الجمهورية الجزائرية الديمقراطية الشعبية

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR
ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE YAHIA FARES DE MEDEA

FACULTE DE TECHNOLOGIE

Département de Génie Électrique



Mémoire de Fin d'Études

Présenté pour l'obtention du Diplôme
de Master en Télécommunications

Option : Systèmes Intelligents et Communicants

THEME

Etude et implémentation sur un circuit FPGA de l'algorithme de cryptage ECC

Proposé et dirigé par :

Dr. OULD ZMIRLI Mohamed

Préparé et présenté par :

Melle. MADANI Wassila

Melle. HAMDANE Meroua

AnnéeUniversitaire2016-2017

ملخص :

الهدف من هذا المشروع هو تنفيذ نظام التشفير ECC يستند على المعالج Microblaze ، محمل على دارة منطقية البرمجة (FPGA) ل Xilinix . الجداء السلمي هي العملية الرئيسية للتشفير و فك التشفير في ECC. هذه الأخيرة ترتكز على سلسلة من جمع ومضاعفة النقاط على المنحنى الاهليجي.

الكلمات المفتاحية :المنحنى الاهليجي، ECC , الجداء السلمي . نظام مضمن ، FPGA,

Résumé :

Ce projet consiste à l'implémentation d'un crypto système ECC (Elliptic Curve Cryptography) embarqué sur un circuit FPGA de Xilinx. La multiplication scalaire est l'opération principale du chiffrement et du déchiffrement basés sur les courbes elliptiques. Cette opération est établie par une série d'additionnement et doublement des points sur la courbe elliptique.

Mots clés : courbes elliptiques, ECC, FPGA, multiplication scalaire, système embarqué.

Abstract :

This project involves the implementation of an ECC (Elliptic Curve Cryptography) system embedded on a Xilinx FPGA circuit. Scalar multiplication is the main operation of encryption and decryption based on elliptic curves. This operation is established by a series of addition and doubling of points on the elliptic curve.

Keywords : Elliptic curves, ECC, FPGA, scalar multiplication, embedded system.

.

Remerciement

C’est avec l’aide de dieu qu’a un le jour ce présent travail.

Nous exprimons tous nos remerciements à notre Directeur de mémoire

Dr. OULD ZMIRLI Mohamed de nos encadré, orienté, aidé et conseillé.

Nous adressons nos sincères remerciements à tous les professeurs, intervenants et toutes les personnes qui par leurs paroles, leurs écrits, leurs conseils et leurs critiques ont guidé notre réflexions et ont accepté à nos rencontrer et répondre à notre questions durant notre recherches.

Nous remercions tout particulièrement les membres de jury qui ont accepté de juger notre travail ainsi que tous les enseignants qui ont contribué à notre formation.

Nous tenons enfin également à remercie sincèrement tous ceux qui ont participé de prés et de loin dans la réalisation de ce travail.

Dédicace

Je dédie ce travail à la lumière
de ma vie : Mes chers parents, que nulle dédicace ne peut
exprimer mes sincères sentiments, pour leur patience
Illimitée, leur encouragement contenu, leur aide, en
Témoignage de mon profond amour et respect Pour ses
grands sacrifices.
A mes frères que Dieu vous paye pour tous vos bienfaits :
RABEH, pour ton énorme soutien et ta collaboration.
AMINE, avec tous mes sentiments de respect,
d'amour et de gratitude.
Et à toute ma grande famille sans exception.
A mes chères amies et A mon binôme Meroua.
A toutes personnes Ayant contribué de loin ou de près
À accomplir ce travail.

Wassila

Dédicace

Je dédie ce travail à

Mes chers parents, mon père et ma mère

Pour leur patience, leur amour, leur soutien

Et leurs encouragements.

A mes soeurs, mes frères, mes grands-mères

et tous les

membres de la famille HAMDANE.

A mon binôme Wassila

A tous mes chères amis et collègues de la classe SIC

2016/2017.

A tous ceux qui me connaissent et qui m'aiment surtout.

Meroua

Tables des matières

Introduction générale……...…… 1

Chapitre 1 : Cryptographie sur les courbes elliptiques (ECC)

1.1 Introduction……...……...…….…. 3

1.2 La cryptographie……...……. 3

1.2.1 Cryptographie symétrique(chiffrement à clé privée)…….…….. 4

1.2.2 Cryptographie asymétrique(chiffrement à clé public)……... 4

1.3 Généralités…….…. 5

1.3.1 Groupe……..… 5

1.3.2 Groupe abélien …… 5

1.3.3 Groupe cyclique ……..… 5

1.3.4 Anneau unitaire et anneau commutatif ……...……. 6

1.3.5 Corps fini ……...…..…. 6

1.3.5.1 Corps premier …….…….…..... 6

1.3.5.2 Corps binaire……..…….…. 7

1.4 Présentation de la courbe elliptique……...… 7

1.4.1 Définition de courbe elliptique…….…. 7

1.4.2 Loi de composition sur les courbes elliptiques…….…. 7

1.4.3 Multiplication de point……..…. 9

1.5 Protocoles cryptographiques basés sur ECC……. 10

1.5.1 Protocole d'échange de clés de Diffie-Hellman……..….. 10

1.5.2 Protocole de chiffrement et déchiffrement ……... 11

1.6 Comparaison avec le crypto-système de RSA ……..... 12

1.7 Conclusion…….……..... 13

Chapitre 2 : Conception des systèmes numériques embarqués

2.1 Introduction……..... 14

2.2 Les circuits logiques programmables……...……. 14

2.3 Les systèmes numériques embarqués ……... 16

2.3.1 Codesign : Conception conjointe logicielle/matérielle……...…….. 16
2.3.2 Flot de la Conception conjointe logicielle/matérielle……..…...…….. 17
2.3.3 Les composants virtuels (IPs) ……..……...…….. 18
2.4 Les systèmes sur puce (SoC)…….. 19
2.4.1 Les contraintes liées aux systèmes embarqués……...…….. 19
2.4.2 Les SoC et les SoPC…….... 20
2.5 Communications sur puce……...… 21
2.6 Plateformes multi-composants…… 23
2.6.1 Les processeurs embarqués…..…….…….. 23
2.6.2 Les Outils de développement de xilinx……...….……...…….…... 26
2.7 Conclusion……..…..….. 27

Chapitre 3 : Implémentation de l'algorithme ECC sur le circuit FPGA

3.1 Introduction……...……. 28
3.2 Présentation de la carte FPGA…….……..…… 28
3.3 Outil software utilisé…….……....… 29
3.4 Réalisation du projet sous EDK……...…….. 29
3.4.1 Utilisation du Base System Builder de XPS…….…….…….. 29
3.4.2 Implémentation du matériel…… 34
3.4.3 Utilisation de l'outil SDK……..……. 36
3.5 Implémentation logicielle de l'algorithme ECC……...…… 36
3.5.1 Implémentation des fonctions d'addition, de multiplication, congruence et d'inversion ……..... 37
3.5.2 Implémentation du protocole d'échange de clés Diffie- Hellman…..… 39
3.5.3 Implémentation des fonctions de cryptage/décryptage ECC…….…….... 40
3.5.3.1 Cryptage…….… 40
3.5.3.2 Décryptage……..……... 40
3.5.4 Programme principal de test de l'algorithme de cryptage/décryptage ECC…….. 41
3.6 Résultats des tests pratiques sur la carte FPGA ……..…….……... 41
3.6.1 Résultats du 1er test…….……...…… 42
3.6.2 Résultats du 2eme test pratiques…….…… 43

3.7 Conclusion…… 44

Conclusion Générale…… 45

Références ……..……....…. 46

Annexe ……...…….……..…. 47

Liste des figures

Figure 1.1	Addition de points sur les courbes elliptiques ……..……	8
Figure 1.2	Etapes d établissement de clés symétriques …….…….	10
Figure 1.3	Protocole de chiffrement /déchiffrement ……...……	12
Figure 2.1	Les différents circuits programmables……...……	14
Figure 2.2	Architecture conceptuelle d'un FPGA (Architecture, Interconnections et CLB)…….……..	16
Figure 2.3	Flot de conception standard……	18
Figure 2.4	Flot de conception Xilinx……..…….	21
Figure 2.5	Le bus PLB dans un système à base de MicroBlaze……..……	22
Figure 2.6	Détails d'une plateforme multi-composants……...……...	23
Figure 2.7	Schéma-bloc fonctionnel du noyau MicroBlaze…….……....	25
Figure 3.1	système de test pratique ……..……...	45
Figure 3.2	carte DigilentNexys 2…….……	46
Figure 3.3	Les différents modules implémentés du système à base d'un Microblaze.…….	57
Figure 3.4	Démarrage application XPS ……..	57
Figure 3.5	création nouveau projet XPS ……..…	58
Figure 3.6	Choix carte FPGA …….	58
Figure 3.7	Sélection solution mono processeur ……..	59
Figure 3.8	sélection taille mémoire et fréquence d'horloge microblaze ……	9
Figure 3.9	sélection modules utiles ……...…….	11
Figure 3.10	Le navigateur des projets de Xilinx Platform Studio ……	12
Figure 3.11	Capture d'écran des ports utilisés ……...……...……	27
Figure 3.12	Capture d'écran des adresses réservées aux différents modules…	38
Figure 3.13	fichier de spécification de matériel MHS ……..……	39
Figure 3.14	fichier de contraintes UCF ……..……..	39
Figure 3.15	Schéma RTL du système à base du MicroBlaze implémenté ..….	41
Figure 3.16	Le navigateur des projets de Software Development Kit……...	42
Figure 3.17	Organigramme de la fonction *Diffie-Hellman*……...…….……...	42
Figure 3.18	Organigramme de la fonction $mult$ ……..……...	44
Figure 3.19	Organigramme de la fonction add …...…….	45

Figure 3.20 Organigramme du programme principal *main* ……..... 45
Figure 3.21 Organigramme de la fonction *congruence* …….……... 46
Figure 3.22 Organigramme de la fonction *inversion*…….....……. 57

Liste des tableaux

Tableau 1.1 Comparaison entre ECC et RSA …… 13

Tableau 3.1 Rapport des résultats concernant l'occupation de surface.. 48

Liste des Acronymes

AES	Advanced Encryption Standard
ASIC	Application Specific Integrated Circuit
CLB	Configurable Logic Block
CPLD$_S$	Complex Logic Programmable Device
DES	Data Encryption Standard
DCM	Digital Clock Manager
DCR	Device Control Register
DSP	Digital Signal Processors
ECC	Elliptic Curve Cryptography
ECS	Engineering Capture System
EDK	Embedded Development Kit
FPGA	Field-Programmable Gate Array
FPU	Floating Point Unit
HDL	Hardware Design Language
HW/SW	Hardware/Software
IP	Intellectual Property
ISE	Integrated Software Environment
LUT	Look Up Table
OPB	On-chip Peripheral Bus
PLB	Processor Local Bus
PLD	Programmable Logic Device
RSA	Rivest Shamir Adleman
RC4	Rivest Cipher 4
SoC	System on Chip
SoPC	System on Programmable Chip
VHDL	Very High Speed Integrated Circuit Hardware Description Language
XPS	Xilinx Platform Studio
XST	Xilinx Synthesis Technology

Introduction générale

Introduction générale

Le domaine de la cryptographie est un sujet qui s'est développé fortement ces dernières années, en raison d'un besoin croissant de sécurité dans les échanges commerciaux et plus généralement dans toutes les transactions des données numériques.

Les mécanismes proposés par la cryptographie moderne, permettent de répondre aux besoins de la sécurité de l'information par le déploiement de crypto systèmes symétriques et asymétriques. Dans la cryptographie symétrique, la clé utilisée pour chiffrer une donnée est la même clé que celle utilisée pour déchiffrer cette dernière. Elle est dite clé privée. Dans la cryptographie asymétrique, la clé utilisée pour chiffrer une donnée est différente de celle utilisée pour la déchiffrer. La clé du chiffrement est dite clé public, la clé du déchiffrement est dite clé privée.

Dans ce travail, nous nous intéressons à la cryptographie asymétrique. Les crypto systèmes à clé publique les plus connus de nos jours sont : le RSA (Rivest Shamir Adleman) et l'ECC (Elliptic Curve Cryptosystems). Ces crypto systèmes sont généralement construits sur des problèmes mathématiques difficiles à résoudre. Leur sécurité repose sur l'utilisation de grande taille de clés, pouvant varier de quelques centaines de bits jusqu'à des milliers de bits.

Dans le protocole de cryptographie RSA , les opérations de chiffrement et de déchiffrement sont basées sur le calcul de l'exponentiation modulaire, défient par $Y = X^z \ mod \ N$, X est la donnée à chiffrer ou à déchiffrer, Z est la clé public ou la clé privée, N est le modulo [7]. Dans le crypto-système ECC, les opérations de chiffrement et de déchiffrement sont basées sur le calcul de la multiplication scalaire, notée $Q = k \times P$, ou k est un scalaire et P est un point.

Notre objectif dans ce mémoire consiste à la conception et à l'implémentation sur un circuit FPGA, d'un crypto système à clé publique ECC, sur le processor softcore Microblaze de Xilinx .

Le mémoire est organisé comme suit :

- Le premier chapitre décrit les différents types de protocoles utilisés dans la cryptographie moderne. Une étude détaillée est réservée au protocole de cryptographie à clé publique ECC, objet principal des travaux menés dans ce projet.

- Le deuxième chapitre est réservé à l'étude des systèmes sur puce, plus particulièrement les systèmes implémentés sur des circuits FPGA.

- Le dernier chapitre présente la méthodologie développée pour la réalisation du crypto-système ECC, ainsi que les résultats pratiques.

Nous terminerons par une conclusion générale.

Chapitre 1

Cryptographie sur les courbes elliptiques (ECC)

1.1. Introduction

Ces dernières années, la technologie numérique ne cesse de croitre pour faciliter la vie de l'homme. Nous sommes donc entourés de données numériques qui transitent via des réseaux. En raison d'un besoin croissant de sécurité de ces transactions , le domaine de la cryptographie s'est fortement développé. De nos jours plusieurs protocoles de cryptographie existent. On peut citer entre autres le RSA, l'ECC et l'AES etc.

Dans notre travail, on s'intéresse à un protocole de cryptographie asymétrique basé sur les courbes elliptiques ECC. Dans ce chapitre nous allons présenter les concepts généraux de la cryptographie. Ensuite nous étudions les notions de base qui permettent de manipuler les courbes elliptiques pour le chiffrement et le déchiffrement des données.

1.2. La cryptographie

La cryptographie est une solution de sécurité qui est relativement beaucoup plus sûre. L'idée de base est d'utiliser des méthodes mathématiques pour transformer le message original à une suite de données qui ne peuvent pas être directement interprétées par une tierce partie. Elle comprend un ensemble de techniques qui sont fréquemment utilisées dans le monde informatique pour assurer la confidentialité, l'intégrité et l'authenticité des données[1].

Il existe principalement deux catégories de cryptographie, qui sont respectivement la cryptographie symétrique et la cryptographie asymétrique.

1.2.1. Cryptographie symétrique (chiffrement à clé privée)

La cryptographie symétrique utilise une seule clé pour le chiffrement et le déchiffrement des données. Elle est considérée comme une solution moins coûteuse et plus facile à implémenter.

On suppose que M est le message à chiffrer et k est la clé secrète partagée par l'envoyeur et le récepteur du message. L'envoyeur utilise la fonction de cryptage $E()$ et la clé k pour générer un message chiffré $E(M,k) = M'$.

Après la réception du message chiffré, le récepteur utilise une fonction de décryptage $D()$ et la même clé k pour déchiffrer le message $D(M', k) = M$.

Cependant cette solution possède aussi des inconvénients inhérents. Pour que deux nœuds puissent communiquer entre eux, il faut qu'ils utilisent exactement la même clé, et la distribution de clés est devenue un défi inévitable [1].

Les crypto systèmes à clé privée les plus utilisés sont :

- AES (Advanced Encryption Standard).
- DES (Data Encryption Standard).
- RC4 (Rivest Cipher 4).

1.2.2. Cryptographie asymétrique (chiffrement à clé public)

Le problème de distributions des clés est résolu par la cryptographie asymétrique. Ce concept a été développé par Whlifield Diffie et Martin Hellman [4]. Ce type de cryptographie utilise deux clés différentes respectivement pour le chiffrement et le déchiffrement. Avant l'envoi du message M, on le chiffre avec la clé publique du destinataire kp, qui est diffusée librement dans l'ensemble du réseau, $E(M, kp) = M'$ où $E()$ est la fonction de chiffrement et M' est le message chiffré.

Après la réception de M', le destinataire utilise sa clé privée ks, qui est gardée secrète, pour déchiffrer le message, $D(M', ks) = M$ où $D()$ est la fonction de décryptage. La cryptographie asymétrique donne aussi la possibilité de signer des messages chiffrés[1].

Les crypto systèmes à clé public les plus utilisés sont :

- RSA (Rivest Shamir Adleman).
- ECC (Elliptic Curve Cryptography).

Dans ce travail, nous nous somme basées sur la deuxième crypto-système. Et nous allons présenter les courbes elliptiques et leurs notions pour maitriser bien ce type de crypto système.

1.3. Généralités

Avant de présenter les courbes elliptiques, nous étudions d'abord les notions mathématiques dont nous avons besoin pour comprendre son fonctionnement.

1.3.1. Groupe

En mathématique, un groupe est un couple $(E,\bullet)$ où E est un ensemble et $\bullet$ est une loi de composition interne qui combine deux éléments a et b de E pour obtenir un troisième élément $a \bullet b$. Il faut que la loi satisfasse les quatre axiomes ci-dessous [1].

- Fermeture : $\forall (a,b) \in E \mid a \bullet b \in E$
- Associativité : $\forall (a,b) \in E \mid (a \bullet b) \bullet c = a \bullet (b \bullet c)$
- Élément neutre : $\exists e \in E \mid a \bullet e = e \bullet a = a$
- Symétrique : $\forall a \in E, \exists b \in E \mid a \bullet b = b \bullet a = e$

1.3.2. Groupe abélien

Un groupe abélien, ou un groupe commutatif, est un groupe dont la loi de composition interne est commutative. Un ensemble E est un groupe commutatif lorsque[1]:

$\forall (a,b) \in E | a \bullet b = b \bullet a$

1.3.3. Groupe cyclique

Un groupe fini G est cyclique si tout élément du groupe peut s'exprime sous forme d'une puissance ou d'un multiple d'un élément particulier g, appelé le générateur du groupe, c'est-à-dire $\{G = < g > = g^n | n \in Z^*\}$. Par exemple si : $G = \{g^0, g^1, g^2, g^3, g^4, g^5, g^6\}$ et $g^6 = g^0$, alors G est un groupe cyclique. Tout groupe cyclique est abélien car $g^n g^m = g^{n+m} = g^{m+n} = g^m g^n$.

L'ordre d'un élément e d'un groupe cyclique est le nombre entier n positif tel que $n * e = 0$ (en notation additive) ou $e^n = 1$ (en notation multiplicative). Reprenons le même groupe G du paragraphe précédent, par exemple l'ordre de l'élément g^2 est 3 car l'élément neutre du groupe est $g^0 = 1$ et $(g^2)^3 = g^6 = 1$ [1].

1.3.4. Anneau unitaire et anneau commutatif

Un anneau unitaire, ou simplement anneau, est un ensemble Emuni de deux lois decomposition, notées $+$ (addition) et $\bullet$ (multiplication). E est un anneau, si [1]:

- $(E, +)$ est un groupe commutatif.
- La loi $\bullet$ est associative et distributive par rapport à la loi $+$.
- La loi $\bullet$ possède un élément neutre.

Il existe un élément neutre de la loi de composition +, noté 0 tel que :

$$\forall a \in E \mid a + 0 = 0 + a = a$$

Et

$$\forall a \in E, \exists b \in E \mid a + b = 0.$$

Il existe un autre élément neutre pour la loi de composition $\bullet$, noté 1 tel que :

$$\forall a \in E \mid a \bullet 1 = 1 \bullet a = a.$$

Un anneau commutatif est un anneau dont la loi de composition $\bullet$ est commutative, c'est- à-dire :

$$\forall (a, b) \in E \mid a \bullet b = b \bullet a.$$

1.3.5. Corps fini

Un corps fini F est un corps dont le nombre d'éléments est fini. Le nombre d'éléments est l'ordre du corps, noté q, qui peut être représenté par la puissance d'un nombre premier $q = p^n$, où p est un nombre premier, appelé la caractéristique du corps, et $n \in Z^+$. Pour étudier la cryptographie sur les courbes elliptiques, il faut que nous comprenions les deux types de corps ci-dessous [1].

1.3.5.1. Corps premier

Un corps est un corps premier, noté Fp lorsque l'ordre du corps $q = p^n$ et p est un nombre premier. Le corps est constitué des nombres entiers {0, 1, 2, ..., p − 1} et $\forall a \in Z$, $a \bmod p$ donne le reste unique r qui est compris entre [0, p − 1] [1].

1.3.5.2. Corps binaire

Un corps fini de l'ordre2^n est un corps binaire, noté F_2^n , qui peut être construit en utilisant une représentation polynomiale. Les éléments du corps sont des polynômes binaires dont les coefficients $a_i \in \{0,1\}$ et les degrés sont inférieurs à n. C'est-à-dire $F_2^n = a_{n-1}z_{n-1} + a_{n-2}z_{n-2} + \ldots + a_1 z + a_0 : a_i \in \{0,1\}$ [1].

1.4. Présentation de la courbe elliptique

Après la présentation des notions mathématiques nécessaire, nous allons passer, dans cette section, à la définition des courbes elliptiques avec l'ensemble d'opérations que nous pouvons effectuer sur elles.

1.4.1. Définition de courbe elliptique

La courbe elliptique E est une courbe algébrique qui peut être représentée par l'équation Weierstrass simplifiée suivante [1]:

$$Y^2 = x^3 + ax + b$$

Où a, b $\in F_p$. Le discriminant de la courbe $\Delta = -16(4a^3 + 27b^2)$ et $\Delta \neq 0$.

1.4.2. Loi de composition sur les courbes elliptiques

Nous supposons qu'une courbe elliptique E est définie dans un corps premier F_p, L'ensemble de points sur la courbe avec le point à l'infini, noté O, forment un groupe abélien G dont la loi de composition est l'addition de point satisfaisant les conditions suivantes [1] :

- Fermeture : $\forall (P1, P2) \in G \mid P1 + P2 \in G$.
- Commutativité : $\forall (P1, P2) \in G \mid P1 + P2 = P2 + P1$.
- Associativité: $\forall (P1, P2, P3) \in G \mid (P1 + P2) + P3 = P1 + (P2 + P3)$.
- Elément neutre O: $\forall P \in G \mid P + O = O + P = O$.
- Elément symétrique : $\forall P \in G, \exists Q \in G \mid P + Q = Q + P = O$.

Pour obtenir le point symétrique de P, il suffit de changer le signe de sa coordonnée y, si

$$P = (x, y), alors - P = (x, -y) et P + (-P) = O.$$

Le calcul de l'addition de point est montré dans les formules (1.1), (1.2). Supposons que :$P = (x_1, y_1) \in E, Q = (x_2, y_2) \in E$ et P ≠Q, alors $P + Q = (x_3, y_3)$ où

$$x_3 = \left(\frac{y_2 - y_1}{x_2 - x_1}\right)^2 - x_1 - x_2 \; et \; y_3 = \left(\frac{y_2 - y_1}{x_2 - x_1}\right)(x_1 - x_3) - y_1 \qquad (1.1)$$

$Si\ P = (x_1, y_1) \in E$ et $P \neq -P$, alors$2P = (x_3, y_3)$ où

$$x_3 = \left(\frac{3{x_1}^2 + a}{2y_1}\right)^2 - 2x_1 \; et \; y_3 = \left(\frac{3{x_1}^2 + a}{2y_1}\right)(x_1 - x_3) - y_1 \qquad (1.2)$$

Une représentation géométrique est donnée dans la figure 1.1. Pour additionner les points P et Q, nous traçons une droite qui passe par ces deux points, le résultat de l'addition est le point symétrique par rapport à l'axe abscisse du troisième point d'intersection avec la courbe. Pour doubler le point P, c'est-à-dire 2P, il suffit de trouver la tangente à la courbe au point P, et le résultat du doublement est le point symétrique par rapport à l'axe abscisse du deuxième point d'intersection avec la courbe [1].

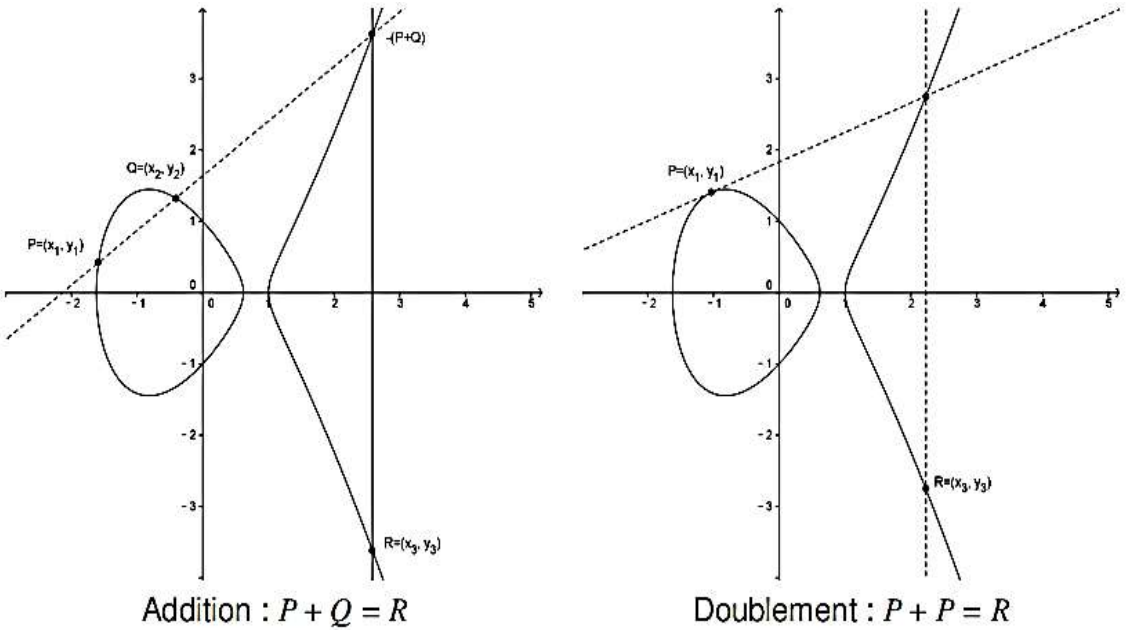


Figure 1.1: Addition de points sur les courbes elliptiques [1].

1.4.3. Multiplication de point

En se basant sur l'addition de point, nous pouvons encore effectuer la multiplication de point, notée $Q = k \times P$ sur un courbe elliptique E où $k \in Z^+$ et $(P, Q) \in$ E. cette opération est appelée aussi la multiplication scalaire, qui est considérée comme l'opération la plus coûteuse et importante sur les courbes elliptiques. Une multiplication de point peut être considérée comme une suite d'additions de point consécutives:

$$Q = k \times P \underbrace{= P + P + \cdots + P}_{k\ fois\ P}$$

La méthode pour calculer la multiplication scalaire est d'utiliser l'algorithme de doublement-et-addition (voir algorithme 1.1). Supposons que P est un point sur une courbe elliptique qui est définie sur un corps premier, notée $E(F_p)$. Pour calculer $k \times P$ où k est un nombre entier positif de longueur l bits, nous représentons k en binaire $\sum_{i=0}^{l-1} k_i\, 2^i$. Ensuite nous parcourons k du bit de poids faible au bit de poids fort.

L'algorithme qui permet de calculer la multiplication scalaire est défini comme suit [1]:

Algorithme 1.1 : Algorithme doublement-et-addition pour calculer $Q = k \times P$

Données: $k = \sum_{i=0}^{l-1} k_i\, 2^i$ *et* $P \in E(F_p)$
Résultat : $Q = k \times P$

1. $\boldsymbol{Q} \leftarrow O$;
2. ***Pour*** i *de* 0 à $l - 1$ ***faire***
3. ***Si*** $k_i = 1$ ***alors***
4. $Q \leftarrow Q + P$
5. ***Fin***
6. $P \leftarrow 2P$
7. ***Fin***
8. ***Retourner*** Q

1.5. Protocoles cryptographiques basés sur ECC

Grâce au problème de logarithme discret, plusieurs Protocoles cryptographiques ont été proposés .Dans cette section, nous présentons les protocoles principaux (échange de clés, chiffrement et déchiffrement).

1.5.1. Protocole d'échange de clés de Diffie-Hellman

Soit un grand nombre premier p et deux paramètres a, b satisfaisant l'équation $(Y^2 mod\ p) = \big((x^3 + ax + b)\ mod\ p\big)$. a et b permettent de définir un $Ep(a,b)$ d'ordre n très élevé. On choisit au hasard un point de départ $G(x_1, y_1)$ dans $Ep(a,b)$ qu'on rendra publique avec $Ep(a,b)$. L'échange de clés entre les deux entités communicantes A et B se réalise comme suit [2] :

- A choisit n_A<n et B choisit n_B<n respectivement comme clé secrète.
- A calcule sa clé publique $P_A = n_A \times G$ et B calcule également sa clé publique $P_B = n_B \times G$.
- A et B échangent publiquement leurs clés publiques P_A et P_B.
- A génère sa clé privée $k = n_A \times P_B$ et B génère sa clé privée $k = n_B \times P_A$.
- La clé k est la clé symétrique partagée de manière sécurisée par A et B.

La figure 1.2 illustre les étapes d'établissement de clés entre les nœuds A et B.

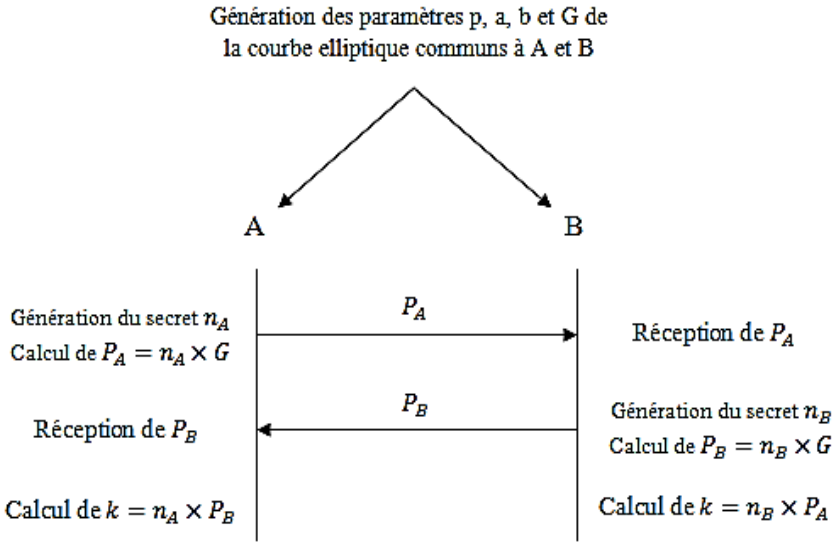


Figure 1.2 : Etapes d établissement de clés symétriques.

1.5.2. Protocole de chiffrement et déchiffrement

La cryptographie à base de courbes elliptique peut être utilisée pour chiffrer un message M, en un message M'. M est transformé en un ensemble de points P_m à partir de l'ensemble de points fini de la courbe elliptique $Ep(a, b)$.

La première étape consiste à choisir un point générateur, $G \in Ep(a, b)$, tel que la plus petite valeur de n pour laquelle $nG = O$ est un très grand nombre premier. L'ensemble des points $Ep(a, b)$ et G sont rendus public. Les utilisateurs doivent également choisir une clé privée et générer la clé publique correspondante.

Pour chiffrer le message, A détermine aléatoirement un nombre entier positif k et produit C_m comme un couple de points en utilisant la clé public de B P_B.

$$C_m = [(kG), (P_m + k\ P_B)] \tag{1.3}$$

Après la réception des paires de points du message crypté C_m, B multiplie le premier point kG, avec sa clé privée n_B , et soustraire le résultat au second point reçu :

$$P_m + k\ P_B - n_B(kG) = P_m + k(n_B G) - n_B(kG) = P_m \tag{1.4}$$

Seulement B connait la clé privée n_B, peut retrancher $n_B(kG)$ du deuxième point du message chiffré [2].

Il est important qu'A utilise, à chaque fois qu'elle envoie un message crypté à B avec la même clé, un k différent. En effet, si elle utilise le même k pour deux messages différents P_m et P_m', alors (kG) = (kG)'. Un espion ayant intercepté les deux messages codés s'en apercevra et pourra calculer

$(P_m + k\ P_B)' - (P_m + k\ P_B) = P_m' + k\ P_B - (P_m + k\ P_B) = P_m' - P_m$.

Supposons que pour une raison quelconque le message P_m soit rendu public dès que l'information n'est plus d'actualité, alors l'espion calculera sans peine P_m' qui vaut

$P_m - (P_m + k\ P_B) + (P_m + k\ P_B)' = P_m - P_m - k\ P_B + P_m' + k\ P_B = P_m'$ [5].

Le protocole de chiffrement et de déchiffrement sont montré sur la figure 1.3.

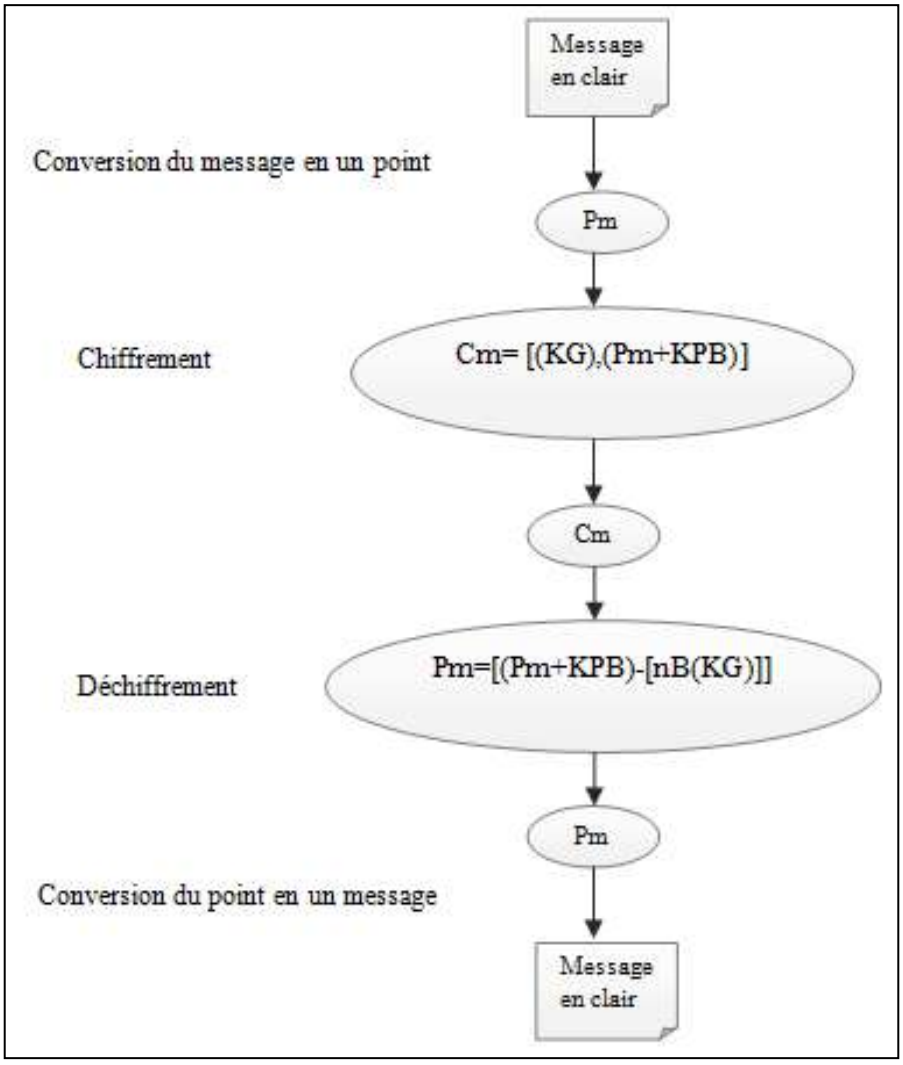


Figure 1.3 : Protocole de chiffrement /déchiffrement.

1.6. Comparaison avec le crypto-système de RSA

La cryptographie sur les courbes elliptiques a un avantage majeur : elle nécessite des clés de taille beaucoup plus petite que RSA. Ceci en fait un concurrent très sérieux pour les systèmes nécessitant un grand niveau de sécurité mais qui disposent de peu de ressources.

Le site de la National Security Agency donne un tableau comparatif des tailles de clé pour un système à clé privée, un système basé sur RSA et un système fondé sur les courbes elliptiques. Chaque ligne correspond à un niveau de sécurité identique. Qui ce présente comme suit :

Tableau 1.1 : Comparaison entre ECC et RSA [2].

Taille de la clé symétrique	Taille de la clé RSA	Taille d'une clé d'ECC
80	1024	160-223
112	2048	224-255
128	3072	256-383
192	7680	384-511
256	15360	512+

On constate que la taille des clés RSA augmente beaucoup plus rapidement que la taille des clés sur les courbes elliptiques. C'est l'avantage principal de la cryptographie sur les courbes elliptiques. C'est ce qui amène ce crypto-système à déloger progressivement le crypto-système RSA.

1.7. Conclusion

Dans ce chapitre, nous avons présenté le protocole de cryptographie à base de courbes elliptiques ainsi que les notions mathématiques de base pour la compréhension des courbes elliptiques ECC. Nous avons constaté que le fonctionnement de ce genre de crypto système est basé sur la multiplication scalaire. Celle-ci est utilisée non seulement pour la génération des clés, mais aussi lors du chiffrement et du déchiffrement des données. Le chapitre suivant est consacré à l'étude des systèmes sur puce, en particulier les systèmes implémentés sur carte FPGA.

Chapitre 2

Conception des systèmes numériques embarqués

2.1. Introduction

Un système embarqué est défini comme un système électronique et informatique autonome, souvent temps réel, spécialisé dans une tâche bien précise. Le terme désigne le matériel informatique ainsi que le logiciel utilisé. Ses ressources sont généralement limitées. Cette limitation est d'ordre spatial (encombrement réduit) et énergétique (consommation restreinte) [6].

2.2. Les circuits logiques programmables [6]

Un circuit logique programmable, est un circuit logique intégré qui peut être reprogrammé après sa fabrication. Il est composé de nombreuses cellules logiques élémentaires librement assemblables. Un circuit logique programmable permet la réalisation de circuits logiques à grand nombre de variables d'entrée et de sortie, selon la spécification de l'utilisateur et en utilisant un même support matériel. Actuellement, on trouve différentes familles de circuits programmables tels que les PLD (Programmable Logic Device), les CPLDs (Complex Logic Programmable Device) et les FPGAs. La figure 2.1 montre les différentes familles des circuits programmables.

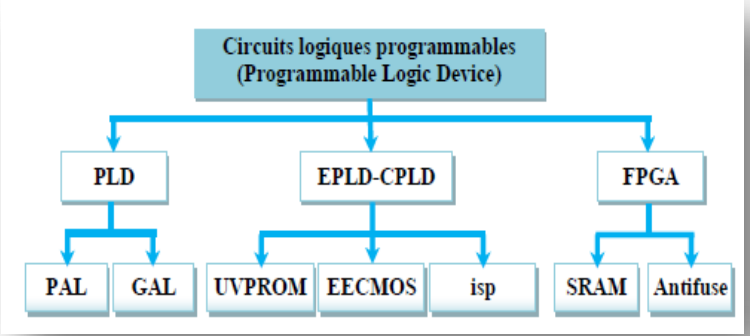


Figure 2.1 : Les différents circuits programmables.

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

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

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

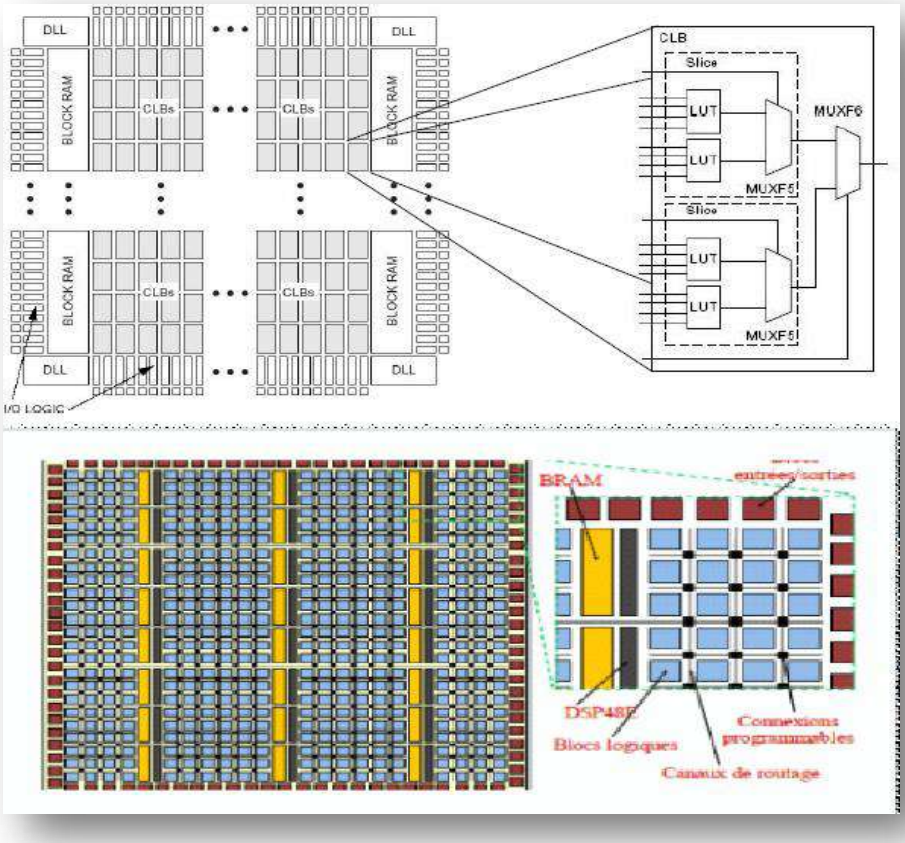


Figure 2.2 : Architecture conceptuelle d'un FPGA (Architecture, Interconnexions et CLB).

2.3. Les systèmes numériques embarqués [6]

De nos jours, les systèmes électroniques embarqués ont été introduits dans de nombreux domaines d'applications tels que l'automobile, l'avionique, les systèmes multimédia, les appareils électroménagers, etc. L'architecture générale d'un système sur puce intègre trois principaux types de composants qui sont : les unités de traitements, les mémoires et les systèmes de communication.

2.3.1. Codesign : Conception conjointe logicielle/matérielle

Les techniques classiques de conception de systèmes imposent de séparer le hardware du software dans le cycle de conception. En outre, le consensus grandissant pour un modèle de systèmes informatiques «en couches» semble éviter aux développeurs software de telles préoccupations. Ainsi, les concepteurs de hardware spécifient leurs produits sans totalement apprécier les besoins, en ressources matérielles, des logiciels qui s'exécuteront dessus. De leur

côté, les concepteurs de logiciels ne tirent pas suffisamment profit des ressources matérielles de la plate-forme hardware sur laquelle seront implantés leurs produits. Finalement, la phase d'intégration des deux parties nécessite souvent des modifications parfois majeures du logiciel et/ou du matériel. Cette approche présente peu de flexibilité dans l'évaluation des différentes options de conception et de répartition du hardware et du software. La tendance actuelle, principalement lorsqu'il s'agit de systèmes temps réel complexes, est de tenir compte dès la phase de spécification des interactions entre les deux parties. Ainsi, la séparation est effectuée le plus tard possible dans le cycle de conception tendant à unifier la spécification pour qu'elle englobe le hardware et le software. Cette approche est nommée "hardware/software codesign", ou "codesign". Le système est conçu à partir d'une spécification unique décrivant son architecture et/ou son comportement. Après la phase de spécification survient une phase de partitionnement du système ayant pour but de décomposer ce dernier en partitions comportant trois parties :

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

2.3.2. Flot de la conception conjointe logicielle/matérielle

Le développement de systèmes complexes fait de plus en plus appel à des composants logiciels et spécifiques. La conception conjointe logicielle/matérielle apporte des solutions à ce type de développement. Un flot de conception standard est généralement constitué de trois étapes principales : la spécification, le partitionnement, et la vérification conjointe HW/SW (Figure 2.3).

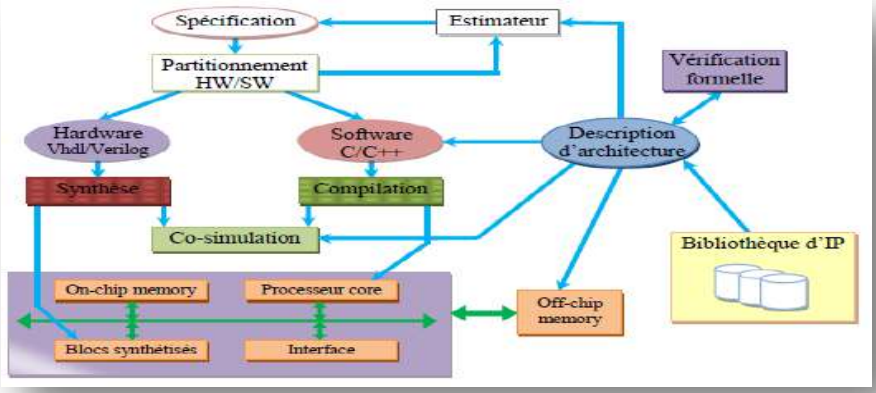


Figure 2.3 : Flot de conception standard.

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

2.3.3. Les composants virtuels (IPs)

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

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

- **IP Logiciel** (softcore) : Le composant est livré sous sa forme HDL synthétisable. Son principal avantage est sa portabilité. La propriété du fichier source est en soi la meilleure documentation, on peut ainsi maintenir le produit pendant des années et éventuellement modifier à volonté la source et même changer de technologie cible. L'inconvénient majeur est qu'il ne peut être prédictif en termes de superficie, puissance et temps. Le travail d'optimisation du circuit final est à la charge de l'intégrateur du système.

- **IP Matériel** (hardcore) : Dans ce cas, la description du bloc IP est au niveau porte logique, optimisée pour une technologie particulière (optimisation garantie). Cela englobe la netlist entière, le routage et l'optimisation pour une librairie technologique spécifique et un lay-out personnalisé. L'inconvénient est qu'il est moins flexible car le processus est dépendant de la technologie, par contre il a l'avantage d'être prédictif.

- **IP Firm** (firmcore) : Le bloc IP firmcore offre un compromis entre les deux autres IPs, plus flexible que le hardcore, plus prédictif en termes de performance et de surface que le softcore (par exemple une interface USB où la vitesse de la norme ne pourra être assurée que dans une technologie bien définie). En général, le travail de synthèse HDL est déjà réalisé pour une technologie cible donnant lieu à une description par netlist (format EDIF par exemple).

2.4. Les systèmes sur puce (SoC) [6]

2.4.1. Les contraintes liées aux systèmes embarqués

Les contraintes lors de la construction d'un système embarqué sont relatives au temps, imposés par l'interaction du système avec son environnement et aussi au caractère limité et/ou périssable des ressources dont dispose le système (énergie, mémoire, etc.). A ces contraintes s'ajoutent des contraintes de sûreté de fonctionnement plus ou moins importantes selon le type d'utilisation du système embarqué. En plus, un système embarqué exécute des tâches prédéfinies et a un cahier des charges contraignant à remplir, qui peut être de l'ordre de coût, d'espace compté ou bien les contraintes physiques (le système doit présenter un faible encombrement et un faible poids) ; de puissance de calcul ; de consommation énergétique (la plus faible possible), etc. Des choix doivent donc être réalisés pour privilégier un ou plusieurs de ces critères. Par exemple, pour un ordinateur portable, l'autonomie est souvent négligée

par rapport aux contraintes de performances et de taille. En effet, un ordinateur peu performant perd tout son intérêt, d'autant plus qu'il est souvent possible de travailler à proximité d'une prise de courant. Pour un téléphone portable au contraire, l'intérêt étant de pouvoir téléphoner de partout, l'autonomie est un critère primordial.

2.4.2. Les SoC et les SoPC

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

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

- **L'approche SoPC** (technologie FPGA) : Spécialement pour résoudre les problèmes de développement et les prototypages rapides, ainsi que les composants reconfigurables en quelques « ms » et à volonté. Il permet donc un développement et prototypage rapide du système. En contrepartie, la densité d'intégration est généralement moindre (~10 Millions de portes) et la consommation d'énergie est plus grande avec une performance plus faible que celle du SoC.

Les composants FPGA permettent d'implanter deux types de SoPC. Le premier type de SoPC utilise uniquement des IP soft de processeur tels que le MicroBlaze et le NIOS. Le nombre de processeurs intégrés sur un même composant dépend des ressources du composant utilisé. Le deuxième type de composants en plus de la logique reconfigurable, implémente en dur un ou plusieurs cœurs de processeurs (PowerPC pour Xilinx, ARM pour Altera). Ceci

n'empêche en rien l'utilisation de processeurs Soft (netlist). Ces deux types de SoPC utilisent les mêmes flots de conception. Chez Xilinx, tout le flot de conception est regroupé sous la forme d'un seul environnement dit Xilinx Platform Studio (XPS).

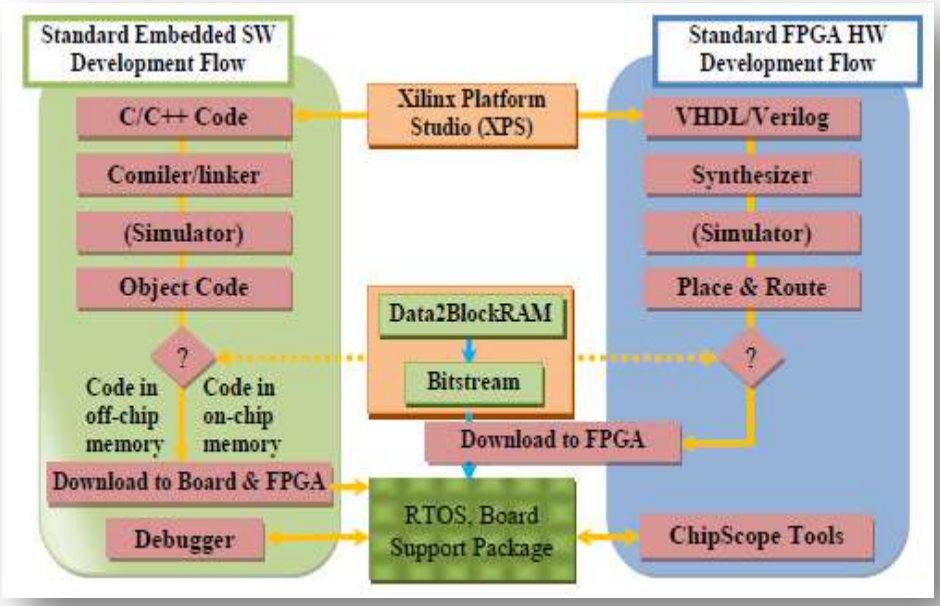


Figure 2.4 : Flot de conception Xilinx.

2.5. Communications sur puce [6]

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

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

- **Processor Local Bus (PLB)** : Généralement, le PLB fournit un chemin de données large bande. En effet il est utilisé pour connecter des composants tels que les cœurs de processeurs, les interfaces avec les mémoires externes, et les contrôleurs DMA (Direct Memory Accès).

- **On-chip Peripheral Bus (OPB)** : L'OPB est utilisé pour réduire la charge du bus PLB. Il est plus adapté aux composants tels que les ports série, les ports parallèles, UARTs, GPIO et tous les composants à faible largeur de bande. Un Maître sur le PLB peut accéder aux composants connectés à l'OPB via une macro passerelle (Bridge). Cette passerelle est vue en tant que maître par l'OPB et en tant qu'esclave par le PLB. Les registres d'état et de configuration de faible performance sont généralement lus et écrits via le bus DCR.
- **Device Control Register (DCR)** : Le DCR est principalement utilisé pour accéder aux registres de l'état et de contrôle au sein des différents bus PLB et OPB.

Xilinx a repris le comportement de tous les bus CoreConnect d'IBM et les a décrits en VHDL, ce qui permet d'avoir des bus paramétrables. En effet, pour simplifier la connexion d'un module logique utilisateur à un des bus CoreConnect, Xilinx a ajouté une interface de bus préconçue paramétrable appelée IPIF (IP InterFace). L'IPIF gère les signaux du bus, le protocole de communication et plus généralement, l'ensemble des caractéristiques du bus. L'IPIF intègre aussi une interface de gestion, du logique utilisateur, appelée IPIC (IP InterConnect). Lorsque le logique utilisateur est conçu avec un IPIC, elle peut être portable et facilement réutilisable sur d'autres bus en changeant seulement l'IPIF. Des fichiers VHDL qui instancient l'IPIF et fournissent le code nécessaire à l'utilisateur pour ajouter ses modules, simplifient la tâche de connexion du logique utilisateur. Ces fichiers sont les User Core Reference Design. Un système de traitement réalisé par EDK doit être connecté le bus OPB et/ou le bus PLB, le périphérique ajouté au système (accélérateur matériel) doit être connecté à OPB ou bien PLB. La figure 2.5 montre la connexion du bus PLB dans un système à base d'un MicroBlaze, d'une DDR2 et d'une mémoire Compact Flash.

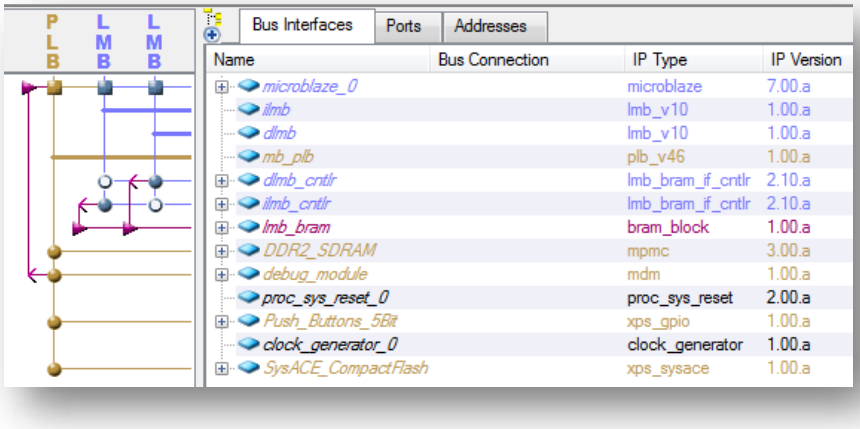


Figure 2.5: Le bus PLB dans un système à base de MicroBlaze.

2.6. Plateformes multi-composants [6]

La maîtrise des Socs nécessite des compétences en électronique et en informatique d'un niveau assez élevé. Pour aider les nouveaux utilisateurs de SoCs à bien maîtriser l'ensemble de la chaîne de mise en place d'applications, des plateformes de prototypage rapide existent aujourd'hui sur le marché. Ces plateformes utilisent un gros FPGA comme cœur et disposent, en plus, de mémoires SRAM, de convertisseurs A/N et N/A, des interfaces (boutons poussoirs, voyants LED, connecteurs d'usage général, connecteurs pour debug, zone de prototypage, connecteur RS232 ainsi qu'un connecteur JTAG), d'oscillateurs, sortie VGA, etc. Généralement, la configuration des composants se fait par liaison JTAG [6].

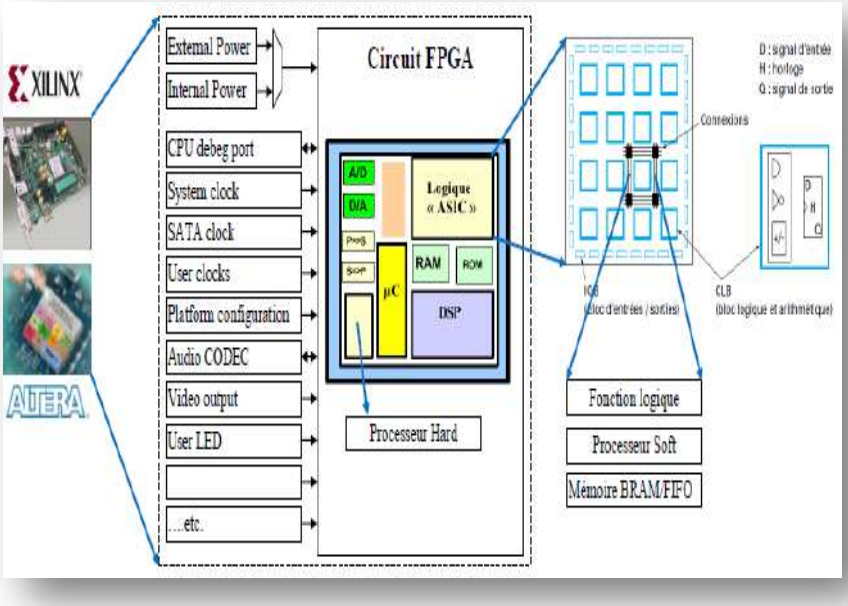


Figure 2.6 : Détails d'une plateforme multi-composants.

2.6.1. Les processeurs embarqués

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

performances en fonctions des différents environnements rencontrés (Outils de développement, taille de la cible ou électronique adjacente).

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

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

a. Le processeur MicroBlaze de Xilinx

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

- 32 registres internes de 32 bits,

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

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

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

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

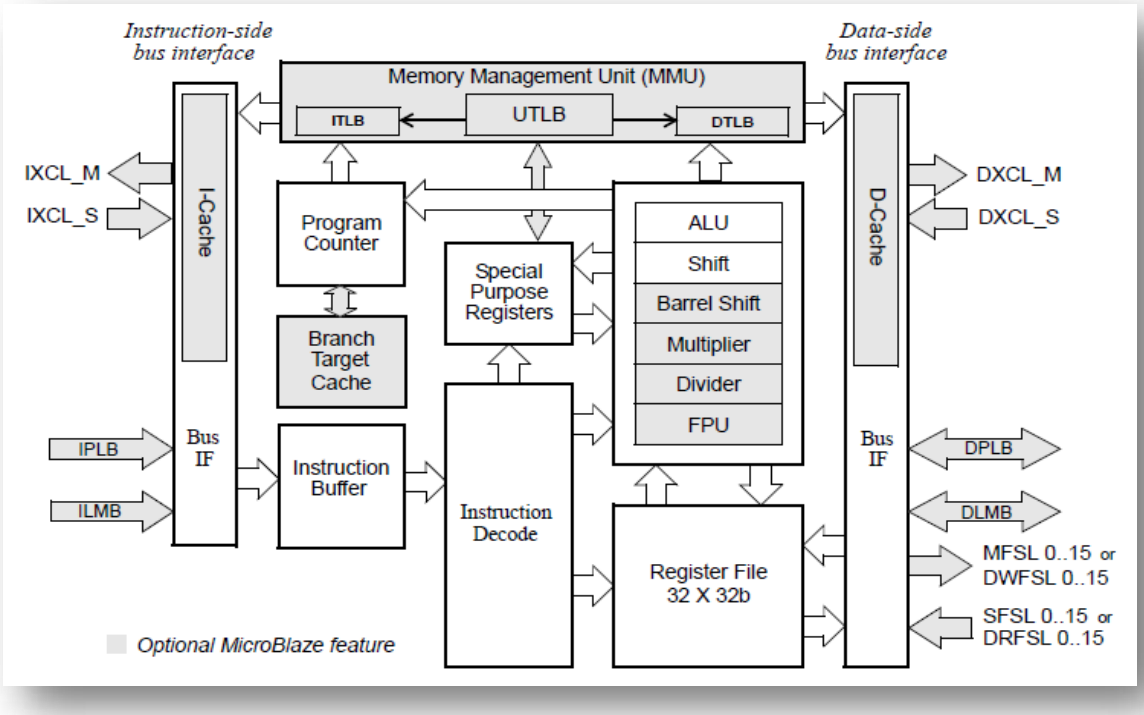


Figure 2.7: Schéma-bloc fonctionnel du noyau MicroBlaze.

2.6.2. Les Outils de développement de Xilinx

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

a. Xilinx ISE

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

1. Saisie de conception (Design Entry) : Editeur HDL, éditeur de Machine à Etat Fini (StateCAD), éditeur de schéma (Engineering Capture System (ECS)) et générateur d'IP (CORE Generator) ;
2. Synthèse : XST (Xilinx Synthesis Technology), Leonardo Spectrum de Mentor Graphics et Synplify de Cadence ;
3. Simulation : Générateur de TestBench HDL et intégration du simulateur ModelSim de Model Technology ;
4. Implémentation : Translate, MAP, placement et routage (PAR), floor planner, éditeur FPGA, timing Analyzer XPower, Fit (seulement CPLD) et Chipviewer (seulement CPLD) ;
5. Programmation du composant et formatage du fichier de Bitstream : BitGen et iMPACT.

b. Xilinx EDK

L'environnement EDK permet de développer une application complète à base de processeur embarqué et de l'intégrer à un FPGA. EDK donne accès à tous les réglages nécessaires pour l'application embarquée que l'on souhaite créer. Il permet la programmation d'un ou plusieurs MicroBlazes et de leurs périphériques. Il va s'occuper de placer les programmes des MicroBlazes en mémoire lors de la programmation du FPGA.

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

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

2.7. Conclusion

Dans ce chapitre, nous avons exposé la conception des systèmes numériques embarqués. Ensuite nous avons décrit les circuits logiques programmables, en particulier les circuits FPGAS. En plus, la conception conjointe (logicielle et matérielle) a été détaillé. En fin, les outils de développement de xilinx (ISE et EDK) ont été présentés.

Chapitre 3

Implémentation de l'algorithme ECC sur le circuit FPGA

3.1. Introduction

Dans ce chapitre, on présente une implémentation logicielle de l'algorithme de cryptage et décryptage ECC sur une carte de développement basée sur un circuit FPGA de type Xilinx Spartan 3E-500. Enfin, on expose les résultats de différents tests pratiques réalisés entre la carte FPGA et un PC interfacés entre eux par le port série (RS232) (Figure 3.1).

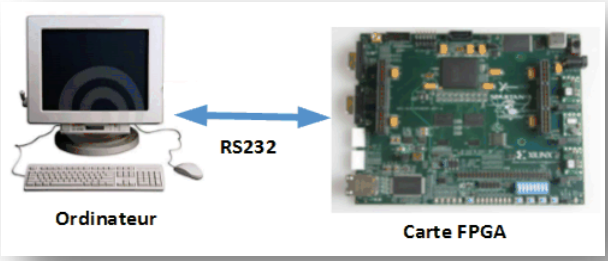


Figure 3.1: système de test pratique.

3.2. Présentation de la carte FPGA

La carte « Digilent Nexys2 » est une plate-forme complète pour le développement des systèmes basés sur les circuits FPGA Xilinx Spartan 3E. Elle est idéale pour les systèmes numériques de toutes sortes, y compris les systèmes avec processeurs embarqués basés sur le MicroBlaze de Xilinx. Cette carte possède les caractéristiques suivantes [8]:

- FPGA de type Xilinx Spartan 3E de 500K portes.
- Configuration FPGA basée sur l'USB 2.
- 16 Mo de SDRAM Micron et 16 Mo de ROM Intel Strata flash.
- Oscillateur 50MHz.
- 8 LEDs, afficheur 7 segments 4 chiffres, 4 boutons, 8 commutateurs à glissière.

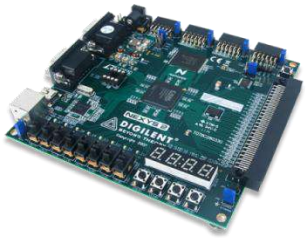


Figure 3.2: carte DigilentNexys 2.

3.3. Outil software utilisé

Nous avons utilisé l'environnement de développement EDK de Xilinx englobant les deux outils XPS et SDK, présentés en deuxième chapitre Avec le Base System Builder, un outil d'assistance de XPS, on pourra créer rapidement un nouveau projet. XPS utilise les outils fournis avec ISE pour synthétiser, placer et router la logique dans le matériel. En outre, d'autres outils sont fournis avec l'EDK (comme le SDK) et utilisés pour concevoir la partie software pour le système embarqué à base du Microblaze.

Les modules du système Microblaze, implémentés dans cette partie sont les suivants :

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

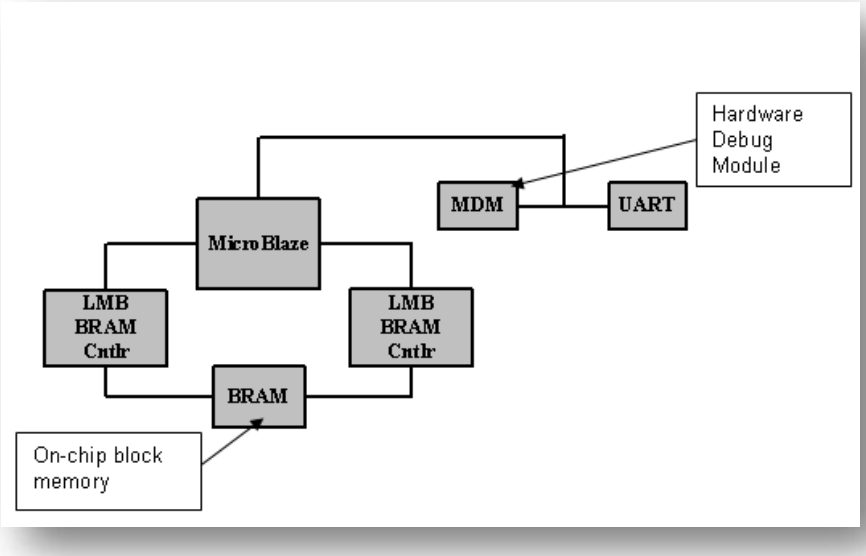


Figure 3.3 : Les différents modules implémentés du système à base d'un Microblaze.

3.4. Réalisation du projet sous EDK

3.4.1. Utilisation du Base System Builder de XPS

On crée un nouveau projet en utilisant « Base System Builder » (Figures 3.4 et 3.5).

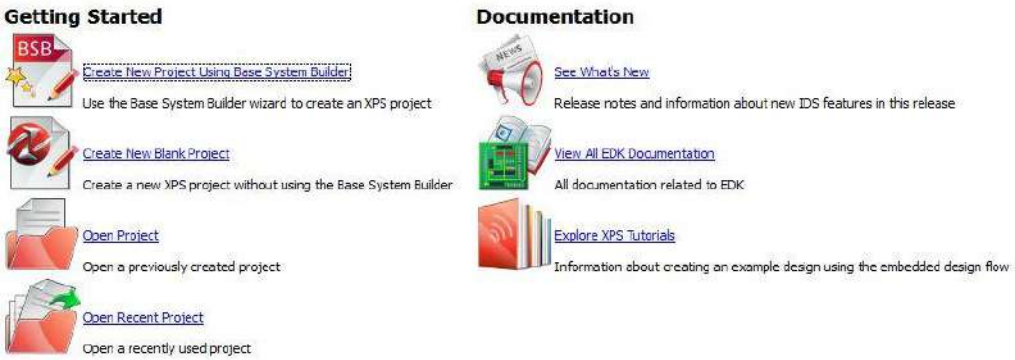


Figure 3.4: Démarrage application XPS.

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

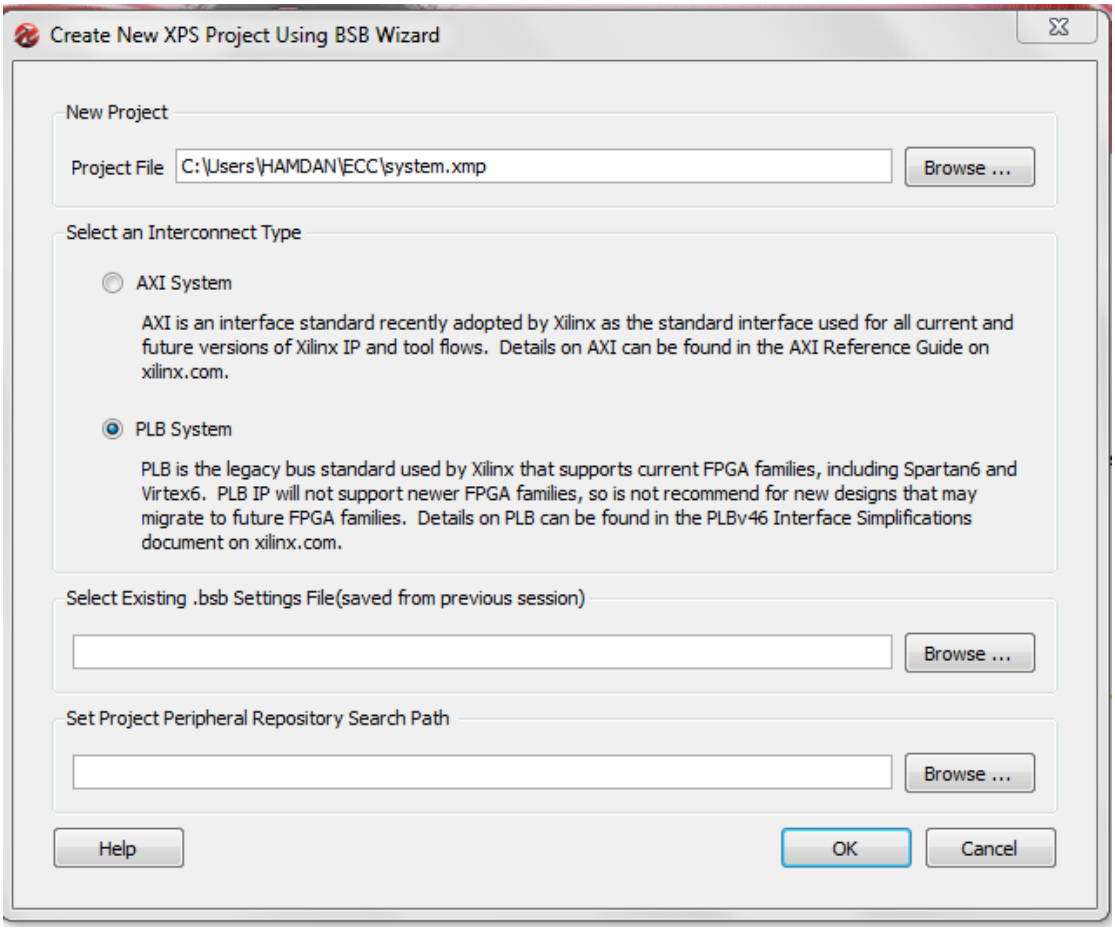


Figure 3.5: création nouveau projet XPS.

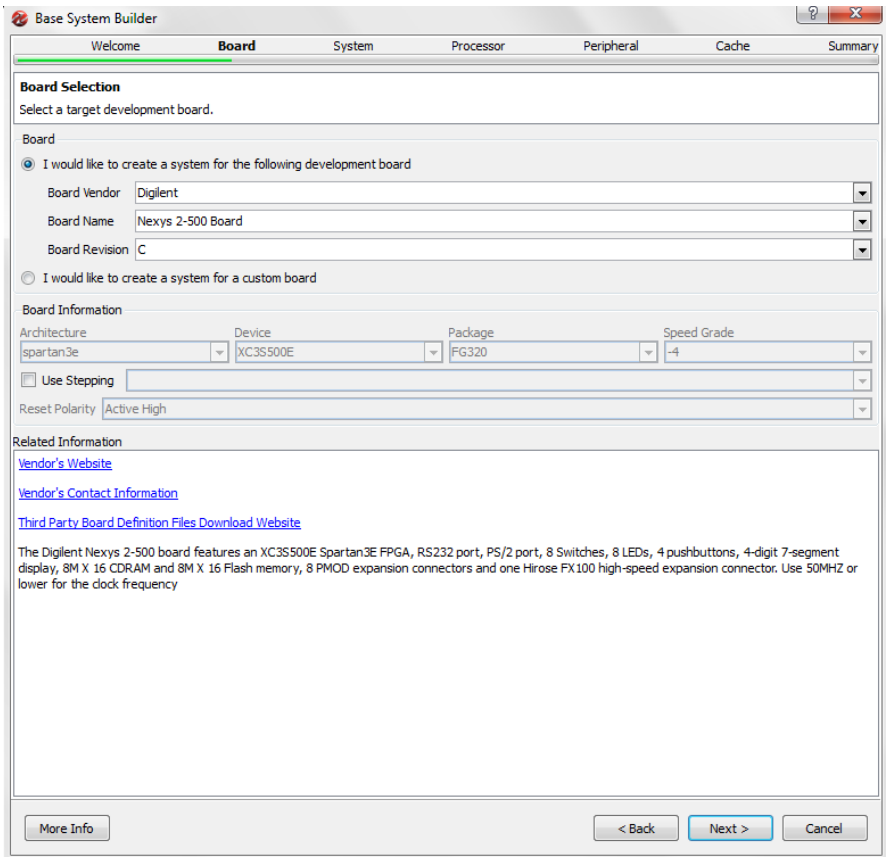


Figure 3.6 : Choix carte FPGA.

Ensuite, on aura la figure 3.6, où on choisit la carte de développement (Xilinx Spartan 3E-500) et on valide ce choix. On obtient la figure 3.7.

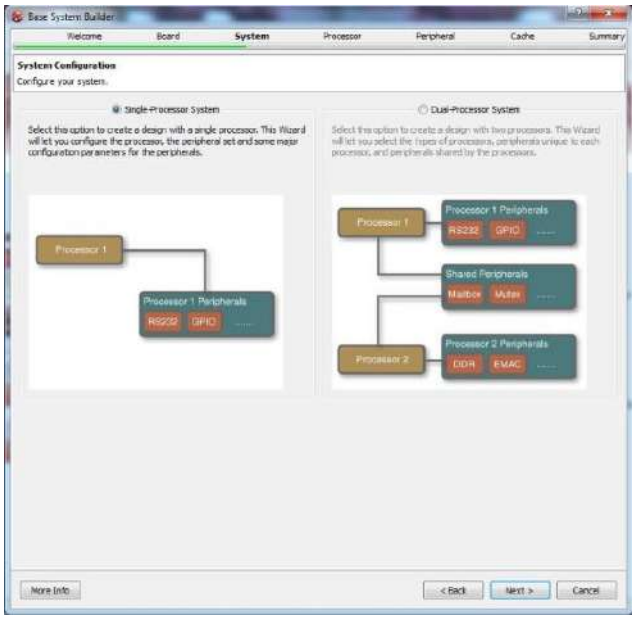


Figure 3.7 : Sélection solution mono processeur.

On choisit une architecture mono processeur. Après validation on passe à la figure 3.8.

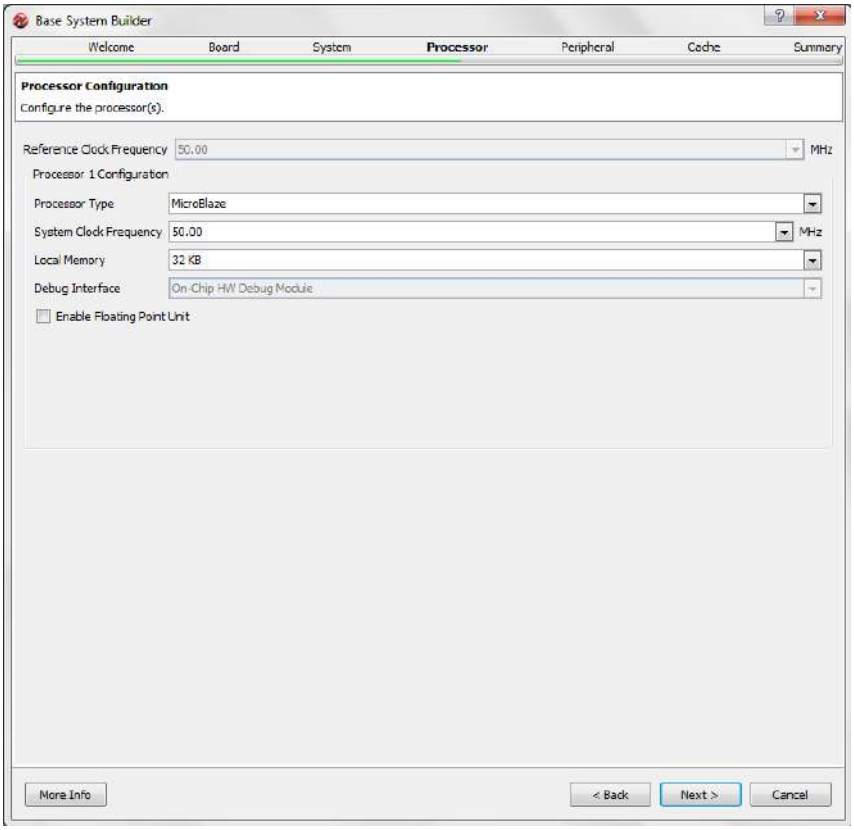


Figure 3.8: sélection taille mémoire et fréquence d'horloge microblaze.

On fixe la taille mémoire interne à 32 koctets avec une fréquence d'horloge du processeur à 50 MHz. Ensuite, on supprime les modules non utilisés comme le montre la figure 3.9.

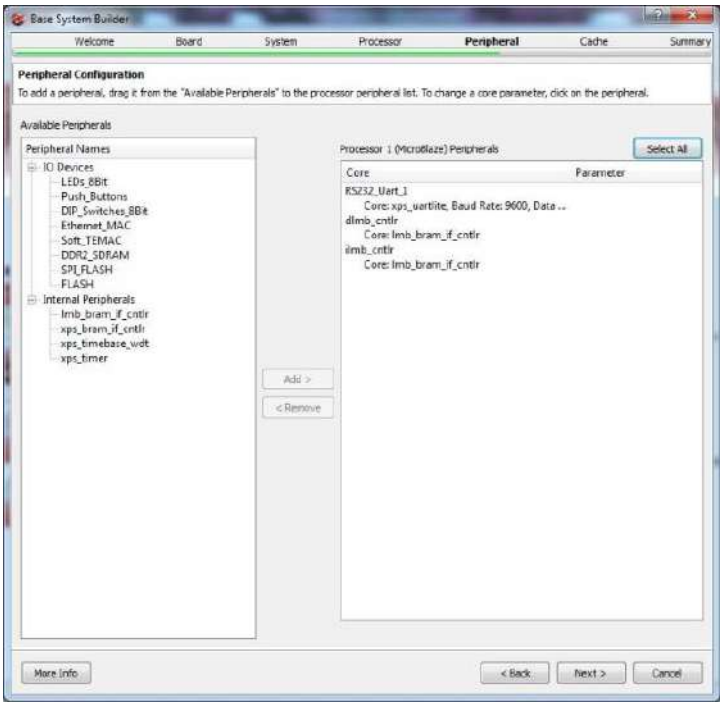


Figure 3.9 : sélection modules utiles.

Après validation, le navigateur des projets de Xilinx Platform Studio charge notre projet comme il est représenté sur la figure 3.10. À ce moment, XPS génére un fichier de contraintes UCF (figure 3.14), un fichier projet (system.xmp), un fichier de spécification de matériel (system.mhs : figure 3.13) et un fichier de configuration logiciel (system.mss) dans notre répertoire.

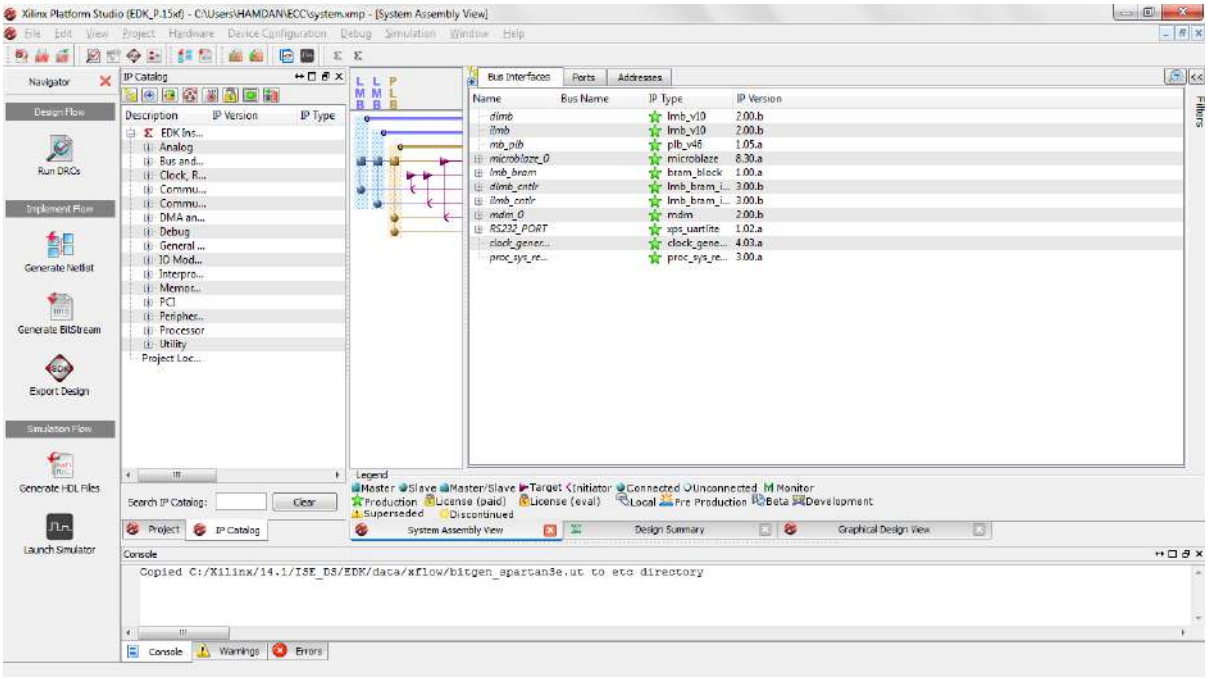


Figure 3.10 : Le navigateur des projets de Xilinx Platform Studio.

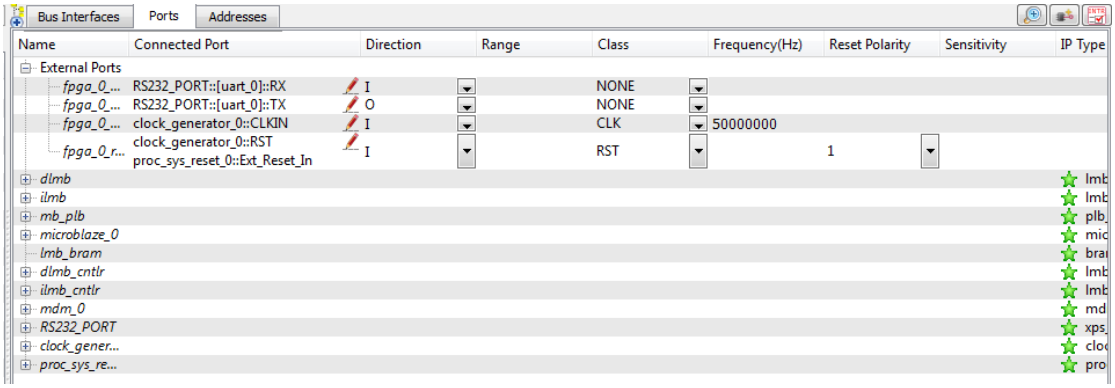


Figure 3.11 : Capture d'écran des ports utilisés.

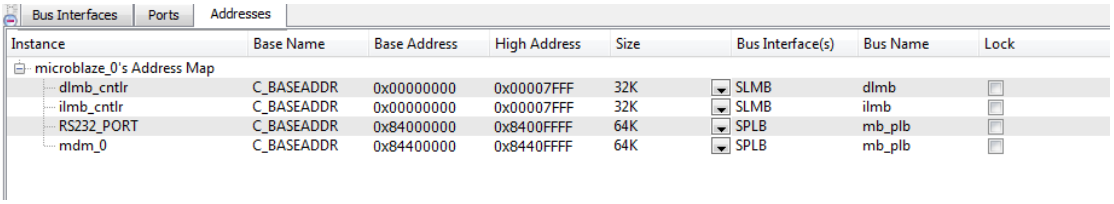


Figure 3.12 : Capture d'écran des adresses réservées aux différents modules.

```
1
2  # ##############################################################################
3  # Created by Base System Builder Wizard for Xilinx EDK 14.1 Build EDK_P.15xf
4  # Tue Apr 18 11:31:00 2017
5  # Target Board:  Digilent Nexys 2-500 Board Rev C
6  # Family:    spartan3e
7  # Device:    XC3S500E
8  # Package:   FG320
9  # Speed Grade:  -4
10 # Processor number: 1
11 # Processor 1: microblaze_0
12 # System clock frequency: 50.0
13 # Debug Interface: On-Chip HW Debug Module
14 # ##############################################################################
15  PARAMETER VERSION = 2.1.0
16
17
18  PORT fpga_0_RS232_PORT_RX_pin = fpga_0_RS232_PORT_RX_pin, DIR = I
19  PORT fpga_0_RS232_PORT_TX_pin = fpga_0_RS232_PORT_TX_pin, DIR = O
20  PORT fpga_0_clk_1_sys_clk_pin = CLK_S, DIR = I, SIGIS = CLK, CLK_FREQ = 50000000
21  PORT fpga_0_rst_1_sys_rst_pin = sys_rst_s, DIR = I, SIGIS = RST, RST_POLARITY = 1
22
23
24 BEGIN microblaze
25  PARAMETER INSTANCE = microblaze_0
26  PARAMETER C_AREA_OPTIMIZED = 1
27  PARAMETER C_USE_BARREL = 1
28  PARAMETER C_DEBUG_ENABLED = 1
29  PARAMETER HW_VER = 8.30.a
30  BUS_INTERFACE DLMB = dlmb
31  BUS_INTERFACE ILMB = ilmb
```

Figure 3.13 : fichier de spécification de matériel MHS.

```
1  #  Nexys 2-500 Board
2  Net fpga_0_RS232_PORT_RX_pin LOC=U6  |  IOSTANDARD = LVCMOS33  |  PULLUP;
3  Net fpga_0_RS232_PORT_TX_pin LOC=P9  |  IOSTANDARD = LVCMOS33  |  DRIVE = 2;
4  Net fpga_0_clk_1_sys_clk_pin TNM_NET = sys_clk_pin;
5  TIMESPEC TS_sys_clk_pin = PERIOD sys_clk_pin 50000 kHz;
6  Net fpga_0_clk_1_sys_clk_pin LOC=B8  |  IOSTANDARD = LVCMOS33;
7  Net fpga_0_rst_1_sys_rst_pin TIG;
8  Net fpga_0_rst_1_sys_rst_pin LOC=B18  |  IOSTANDARD = LVCMOS33;
9
```

Figure 3.14 : fichier de contraintes UCF.

3.4.2. Implémentation du matériel

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

Résultats de l'implémentation

La figure 3.15 représente le schéma RTL du système à base du microblaze implémenté. Le rapport des résultats concernant l'occupation de surface est comme le montre le tableau 3.1.

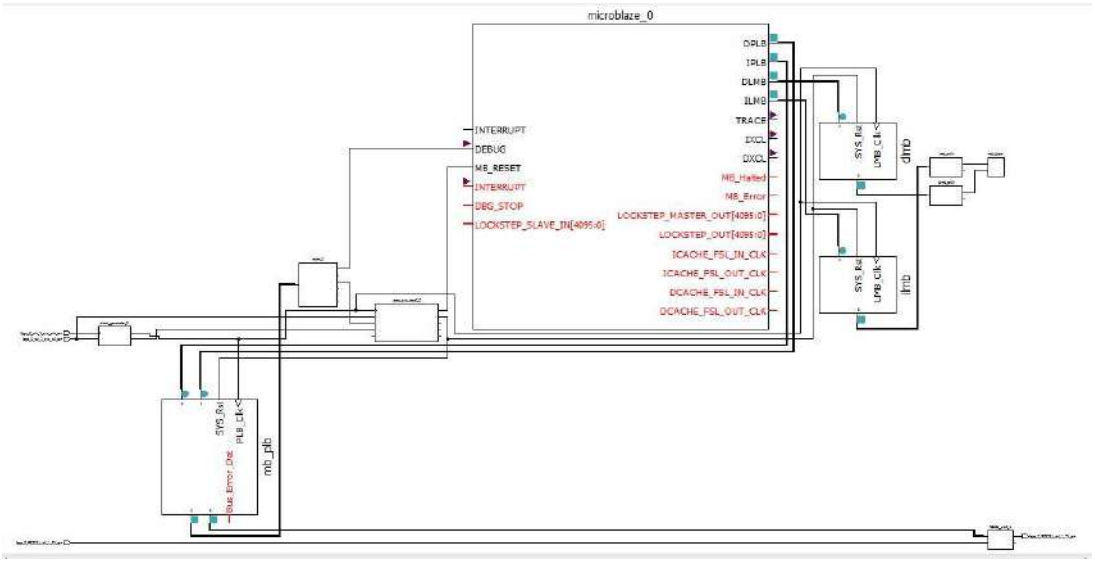


Figure 3.15 : Schéma RTL du système à base du MicroBlaze implémenté.

Tableau 3.1: Rapport des résultats concernant l'occupation de surface.

Device Utilization Summary (actual values)				**[-]**
Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of Slice Flip Flops	1,024	9,312	10%	
Number of 4 input LUTs	2,167	9,312	23%	
Number of occupied Slices	1,500	4,656	32%	
Number of Slices containing only related logic	1,500	1,500	100%	
Number of Slices containing unrelated logic	0	1,500	0%	
Total Number of 4 input LUTs	2,222	9,312	23%	
Number used as logic	1,783			
Number used as a route-thru	55			
Number used for Dual Port RAMs	256			
Number used as Shift registers	128			
Number of bonded IOBs	4	232	1%	
IOB Flip Flops	2			
Number of RAMB16s	16	20	80%	
Number of BUFGMUXs	2	24	8%	
Number of DCMs	1	4	25%	
Number of BSCANs	1	1	100%	
Number of MULT18X18SIOs	3	20	15%	
Average Fanout of Non-Clock Nets	3.79			

3.4.3. Utilisation de l'outil SDK

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

On clique sur l'icône Export Design, l'environnement SDK démarre. On crée un nouveau projet (projet_ecc), avec les fichiers : main.c, add.h, inverse.h, modulo.h, mult.h, type_coord.h et type_message_code.h.

La figure 3.16 montre l'environnement SDK avec notre projet développé.

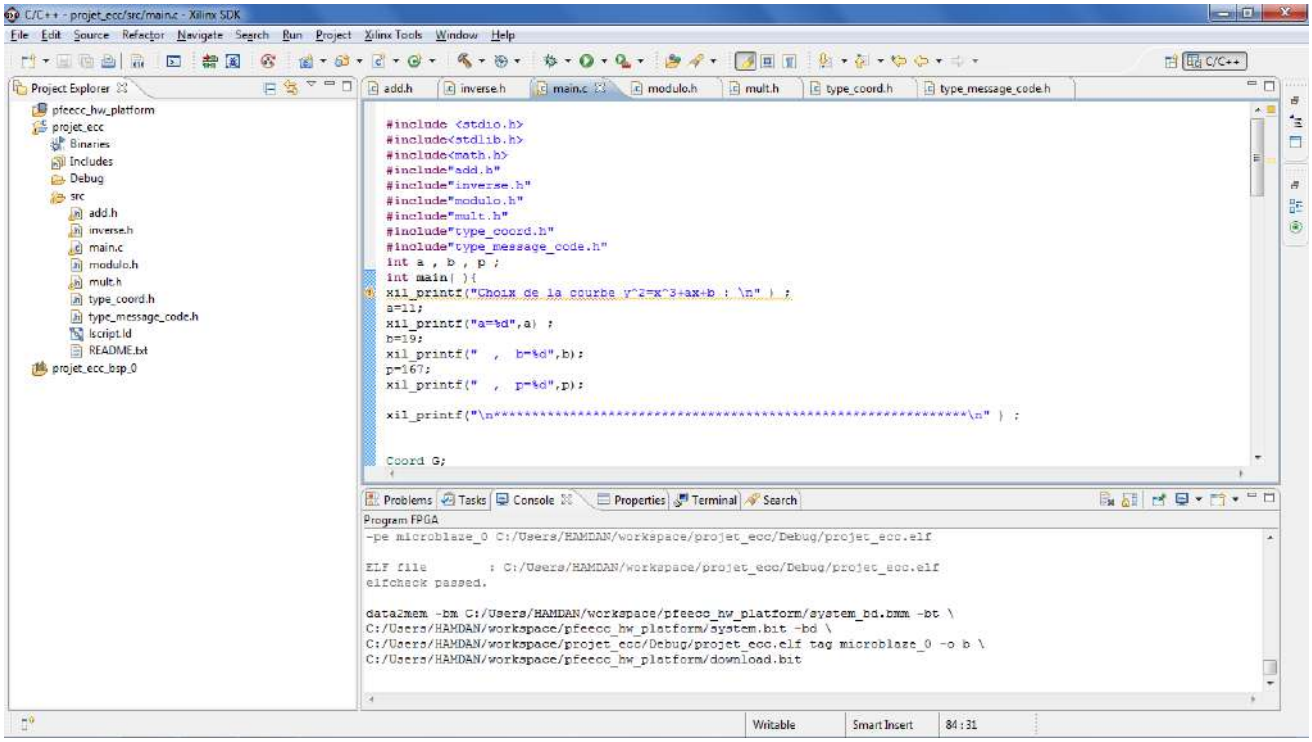


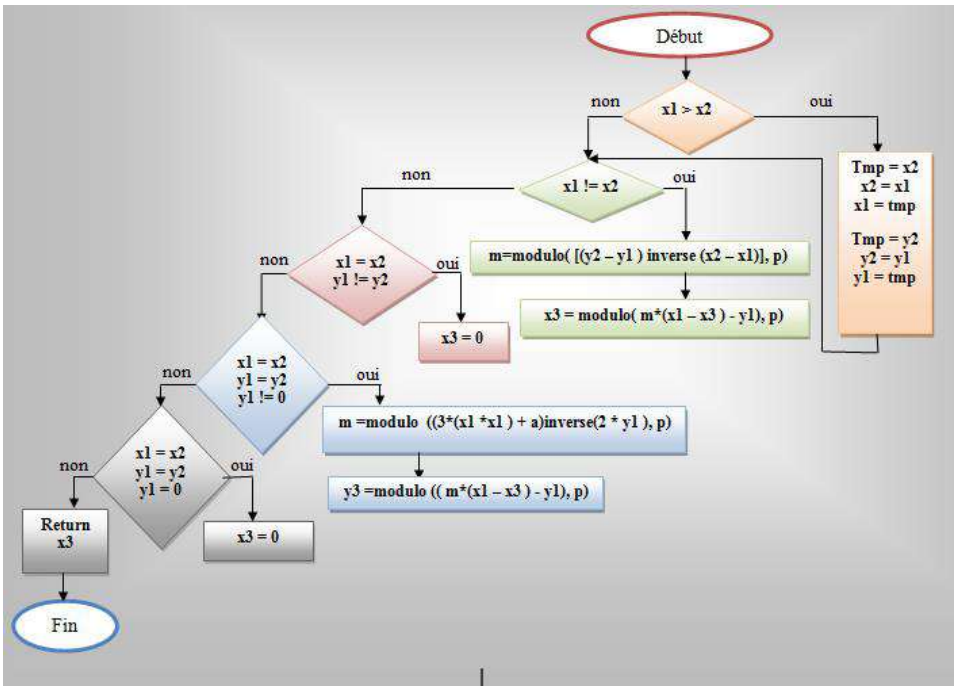
Figure 3.16 : Le navigateur des projets de Software Development Kit.

3.5. Implémentation logicielle de l'algorithme ECC

Notre programme principal exécute 3 taches successives. Il commence par l'échange de clés en utilisant le protocole de Diffie-Hellman, puis il code le message (un point de la courbe) et puis le décode. La partie logicielle développée est basée sur l'implémentation de deux opérations élémentaires : l'addition et la multiplication. Ces deux fonctions nécessitent elles-mêmes la mise en place de deux fonctions auxiliaires, la congruence et l'inversion.

3.5.1. Implémentation des fonctions d'addition, de multiplication, de congruence et d'inversion

a) Fonction d'addition : Cette fonction distingue cinq cas pour l'addition de deux points $P1(x1, y1)$ et $P2(x2, y2)$. Le résultat est un point $P3(x3, y3)$. L'organigramme de la fonction d'addition est montré sur la figure 3.17.



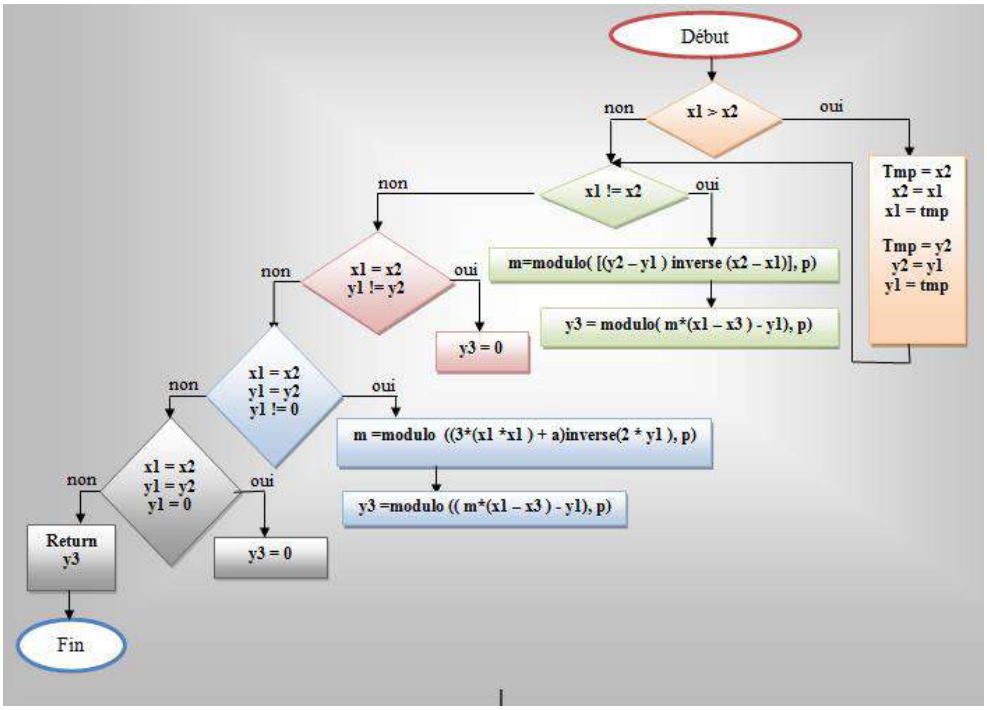


Figure 3.17: Organigrammes de la fonction d'addition « add_x et add_y ».

b) Fonction de multiplication : Cette fonction effectue l'opération d'addition plusieurs fois (une boucle). Elle permet de calculer les multiplications scalaires nécessaires. Cette fonction est représenté dans l'organigramme de la figure 3.18 pour le cas de calcul du produit $k \times G$, où x et y sont les coordonnées du point *G*, *Rx et Ry* sont les coordonnées du point $k \times G$.

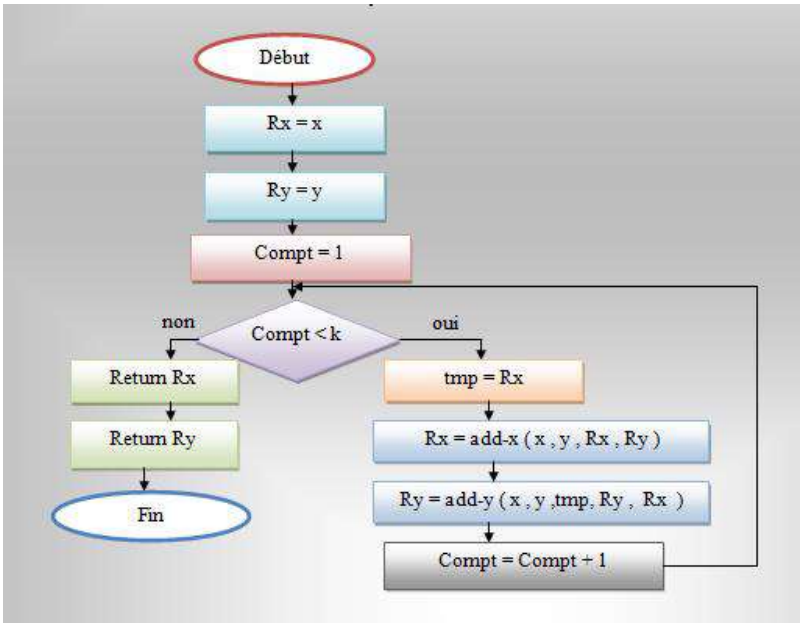


Figure 3.18 : Organigramme de la fonction de multiplication « $mult$ ».

c) Fonction de congruence : Cette fonction prend comme argument d'entrée deux entier x et n et calcul x modulo n. La vérification de la condition sur x (négatif ou positif) est que dans le langage C la fonction modulo (%) donne des résultats non satisfaisants pour les entiers négatifs. L'organigramme de la fonction congruence est montré sur la figure 3.19.

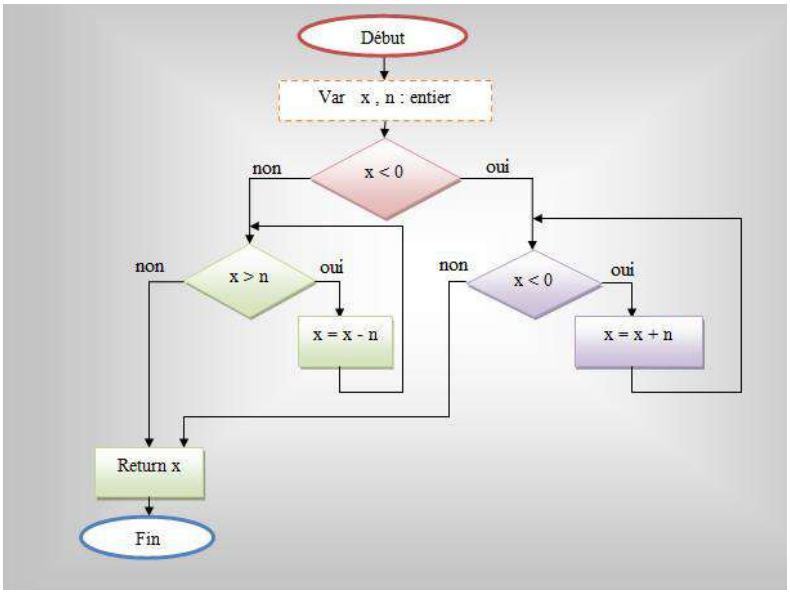


Figure 3.19 : Organigramme de la fonction *congruence.*

d) Fonction d'inversion : Cette fonction permet de réaliser l'opération inverse (a/b), elle n'est pas tout à fait une division classique.

L'organigramme de la fonction de l'inversion est montré sur la figure 3.20.

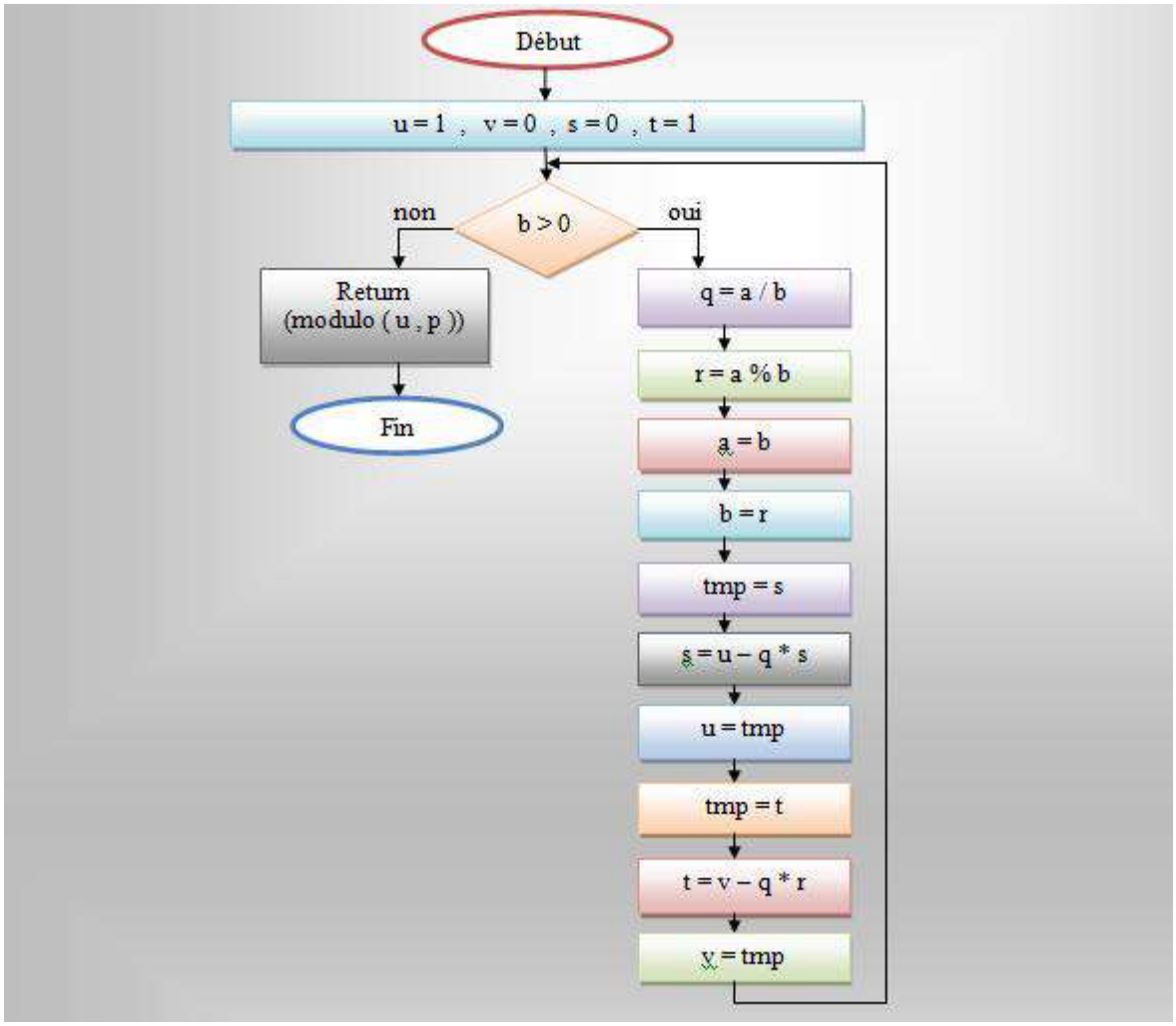


Figure 3.20: Organigramme de la fonction d'inversion « *inverse* ».

3.5.2. Implémentation du protocole d'échange de clés Diffie-Hellman

Nous avons utilisé le protocole Diffie-Hellman pour assurer l'échange de clés. Ce protocole permet de générer la clé privée de chaque entité : n_A et n_B, Les clés publiques sont calculées par la multiplication scalaire : $n_A \times G \; et \; n_B \times G$.

L'organigramme du protocole Diffie-Hellman est montré sur la figure 3.21.

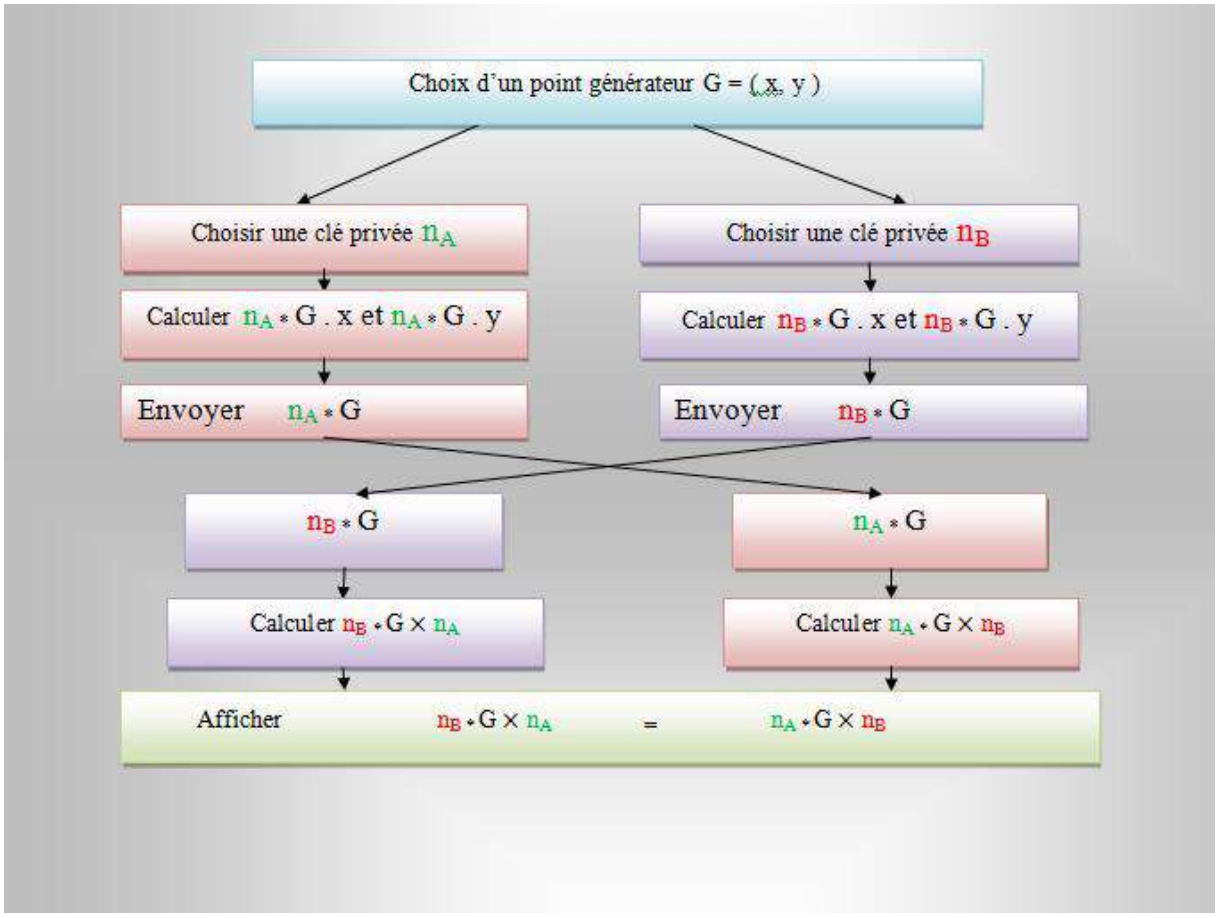


Figure 3.21: Organigramme du protocole Diffie-Hellman implémenté.

3.5.3. Implémentation des fonctions de cryptage/décryptage ECC

3.5.3.1. Cryptage :

La fonction du cryptage permet d'exécuter un chiffrement sur les cordonnées (x,y) du message (P_m). L'exécution de la fonction de cryptage utilise les fonctions $mult\ et\ add$ suivant l'équation (1.3) du premier chapitre. Le résultat est un couple de point $C_m = [(kG, (P_m + kP_B)]$.

3.5.3.2. Décryptage :

La fonction de décryptage permet de restituer les cordonnées $(P_m.x, P_m.y)$ du message clair M à partir des cordonnées de KG et $P_m + kP_B$ du message chiffré. Elle reçoit en entrée le couple C_m. La fonction de décryptage utilise les fonctions $mult\ et\ add$. La fonction $mult$ permet de calculer la multiplication scalaire $n_B \times (KG)$). La fonction add permet de calculer le message $P_m = (P_m + kP_B) - n_B(KG)$.

3.5.4. Programme principal de test de l'algorithme de cryptage/décryptage ECC

Le programme principal appelle la fonction du protocole Diffie-Hellman, la fonction de cryptage et la fonction de décryptage. L'organigramme du programme principal est montré sur la figure 3.22.

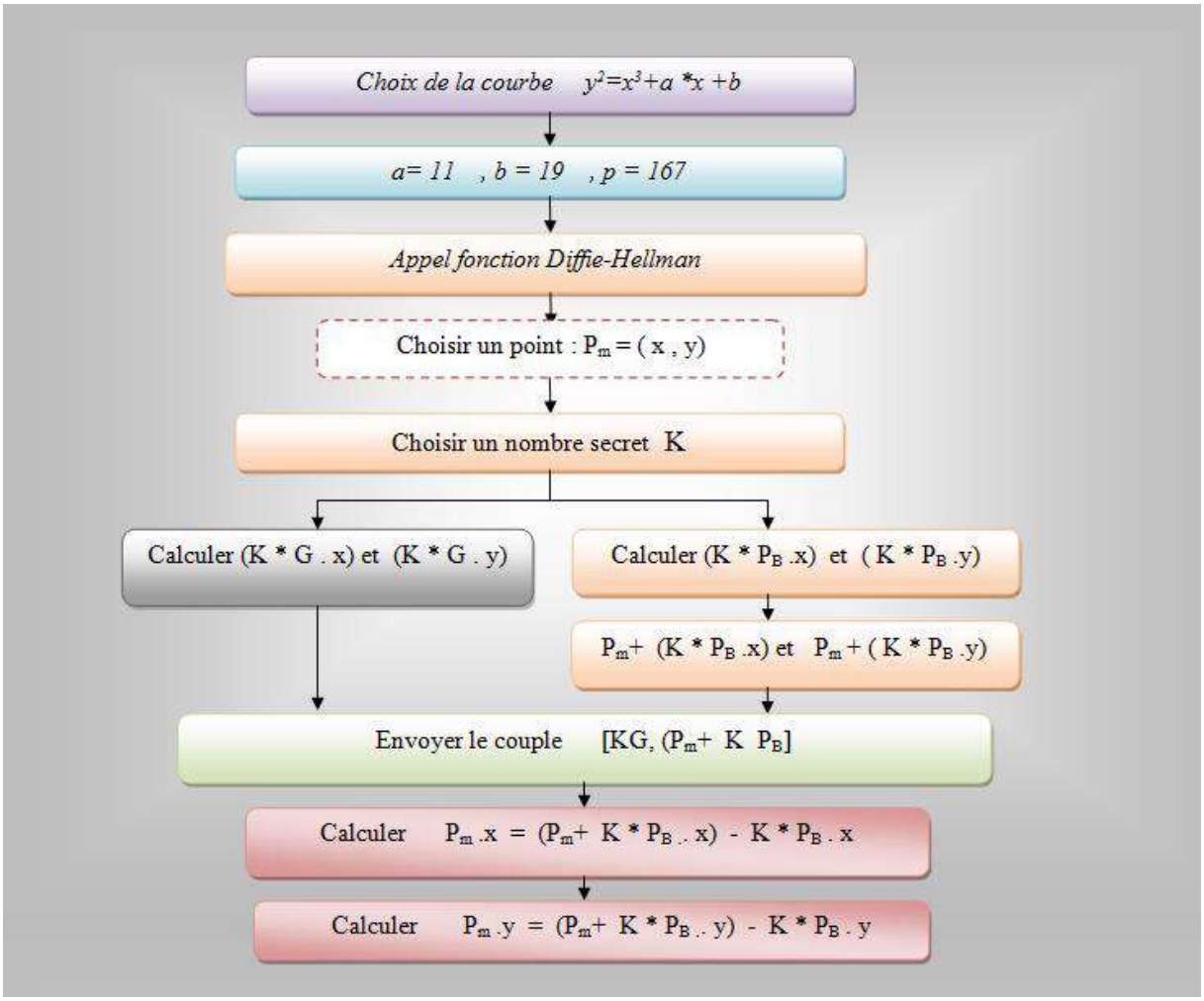


Figure 3.22: Organigramme du programme principal « main ».

3.6. Résultats des tests pratiques sur la carte FPGA

On a utilisé les outils suivants pour faire le test pratique :

- La carte FPGA « Digilent Nexys 2 ».
- Un ordinateur portable.
- Un câble adaptateur liaison série RS232 / USB.
- Un hyper Terminal HERCULES 3.2.5 pour la réception des données en hexadécimal via le port USB émulé port série RS232.

Et pour valider le fonctionnement de l'algorithme ECC implémentée sur le circuit FPGA, nous avons réalisé deux tests pratiques**.**

3.6.1. Résultats du 1er test :

Considérons la courbe d'équation $Y^2 = x^3 + 11\,x + 19$ dans le corps $\mathbb{Z}/167\,\mathbb{Z}$.. Le discriminant de cette courbe est : $\Delta = -(4(11)^3+27(19)^2) = -15071 \neq 0$.

Il s'agit donc d'une courbe utilisable pour la cryptographie sur les courbes elliptiques (même si l'entier 167 est trop faible pour assurer la sécurité des échanges). Les résultats du test pratique sont représentés dans la figure 3.23.

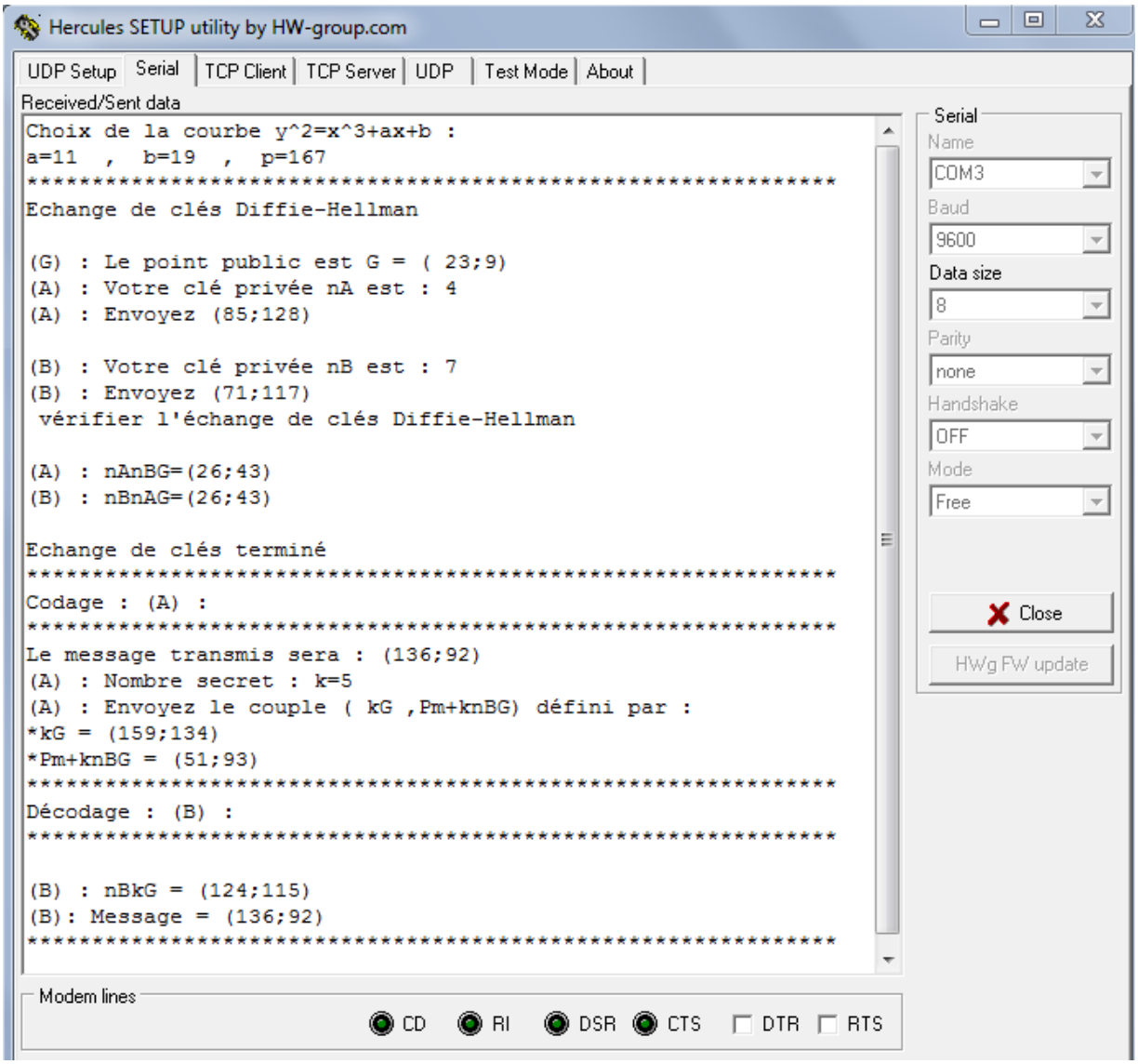


Figure 3.23: Résultat du 1er test pratique sur la carte.

3.6.2. Résultats du 2ème test pratique :

Nous avons réalisé un 2ème test pratique qui utilise la courbe d'équation $Y^2 = x^3 + x + 1$ dans le corps $\mathbb{Z}/23\mathbb{Z}$. On peut noter que les valeurs obtenues ont été validées en les comparants avec les données calculées (**voir l'annexe**).

Les figures 3.24 et 3.25 montrent les résultats de cryptage/décryptage en variant les valeurs des clés privés (n_A , n_B) et le message P_M.

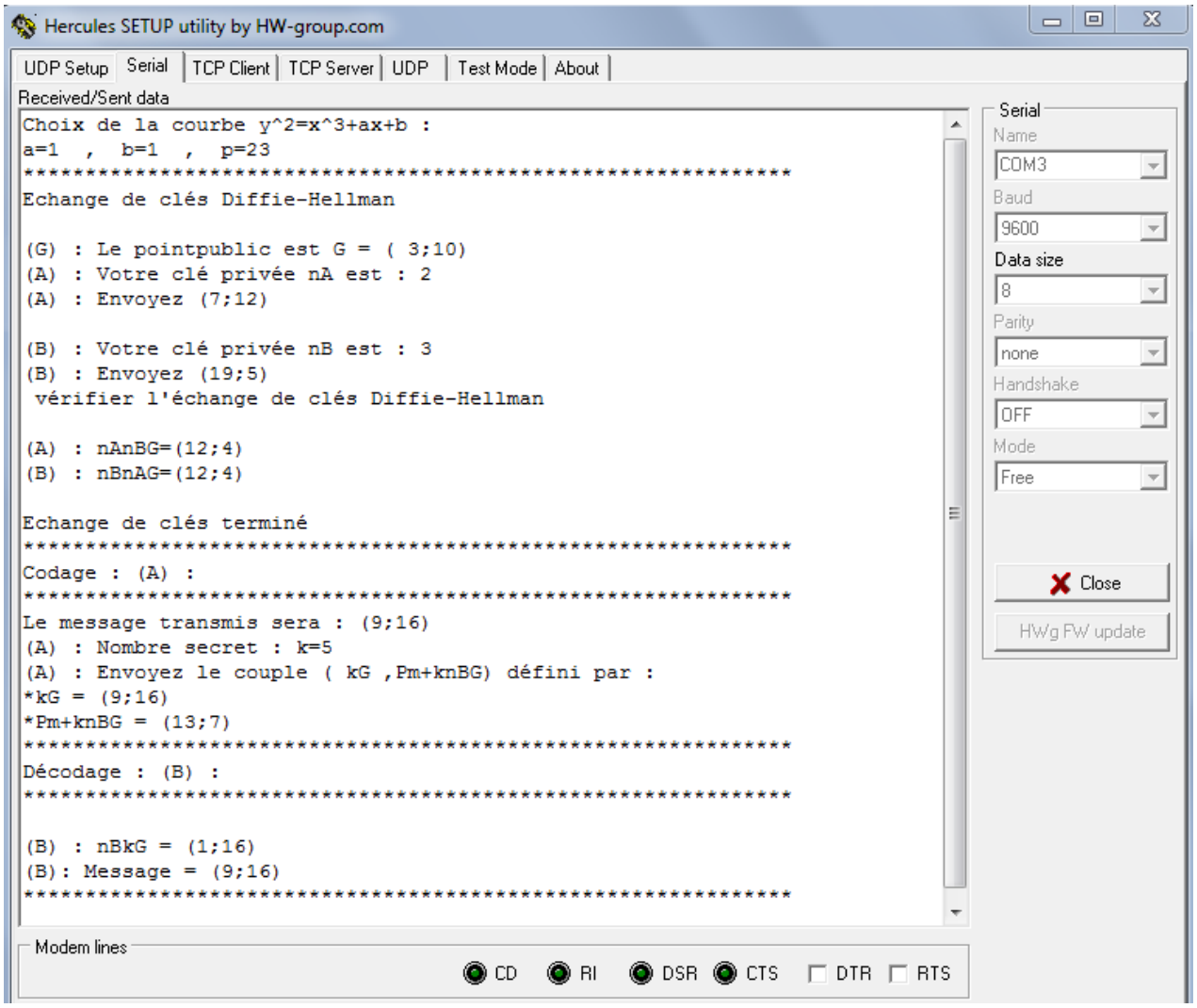


Figure 3.24: Résultat du 2ème test pratique pour $n_A = 2$, $n_B = 3$ et $P_M = (9,16)$.

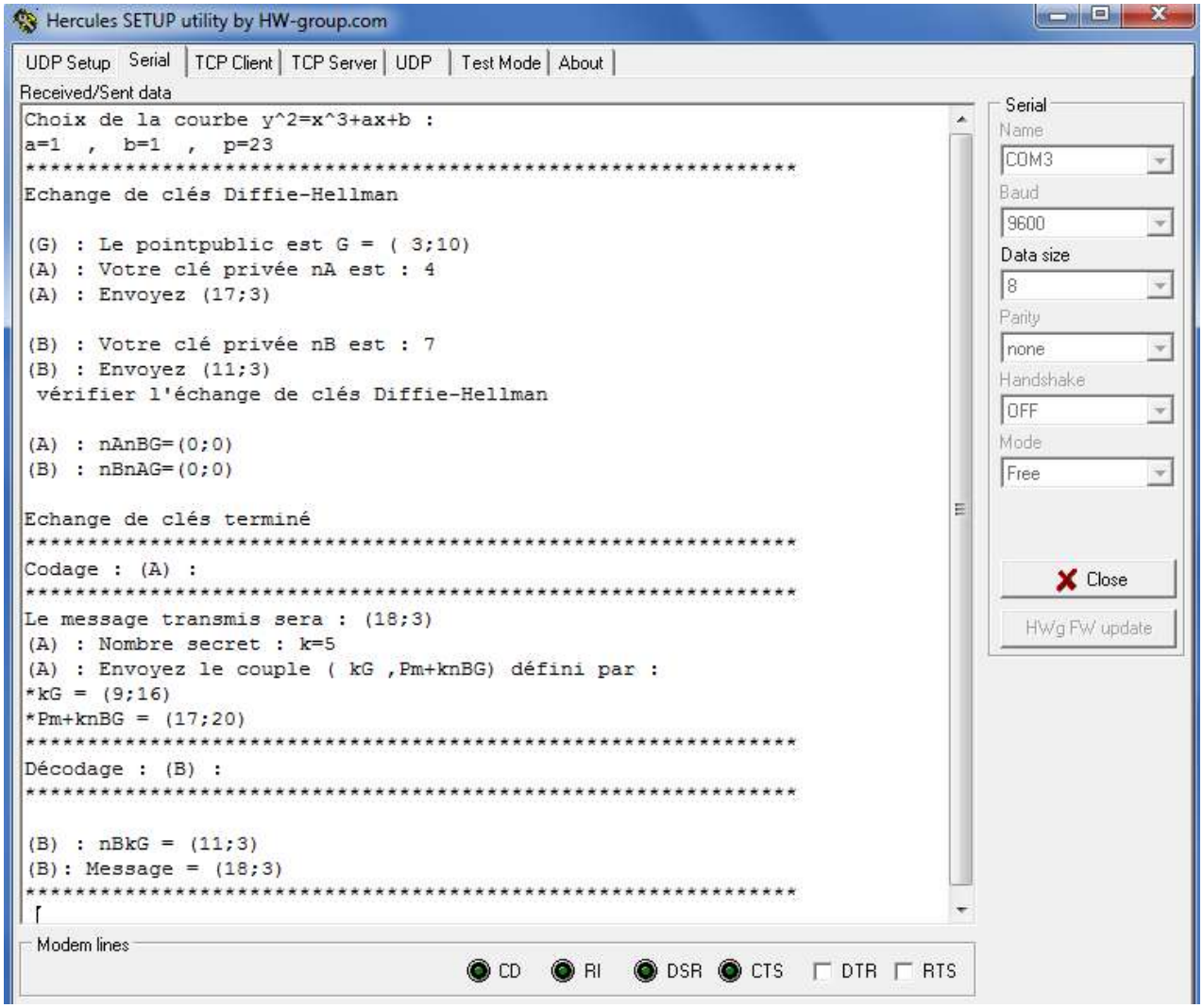


Figure 3.25: Résultat du 2ème test pratique pour $n_A = 4$, $n_B = 7$ et $P_M = (18,3)$.

3.7. Conclusion

Dans ce chapitre nous avons présenté la plateforme du crypto système ECC implémentée. Nous avons présenté par la suite les fonctions décrites en langage C. Ces fonctions sont implémentées sur circuit FPGA et utilisées par le processeur Microblaze pour afin d'exécuter les opérations du chiffrement et du déchiffrement ECC.

Conclusion générale

L'objectif de notre projet consiste à l'implémentation du crypto système à clé publique ECC, sur le processor softcore Microblaze de Xilinx. L'opération cœur de ce crypto système est la multiplication scalaire. Celle-ci est utilisée non seulement pour la génération des clés, mais aussi lors du chiffrement et du déchiffrement des données.

De ce fait, nous avons entamé l'étude théorique en deux parties, à savoir, l'étude des protocoles cryptographiques, en particulier le crypto-système à clé publique ECC et l'étude des systèmes embarqués à base des circuits FPGA.

Les opérations arithmétiques de base, pour effectuer le chiffrement et le déchiffrement sont l'addition, la multiplication, la congruence et l'inverse, Ces opérations ont été implémentées d'une manière logicielle et exécutées par le processeur Microblaze implémenté sur un circuit FPGA de type Spartan 3E 500.

Dans ce travail, la méthodologie utilisée pour la réalisation du crypto-système ECC est répartie en deux étapes :

- Développement de la partie matérielle du crypto-système embarqué, par l'utilisation de l'outil XPS.
- Développement de la partie logicielle en langage C, par l'utilisation de l'outil SDK.

Ce projet était bénéfique pour nous dans plusieurs sens, ça nous a permis d'acquérir une expérience dans divers domaines, à savoir les systèmes embarqués, les outils d'implémentations de Xilinx et la cryptographie.

Cependant, des limites existent dans notre réalisation. La première est qu'on a une condition sur le coefficient b de la courbe elliptique qui doit être non nul afin de s'assurer que le point de coordonnées (0,0) peut être utilisé comme le point à l'infini. La deuxième est que le type des variables utilisées pour la mise en place est de type entier qui limite la taille des coefficients et des clés que l'on utilise. Il faudrait le remplacer par le type double et adapter toutes les fonctions.

Références

[1] Yanbo Shou, "Cryptographie sur les courbes elliptiques et tolérance aux pannes dans les réseaux de capteurs", Thèse de Doctorat, Faculté d'Electronique et d'Informatique, Université de Franche-Comté, 2014.

[2] William Stallings, "Cryptography and Network Security", Principals and Practice, Pearson Sixth Edition, 2014.

[3] Paar Christof, Pelzl Jan, "Understanding Cryptography: A Textbook for Students and Practitioners", Springer, 2010.

[4] W.Diffie and M.E.Hellman, "New Directions in Cryptography", IEEE Transactions on Information Theory, vol.IT-22, No.6, November 1976.

[5] Hagler Michael, "Courbes elliptiques et cryptographie", http://math.univ-bpclermont.fr/~rebolledo/page-fichiers/projetMichael.pdf , consulté le 05/06/2017.

[6] Hamzi Imane " Contribution à la sécurité des communications d'un réseau WSN appliqué dans un système de contrôle du trafic routier", mémoire de Master, Faculté des Sciences et de la Technologie, Université Dr. Yahia Farès de médéa, 2015.

[7] Issad Mohamed "Crypto système embarqué sur FPGA", Thèse de Doctorat, Faculté d'Electronique et d'Informatique, Université des sciences et de la Technologie Houari Boumediene, 2015.

[8] Digilent Nexys2 Board Reference Manual, 2011.

Annexe

Dans cette annexe, nous présentons un exemple de calcul d'une courbe elliptique [2] pour un **corps réduit** $\mathbb{Z}/p\mathbb{Z}$, puis on applique le principe de cryptage/décryptage sur un point de la courbe.

a) Exemple de calcul des points de la courbe elliptique :

Nous choisissons un nombre premier p = 23 et a = 1 ; b = 1. Au début il faut vérifier si :

$$4\,a^3 + 27\,b^2 \bmod p = 4 * 1^3 + 27 * 1^2 \bmod 23$$

$$4\,a^3 + 27\,b^2 \bmod p = 31 \bmod 23$$

$$4\,a^3 + 27\,b^2 \bmod p = 8 \neq 0$$

Nous déterminons ensuite, les résidus quadratiques Q_{23} à partir de l'ensemble réduit de résidus $\mathbb{Z}_{23} = \{1\,,2\,,3\,,\ldots\ldots,21\,,22\}$

y^2 mod p	$(p - y)^2$ mod p	=
1^2 mod 23	22^2 mod 23	1
2^2 mod 23	21^2 mod 23	4
3^2 mod 23	20^2 mod 23	9
4^2 mod 23	19^2 mod 23	16
5^2 mod 23	18^2 mod 23	2
6^2 mod 23	17^2 mod 23	13
7^2 mod 23	16^2 mod 23	3
8^2 mod 23	15^2 mod 23	18
9^2 mod 23	14^2 mod 23	12
10^2 mod 23	13^2 mod 23	8
11^2 mod 23	12^2 mod 23	6

Donc, l'ensemble de $\frac{p-1}{2} = 11$ résidus quadratiques est :

$$Q_{23} = \{1\,,2\,,3\,,4\,,6\,,8\,,9\,,12\,,13\,,16\,,18\}.$$

Maintenant, pour $0 \leq x < 23$ nous calculons $y^2 = x^3 + x + 1 \; mod \; 23$ et nous déterminons si le y^2 est dans l'ensemble des résidus quadratiques Q_{23} (Tableau 1).

Tableau 1

x	0	1	2	3	4	5	6	7	8	9	10	11
y^2	1	3	11	8	0	16	16	6	15	3	22	9
$y^2 \in Q_{23}$	Oui	Oui	Non	Oui	Non	oui	Oui	oui	non	Oui	non	oui
y_1	1	7		10	0	4	4	11		7		3
y_2	22	16		13	0	19	19	12		16		20

x	12	13	14	15	16	17	18	19	20	21	22
y^2	16	3	22	10	19	9	9	2	17	14	22
$y^2 \in Q_{23}$	oui	Oui	Non	Non	Non	oui	Oui	Oui	non	Non	non
y_1	4	7				3	3	5			
y_2	19	16				20	20	18			

Donc, le groupe elliptique $E_{23}(1, 1)$ inclut les points suivant :

$$E_{23}(1,1) = \left\{ \begin{array}{c} (0,1)\ (0,22)\ (1,7)\ (1,16)\ (3,10)\ (3,13)\ (4,0) \\ (5,4)\ (5,19)\ (6,4)\ (6,19)\ (7,11)\ (7,12)\ (9,7) \\ (9,16)\ (11,3)\ (11,20)\ (12,4)\ (12,19)\ (13,7) \\ (13,16)\ (17,3)\ (17,20)\ (18,3)\ (18,20)\ (19,5)\ (19,18) \end{array} \right\}$$

Le point supplémentaire (4,0), correspondant au point double ayant y = 0. La figure1 représente un diagramme spatial des points constituants le groupe elliptique $E_{23}(1, 1)$.

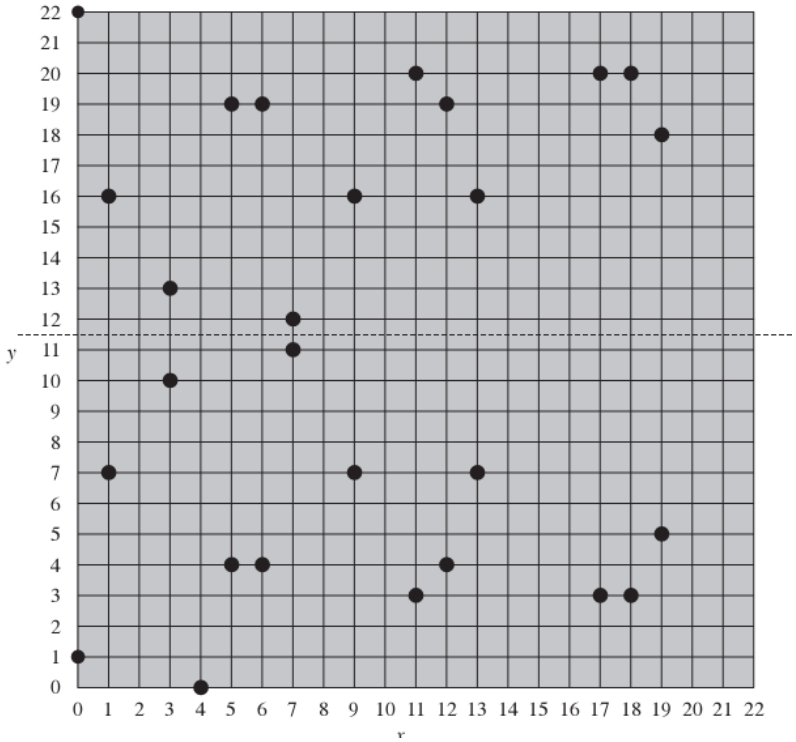


Figure 1 : diagramme spatial du groupe elliptique $E_{23}(1,1)$

b) Exemple de l'opération de multiplication scalaire au-dessus des groupes elliptiques :

Soit un point $P = (3,10) \in E_{23}(1,1)$. Alors $2P$ est égal à :

$$2P = P + P = (x_1, y_1) + (x_1, y_1)$$

Comme $P = Q$ et $x_2 = x_1$, les valeurs de λ, x_3 et $y_3$sont données par :

$$\lambda = \frac{3x_1^2 + a}{2y_1}\ mod\ p = \frac{3 * (3)^2 + 1}{2 * 10} mod\ 23 = \frac{5}{20}\ mod\ 23 = 4^{-1} mod\ 23 = 6$$

$$x_3 = \lambda^2 - x_1 - x_2 mod\ p = 6^2 - 3 - 3\ mod\ 23 = 30\ mod\ 23 = 7$$

$$y_3 = \lambda(x_1 - x_3) - y_1 mod\ p = 6 * (3 - 7) - 10\ mod\ 23 = -34\ mod\ 23 = 12$$

Donc $2P = (x_3, y_3) = (7, 12)$.

La multiplication k P est obtenue en répétant k fois l'opération d'addition. Les résultats des multiplications pour différentes valeurs de k sont récapitulés dans le tableau 2.

Tableau 2

k	λ $\lambda = \frac{y_2 - y_1}{x_2 - x_1}$ (if $P \neq Q$) or $\lambda = \frac{3x_1^2 + a}{2y_1}$ if $P = Q$	x_3 $\lambda^2 - x_1 - x_2 \bmod 23$	y_3 $\lambda(x_1 - x_3) - y_1 \bmod 23$	kP (x_3, y_3)
1				(3 ,10)
2	6	7	12	(7 ,12)
3	12	19	5	(19,5)
4	4	17	3	(17,3)
5	11	9	19	(9,16)
6	1	12	4	(12,4)
7	7	11	3	(11,3)
8	2	13	16	(13,16)
9	19	0	1	(0,1)
10	3	6	4	(6,4)
11	21	18	20	(18,20)
12	16	5	4	(5,4)
13	20	1	7	(1,7)
14	13	4	0	(4,0)
15	13	1	16	(1,16)
16	20	5	19	(5,19)
17	16	18	3	(18,3)
18	21	6	19	(6,19)
19	3	0	22	(0,22)
20	19	13	7	(13,7)
21	2	11	20	(11,20)
22	7	12	19	(12,19)
23	1	9	7	(9,7)
24	11	17	20	(17,20)
25	4	19	18	(19,18)
26	12	7	11	(7,11)
27	6	3	13	(3,13)
28	--	--	--	(0,0)
29	--	--	--	(3,10)

c) Exemple de cryptage et décryptage sur les courbes elliptiques :

Alice veut envoyer le message codé par le point $P_M = (9,16) \in E_{23}(1,1)$. On suppose que le point générateur $G = (3,10)$ a été choisi publiquement. Alice doit utiliser la clé publique de Bob pour crypter. Si la clé secrète de Bob est $n_B = 3$, alors sa clé publique sera : $P_B = n_B G = 3\,(3,10) = (19,5)$.

Ensuite, Alice choisit un nombre aléatoire $k = 5$ et utilise la clé publique de Bob : $P_B = (19,5)$. pour encrypter le message. Donc, elle envoie la paire des points :

$P_c = [kG, P_M + kP_B] = [5 * (3,10)\,, (9,16) + 5 * (19,5)]$

$P_c = [(9,16), (9,16) + (1,16)]$

P_c= [(9,16),(13,7)]

Lors de la réception de cette paire de points, le $P_c = [(9,16), (13,7)]$, Bob emploie sa clé privée, $n_B = 3$ pour calculer le point contenant le message clair, le $P_M$comme suit :

$(P_M + kP_B) - [n_B(kG)] = (13,7) - [3 * (9,16)]$

$(P_M + kP_B) - [n_B(kG)] = (13,7) - [(1,16)]$

$(P_M + kP_B) - [n_B(kG)] = (13,7) + [(1,-16)]$

$(P_M + kP_B) - [n_B(kG)] = (13,7) + [(1,7)]$

$(P_M + kP_B) - [n_B(kG)] = (9,16).$

Donc, le point calculé $P_M = (9,16)$ donne le message clair codé par ce point.