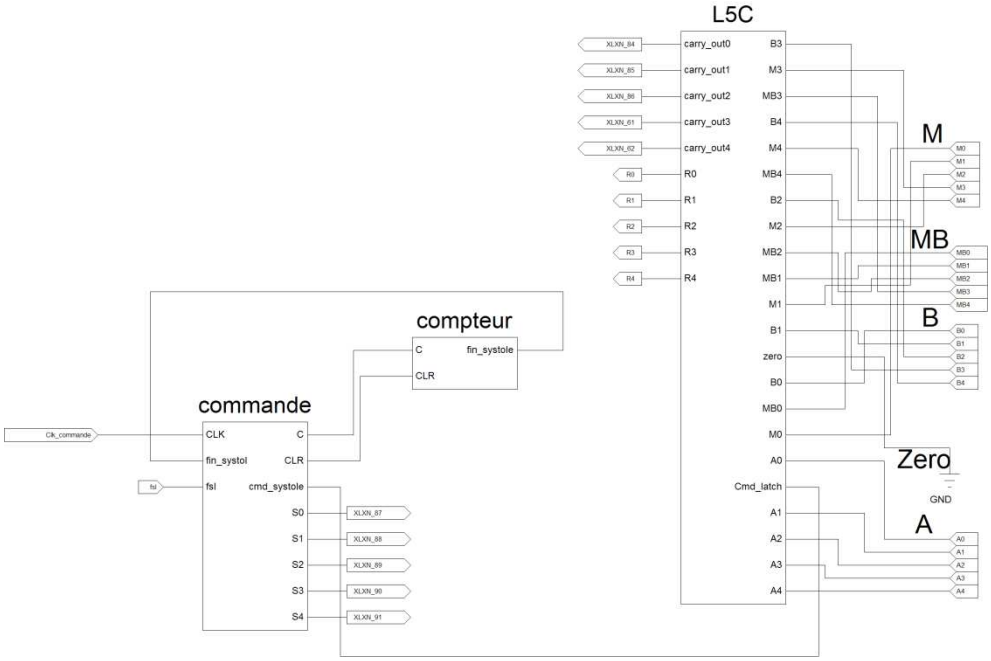

Figure IV-3 : Architecture de l'IP-Core

IV.4.1 Bloc de calcul

Le bloc de calcul est une disposition de cellules selon l'architecture systolique, pour notre application, on a commencé par réaliser une matrice 5x5 (voir figure **IV.4**), son extensibilité à NxN reste à étudier. Cette matrice est formée de 5 blocs interconnectés entre eux, chacun de ces blocs est une chaine de cellules élémentaires, montées en cascade, qui forme une ligne, ces cellules sont semblables sauf ceux de la colonne la plus à droite.

Le bloc de calcul a été réalisé en utilisant l'éditeur graphique fournit par l'environnement ISE.

Chaque cellule du bloc de calcul possède une entrée de commande, elle sert à commander les signaux de sorties, et par conséquent, synchroniser toutes les cellules de manière à ce qu'elles fonctionnent toutes ensemble.

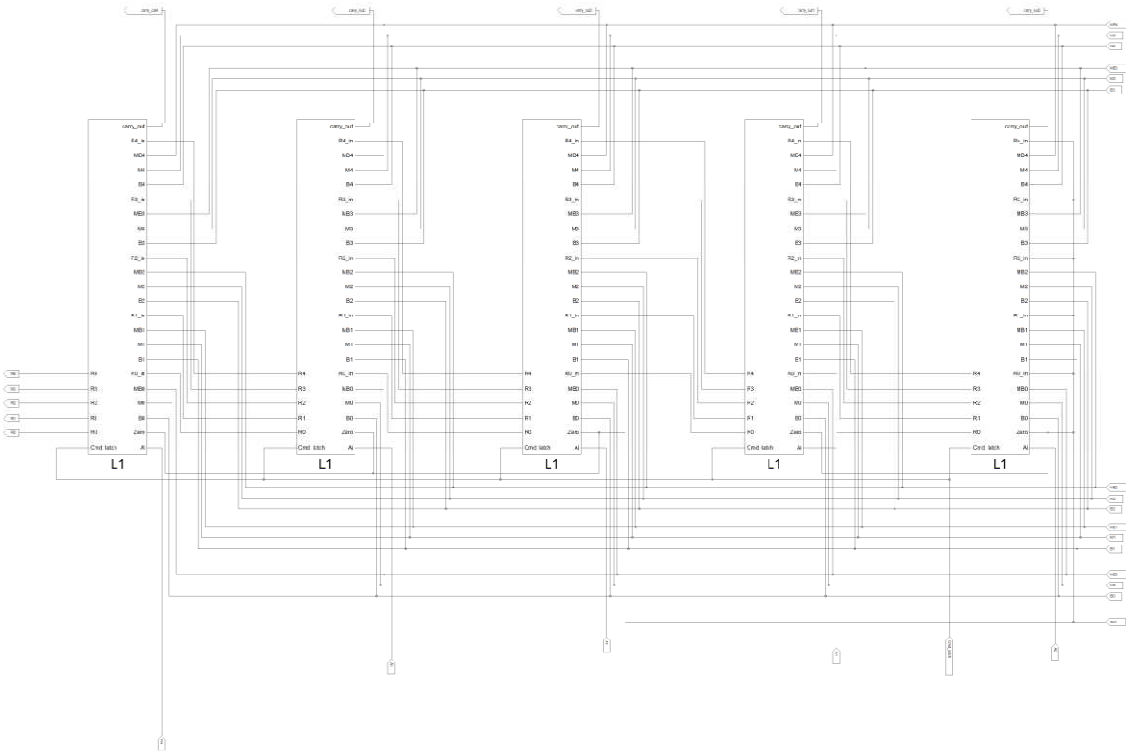

Figure IV-4 : Cellule systolique 5x5

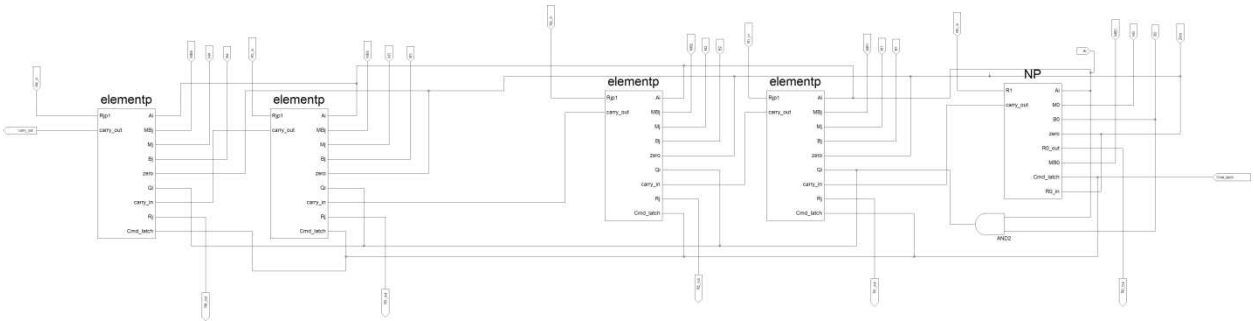

Figure IV-5 : Une ligne de l'architecture systolique

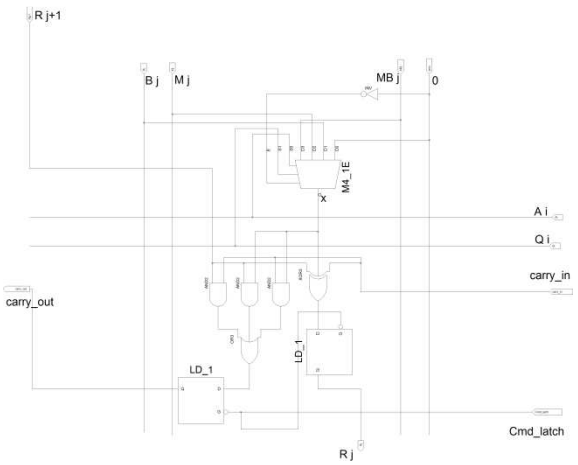

Figure IV-6 : Cellule élémentaire centrale

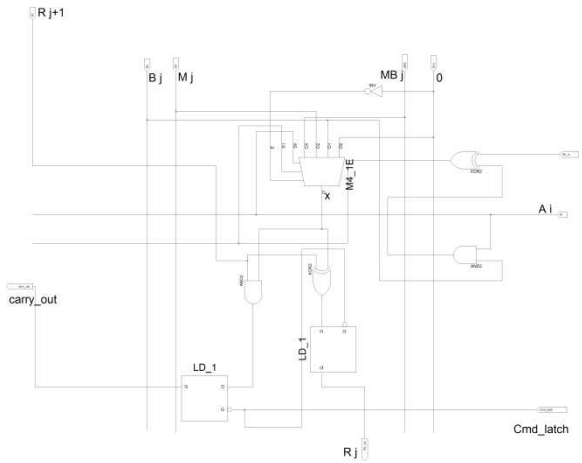

Figure IV-7 : Cellule périphérique

IV.4.2 Bloc de commande

Ce bloc a pour but de générer les signaux de commande pour le chemin de données présenté ci avant. Pour ce faire, on a réalisé la machine d'état (voir figure **IV.8**) en utilisant l'éditeur de machines d'état (State CAD) inclut dans l'environnement ISE de Xilinx.

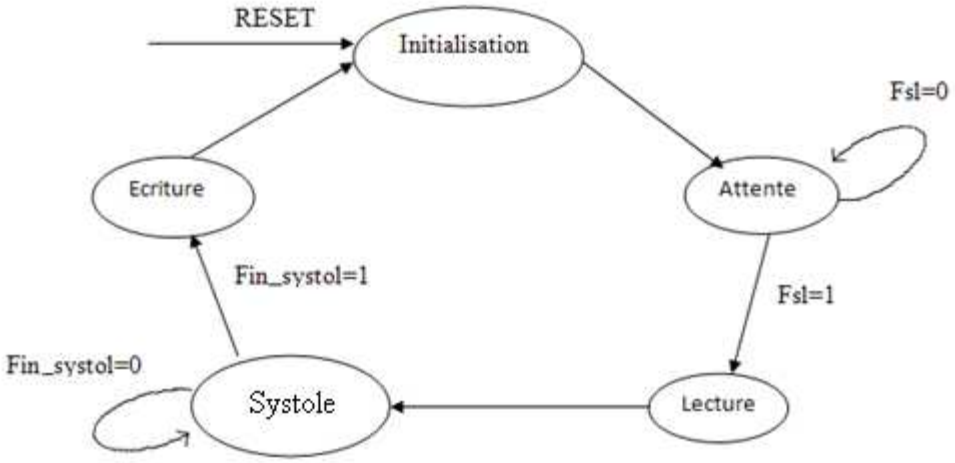

Figure IV-8 : Machine d'état du bloc de commande

Le premier état est atteint via le signal RESET. On y remet tout les signaux à l'état bas initialisant ainsi tous les registres. L'état d'attente correspond à la scrutation de la présence de données d'entrée. L'état de lecture réalise le transfert des données dans les registres dédiés à cet effet. Le calcul est réalisé à la phase suivante. Le fonctionnement systolique est commandé par un compteur de taille corrélée avec celle de la matrice cible. L'écriture des données constitue l'étape finale avant la réinitialisation de toutes les variables.

L'architecture systolique de l'algorithme de Montgomery calcule le résultat final après 3n+2 cycles du moment d'entrée de la première donnée, par conséquent, pour notre matrice 5x5, 17 cycles d'horloges sont nécessaires pendant la phase calcul.

Ce compteur (voir figure **IV.9**) réalisé avec l'éditeur schématique d'ISE, décompte depuis 17 jusqu'à 0, le signal d'entrée est le signal C généré par la machine d'état. Son signal de sortie est le signal fin_systol qui va enclencher l'écriture sur le registre de sortie.

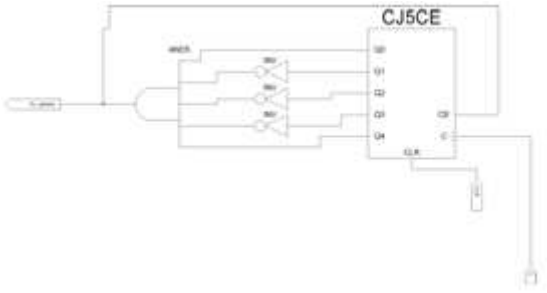

Figure IV-9 : Bloc compteur

IV.4.3 Bloc de communication

Ce bloc aura pour tâche de réaliser la communication entre le bloc de commande et le bus de données FSL lorsqu'une donnée est disponible. Le bloc de commande est enclenché à travers la mise au niveau haut du signal fsl. A son tour le bloc de calcul lit les données depuis les registres d'entrées. A la fin du calcul, le bloc de communication envoie le résultat du bloc de calcul à travers le bus FSL.

IV.5 Résultats de la synthèse

Le résultat de la synthèse du nœud 5x5 (figure **IV.4**) montre la consommation de notre réalisation en termes de ressources physiques du FPGA, en indiquant le nombre de CLB, d'IOB et de bus d'interconnexion ainsi que le pourcentage des ressources consommées (figure IV.10).

On note que la consommation du nœud réalisé est très faible, elle représente 3% de la capacité totale des cellules CLB, 1% du réseau d'interconnexion et 18% du nombre total d'IOB.

Device Utilization Summary				
Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of Slice Latches	40	3,840	1%	
Number of 4 input LUTs	110	3,840	2%	
Logic Distribution				
Number of occupied Slices	59	1,920	3%	
Number of Slices containing only related logic	59	59	100%	
Number of Slices containing unrelated logic	0	59	0%	
Total Number of 4 input LUTs	110	3,840	2%	
Number of bonded IOBs	32	173	18%	
IOB Latches	10			
Number of GCLKs	1	8	12%	
Total equivalent gate count for design	988			
Additional JTAG gate count for IOBs	1,536			

Figure IV.10 : Consommation des ressources physiques du nœud 5x5

L'ajout du bloc de commande induira une faible augmentation du nombre de ressources consommées, cette augmentation sera faible vue la faible complexité du bloc de commande (voir figure **IV.11**).

Device Utilization Summary				
Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Total Number Slice Registers	48	3,840	1%	
Number used as Flip Flops	12			
Number used as Latches	36			
Number of 4 input LUTs	113	3,840	2%	
Logic Distribution				
Number of occupied Slices	65	1,920	3%	
Number of Slices containing only related logic	65	65	100%	
Number of Slices containing unrelated logic	0	65	0%	
Total Number of 4 input LUTs	113	3,840	2%	
Number of bonded IOBs	33	173	19%	
IOB Latches	9			
Number of GCLKs	2	8	25%	
Total equivalent gate count for design	1,080			
Additional JTAG gate count for IOBs	1,584			

Figure IV-11 : Consommation des ressources physiques de l'IP-Core

Une estimation de la consommation des ressources physiques d'un nœud de taille 1024x1024 n'est pas évidente. En effet la relation n'étant pas linéaire, on ne peut estimer une telle extensibilité. Il est à noter qu'on peut estimer une certaine extensibilité en reliant quatre ou cinq nœuds de 5x5 et par la suite, effectuer le même test pour voir ses consommations de ressources.

La figure **IV.12** montre la simulation du bloc de commande (en bas de la figure à gauche).

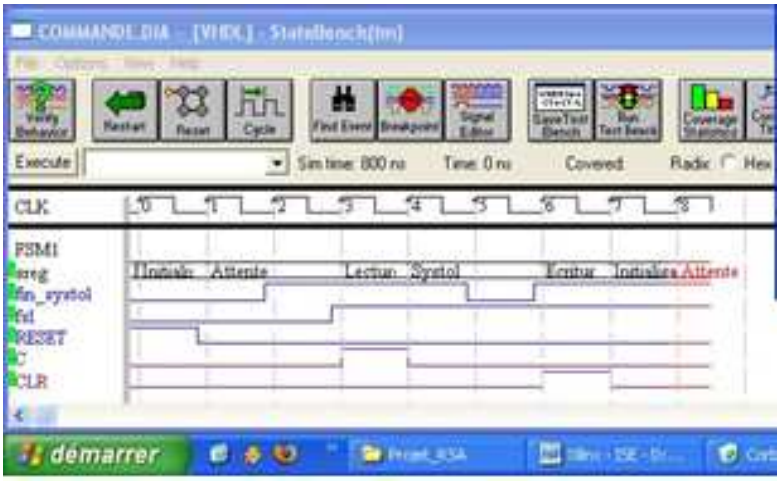

Figure IV-12 : Simulation du bloc de commande

IV.6 Simulation de l'IP-Core

Pour simuler l'IP-Core réalisé, on applique un exemple de : A=14, B=11, M=23, N=6. Sachant que l'algorithme de Montgomery calcule : $R = A.B.2^{-N}\ mod\ M$, le résultat théorique devrait être R=6.

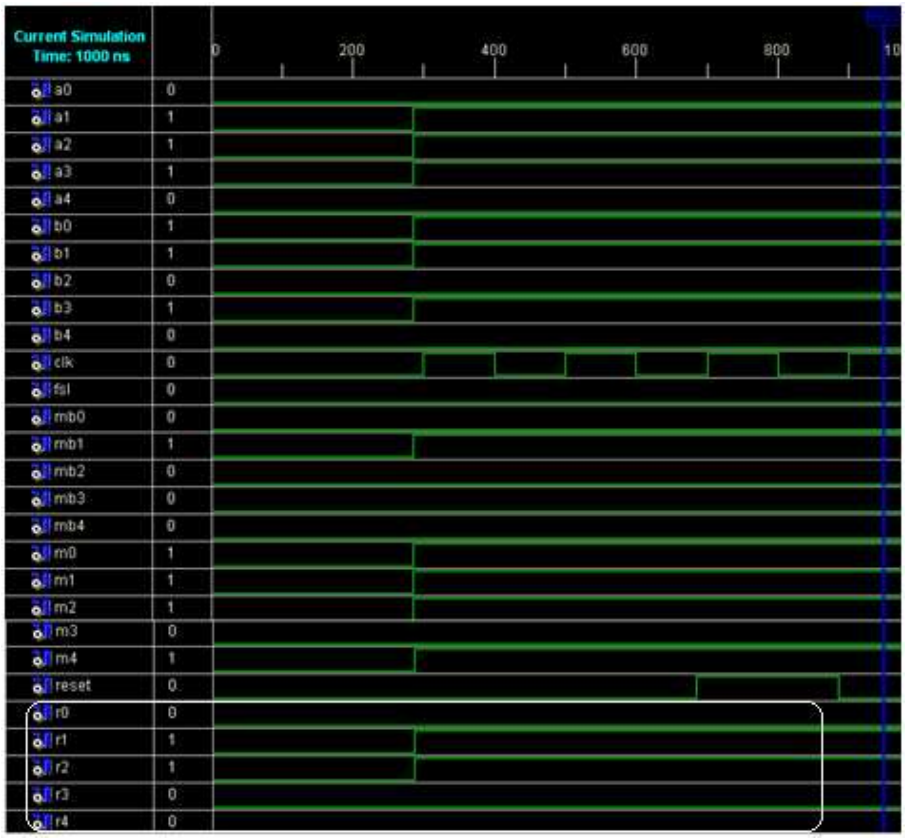

Figure IV-13 : Simulation de l'IP-Core

On voit, au bas de la figure **IV-13**, que le résultat est bien R=6 (00110).

IV.7 Conclusion

La réalisation de l'IP-Core étant faite, le résultat de la synthèse montre la faible consommation en ressources physiques du FPGA, d'où le choix de l'architecture systolique de l'algorithme de Montgomery. On peut étendre notre estimation de ressources consommées pour toute l'architecture y compris le Microblaze, les périphériques d'entrées/sorties et les bus de connexion (FSL, OPB, LMB). Mais aussi pour une estimation du temps d'exécution, ce dernier étant le but de notre implémentation en hardware.

Conclusion générale

Notre projet a pour objet de l'implémentation de l'algorithme de cryptographie RSA sur FPGA et son adaptation à une solution soc.

Comme nous l'avons montré dans nos précédents chapitres, l'algorithme RSA présente l'avantage d'une cryptanalyse plus difficile. Cependant, les temps de calcul restent longs si on envisage son utilisation dans un système nécessitant des traitements sur des flux de données. Pour y remédier, deux orientations sont possibles. Elles peuvent être complémentaires. La première orientation est la recherche d'algorithme introduisant des simplifications au niveau de calcul tenant compte de la spécificité des opérateurs à mettre en œuvre. C'est ainsi que dans le cas de RSA, l'algorithme de Montgomery apporte un gain appréciable dans le calcul de l'exponentiation modulaire. Ce qui a retenu notre choix.

L'implémentation de l'algorithme peut prendre diverses formes. Nous citons les implémentations sérielle, parallèle et systolique. Chacune présente des avantages et des inconvénients tenant des critères de surface et de rapidité de calcul. Nous avons opté pour l'implémentation systolique car elle apporte un meilleur compromis entre la surface et temps de traitement.

Tenant compte du temps qui a été imparti à notre projet, nous avons implémenté et testé, tenant compte de l'algorithme de Montgomery et de sa forme systolique, un réseau de 5x5. Elle a été pensée naturellement sous forme d'un chemin de données et d'un module de commande. Cette dernière a été pensée et implémentée sous forme d'une machine d'état. L'évaluation de la performance reste dans notre cas non objective vu la taille de notre réseau. Un module de communication a été envisagé pour rendre notre implémentation intégrable dans un système sur puce. Ce dernier module reste spécifique à l'environnement EDK.

Il est évident que ce qui reste à faire à notre travail ce situe s'articule sur trois volets :

- Trouver une méthodologie pour réaliser une matrice de taille quelconque
- L'intégration effective dans un environnement soc fonctionnel
- Mise en œuvre dans le cas d'une application réelle.

Bibliographie

[1] Douglas Stinson, Cryptographie Théorie et Pratique, International Thomson Publishing France, PARIS, 1996.

[2] H.X. Mel, Doris Baker, La Cryptographie décryptée, CAMPUSPRESS, Juillet 2001.

[3] P. Quinton, Y. Robert, Algorithmes et architectures systoliques, MASSON, PARIS 1989.

[4] C.D. Walter, "Systolic Modular Multiplication," IEEE Transaction On Computers, 42(3), pages 376-378, 1993.

[4] N. Nedjah, L.M. Mourelle, "Three Hardware Implementations for the Binary Modular Exponentiation: Sequential, Parallel and Systolic," Proc. 15th International Symposium on Computer Architecture and High Performance Computing, Brazil, IEEE computer society Press, 2003.

[5] N. Nedjah, L.M. Mourelle, "Efficient hardware implementation of modular multiplication and exponentiation for public-key cryptography," Proc. 15th International Conference on High Performance Computing for Computational Science, IEEE computer society Press, Porto, Portugal, pages 451-463, 2002.

[6] N. Nedjah, L.M. Mourelle, "A Review of Modular Multiplication Methods and Respective Hardware Implementations," Informatica 30, pages 111–129, 2006.

[7] N. Nedjah, L.M. Mourelle, "Fast hardware for modular exponentiation with efficient exponent pre-processing," Journal of Systems Architecture 53, pages 99–108, 2007.

[8] A.Z. Alkar, R. Sonmez, "A hardware version of the RSA using the Montgomery's algorithm with systolic arrays," The VLSI journal 38, pages 299–307, 2004.

[9] A. Cilardo, A. Mazzeo, L. Romano, G.P. SaggeseExploring, “Exploring the design-space for FPGA-based implementation of RSA,” Microprocessors and Microsystems 28, pages 183–191, 2004.

[10] Ç.K. Koç, “RSA hardware implementation,” Technical Report TR801, RSA Laboratories, 1995.

[11] F. Bounkar, S. Djellouli, “système temps réel embarqué pour commander un robot mobile,” Projet de fin d’étude à l’ENP, Juin 2005.

[12] Chinuk Kim, “VHDL Implementation of Systolic Modular Multiplication on RSA Cryptosystem,” Master of Science (Computer Science) at The City College of the City University of New York, Jan. 2001.

[13] George Joseph, “Design and Implementation of High-speed Algorithms for Public-key Cryptosystem,” Master of Engineering in Faculty of Engineering Built Environment & Information Technology, University of PRETORIA, March 2005.

[14] “EDK Concepts, Tools, and Techniques”, Embedded Development Kit 9.2i, Mai 2007.

[15] “Fast Simplex Link (FSL) Bus (v2.11a)”, Product Specification, Juin 2007.

[16] Hans-Peter Rosinger, “Connecting Customized IP to the MicroBlaze Soft Processor Using the Fast Simplex Link (FSL) Channel”, XAPP529, Mai 2004.

Webographie

http://www.bibmath.net/crypto
http://www.multimania.com/marief
http://www-rocq.inria.frwhoAnne.Canteautcrypto_moderne.pdf
http://www.crypto.freezee.org/crypto/cours.html
http://www.rsa.com/