République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

ÉCOLE NATIONALE POLYTECHNIQUE

DÉPARTEMENT D'ÉLECTRONIQUE

Projet de fin d'études

En vue de l'obtention du diplôme de Master en Électronique

Thème :

Étude et implémentation sur circuit reconfigurable de l'algorithme de Berlekamp-Massey pour le décodage de Reed-Solomon

Réalisé par :

Mr **KACED Karim**

Soutenu publiquement le **25/06/2013** devant le jury composé de :

R. SADOUN	Docteur	ENP	**Président**
M. TAGHI	Chargé de Cours	ENP	**Promoteur**
L. ABDELOUEL	Chargé de Cours	ENP	**Co-Promoteur**
M. MAMERI	Chargé de Cours	ENP	**Examinateur**

Promotion : Juin 2013

ملخص

يندرج هذا العمل في إطار دراسة الشفرات المصححة للأخطاء. اهتممنا بالخصوص بالشفرات ريد - سولومون المستخدمة في الاتصالات اللاسلكية.

بعد القيام بدراسة نظرية حول هذه الشفرات، نقترح تصميما للهيكل بيرلكامب ماسي من اجل حل المعادلة المفتاحية لهذه الشفرات. تمت كتابة البرامج باللغة VHDL فيما تم انجاز الحوصلة و التثبيت لهذا الهيكل للشفرة (15,9) و (255,239) بواسطة الأداة ISE ل Xilinx على FPGA.

الكلمات المفتاحية : شفرة ريد- سولومون ، خوارزمية بيرلكامب ماسي، تشفير القناة، FPGA ، الشفرات المصححة للأخطاء.

Résumé

Ce travail s'inscrit dans le cadre de l'étude des codes correcteurs d'erreurs. Nous nous intéressons tout particulièrement aux codes de Reed-Solomon utilisés dans les communications sans fils.

Après une étude théorique de ces codes, on propose une architecture de bloc de Berlekamp-Massey pour la résolution de l'équation clé de ces codes. Les codes de description sont écrits en VHDL, la synthèse est l'implémentation de ce bloc pour les codes RS(15,9) et RS(255,239) est réalisé à l'aide de l'outil ISE de Xilinx sur carte FPGA.

Mots clés : codes de Reed-Solomon, algorithme de Berlekamp-Massey, codage canal, FPGA, code correcteurs d'erreurs.

Abstract

This work lies within the scope of the studies of the errors correction codes. We are interested particularly to studies the Reed-Solomon codes used in the wireless communication.

After theoretical study of these codes, we propose a design of the block of Berlekamp-Massey for solving the key equation for these codes. The codes of description are written in VHDL, the synthesis and the implementation of this block for RS(15,9) and RS(255,239) codes is performed using the tool ISE of Xilinx on FPGA board.

Key words: Reed-Solomon codes, Berlekamp-Massey algorithm, Channel coding, FPGA, errors correction codes.

Remerciements

Je remercie mes parents qui m'ont beaucoup encouragé le long de ce projet.

À l'issu de mes études faites au sein de l'École Nationale Polytechnique, je voudrais rendre un hommage tout particulier à Mes promoteurs : Mr. ***M. Taghi*** *et Mr.* ***L. Abdelouel*** *qui m'ont honoré en acceptant de m'encadrer.*

Je remercie également Mr ***R. Sadoun****, d'avoir accepté de présider le jury et Mr* ***M. Mameri****, d'avoir accepté d'examiner ce travail.*

Je tiens également à exprimer ma profonde gratitude à tous les enseignants de l'École Nationale Polytechnique qui ont contribué à ma formation.

Dédicaces

À mes très chers parents,

À mes sœurs et frères,

À mes grands parents,

À mes tantes et oncles,

À tous qui m’ont soutenu le long de mon parcours,

À tous mes amis,

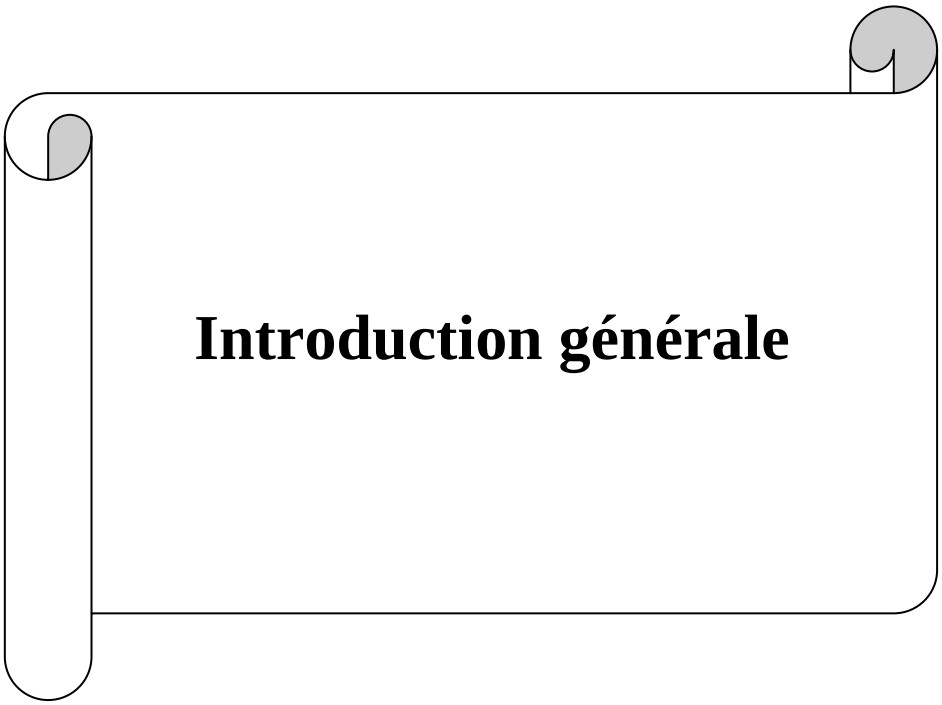
Je dédie ce travail.

Karim

Table des matières

Liste des figures

Liste des tableaux



Introduction générale

Introduction

L'histoire des transmissions a été marquée par plusieurs évolutions, des pigeons voyageurs aux communications sans fils. Actuellement, nous assistons à une forte augmentation des besoins en termes de communication numérique. La communication à distance (téléphone, internet, satellite, . . .) entre machine et usagers nécessite les canaux de transmission pour l'acheminement des informations. Les canaux utilisés sans en général loin d'être parfaits, cela peut entrainer une modification du message émis. L'imprévisibilité du message émis par la source impose alors au récepteur l'utilisation de techniques lui permettant de vérifier à la fois l'exactitude et la certitude de l'information reçue.

Problématique

Sans un souci de fiabilités, tous les efforts d'amélioration dés débits de transmission seraient vains car cela impliquerait que certaines données doivent être retransmises. C'est dans la course au débit que les codes correcteurs d'erreurs entrent en jeu.

Un code correcteur d'erreurs est un code qui permet de corriger une ou plusieurs erreurs en ajoutant au mot d'information des symboles de contrôle. Plusieurs codes existent mais dans ce projet de fin d'´etude nous intéressons aux codes de Reed-Solomon utilisés dans le domaine des communications sans fils pour le meilleur compromis efficacité et complexité qu'ils présentent.

Objectif du projet

Ce travail s'inscrit dans le cadre des activités du Laboratoire des Dispositifs de Communications et de Conversions Photovoltaïques (LDCCP) de l'ENP. Il vise à explorer les possibilités qu'offrent les circuits programmables (FPGA) pour l'implémentation de bloc de Berlekamp-Massey pour la résolution de l'équation clé des codes de Reed-Solomon.

Plus spécifiquement, les objectifs poursuivis pour ce projet de fin d'étude sont :

- Étude des principes des codes de Reed-Solomon,
- Proposition des architectures optimisées pour l'implémentation de bloc de Berlekamp-Massey pour le décodeur de Reed-Solomon.
- Simulation de bloc de Berlekamp-Massey pour les codes RS(15,9), RS(255,239) sous ISE de Xilinx,
- Implémentation de bloc de Berlekamp-Massey pour les codes RS(15,9) et RS(255,239) sur circuit reconfigurable « FPGA».

Organisation du mémoire

Ce mémoire est divisé en deux chapitres :

- Dans le premier, On explique la théorie de codage et de décodage des codes de Reed-Solomon et nous donnons les principaux algorithmes basés sur les registres à décalage de Berlekamp-Massey pour la résolution de l'équation clé.
- Dans le deuxième chapitre, on expose l'architecture proposée pour le bloc de Berlekamp-Massey en choisissant l'algorithme (EIBM). On présente ainsi les résultats de l'implémentation de ce bloc sur les circuits reconfigurables FPGA en donnant les ressources hardware utilisées et les performances temporelles de chaque
- Enfin, en conclusion, on rappelle les objectifs atteints et les perspectives.

Chapitre I

Les codes de Reed-Solomon

1.1. Introduction

Les codes de Reed Solomon constituent une famille de codes d'une importance exceptionnelle, tant du point de vue de la théorie que des applications. Récemment, ces codes ont été largement utilisés dans de nombreuses applications telles que la communication par satellite, la diffusion de vidéo numérique (digital video broadcast) et la communication mobile sans fil. Ces codes, qui fonctionnent en bloc, sont basés mathématiquement sur les champs finis de Galois. Les codes RS permettent de corriger les erreurs et les effacements grâce à des symboles de contrôle ajoutés après l'information.

On commencera ce chapitre par la présentation des corps de Galois. On entamera ensuite la théorie de codage des codes de Reed-Solomon. À la fin, on détaillera leur théorie de décodage des codes de Reed-Solomon basée sur les registres de décalage de Berlekamp-Massey.

1.2. Corps de Galois

Les « champs de Galois » font partie d'une branche particulière des mathématiques qui modélise les fonctions du monde numérique. Ils sont très utilisés dans la cryptographie ainsi que pour la reconstruction des données.

Il y a deux types de corps, les corps finis et les champs infinis. Les « corps de Galois » finis sont des ensembles d'éléments fermés sur eux-mêmes. L'addition et la multiplication de deux éléments du corps donnent toujours un élément du corps fini.

1.2.1. Éléments des champs de Galois

Un « champ de Galois » consiste en un ensemble de nombres, ces nombres sont constitués à l'aide de l'élément base α comme suit :

$$0, \alpha, \alpha^2, \dots, \alpha^{n-1}$$

En prenant $n = 2^m - 1$, on forme un ensemble de 2^m éléments. Le champ est alors noté $GF(2^m)$.

$GF(2^m)$ est formé à partir du champ de base $GF(2)$ et contiendra des multiples des éléments simples de $GF(2)$.

En additionnant les puissances de α, chaque élément du champ peut être représenté par une expression polynomiale du type :

$$\alpha^{m-1}x^{m-1} + \alpha^{m-2}x^{m-2} + \cdots + \alpha x + \alpha^0$$

Avec : $\alpha^{m-1}, \ldots, \alpha^0$: éléments bases du *GF* (2) (valeurs : 0,1)

1.2.2. Opérations algébriques

- **Addition**

L'addition dans un champ fini *GF* (2) correspond à faire une addition modulo 2, donc l'addition de tous les éléments d'un « champ de Galois » dérivés du champ de base sera une addition modulo 2.

Pour effectuer une addition sur le corps *GF* (2^m) , il faut voir que l'on additionne en fait deux vecteurs, et donc que l'on effectue l'addition composante par composante.

- **Soustraction**

Une soustraction dans *GF* (2^m) correspond à faire une addition dans le même corps.

- **Multiplication**

La multiplication dans le « champ de Galois » peut être réalisée suivant deux techniques : la première consiste à utiliser une table de vérité, la deuxième consiste à faire la multiplication entre deux polynômes et ensuite de normaliser le résultat par rapport au polynôme primitif du champ choisi. Cette dernière est utilisée dans ce rapport.

1.2.3. Polynôme irréductible

Soit *P* un polynôme à coefficients dans *K*. Si *P* est irréductible, alors *P* ne s'annule pas sur *K*. Par contre on n'a pas la réciproque.

1.2.4. Polynômes primitifs

Ce polynôme permet de construire le « corps de Galois » souhaité. Tous les éléments non nuls du corps peuvent être construits en utilisant l'élément α comme racine du polynôme

primitif. Chaque m peut avoir plusieurs polynômes primitifs. On mentionne dans le tableau ci-dessous, Les polynômes primitifs pour les principaux « corps de Galois » :

m	P(X)	m	P(X)
3	$1 + X + X^3$	14	$1 + X + X^6 + X^{10} + X^{14}$
4	$1 + X + X^4$	15	$1 + X + X^{15}$
5	$1 + X^2 + X^5$	16	$1 + X + X^3 + X^{12} + X^{16}$
6	$1 + X + X^6$	17	$1 + X^3 + X^{17}$
7	$1 + X^3 + X^7$	18	$1 + X^7 + X^{18}$
8	$1 + X^2 + X^3 + X^4 + X^8$	19	$1 + X + X^2 + X^5 + X^{19}$
9	$1 + X^4 + X^9$	20	$1 + X^3 + X^{20}$
10	$1 + X^3 + X^{10}$	21	$1 + X^2 + X^{21}$
11	$1 + X^2 + X^{11}$	22	$1 + X + X^{22}$
12	$1 + X + X^4 + X^6 + X^{12}$	23	$1 + X^5 + X^{23}$
13	$1 + X + X^3 + X^4 + X^{13}$	24	$1 + X + X^2 + X^7 + X^{24}$

Tableau 1.1 : Polynômes primitifs dans *GF* (2^m).

1.2.5. Exemple de construction d'un corps de Galois d'un CG(2^4)

On veut construire tous les éléments du *GF* (2^4) à partir du polynôme primitif :

$$p(x) = x^4 + x + 1$$

On a α est racine du polynôme primitif

Donc :

$$0 = \alpha^4 + \alpha + 1$$

$$\alpha^4 = \alpha + 1$$

Maintenant, il suffit de multiplier l'élément α par α à chaque étape et réduire par rapport à $\alpha^4 = \alpha + 1$ pour obtenir le champ complet. On aura besoin de 13 multiplications pour compléter ce corps.

Les éléments d'un « champ de Galois » de *GF* (2^4) sont présentés dans le tableau (Tableau 1.2).

Éléments	**Formes polynômiales**	**Formes binaires**	**Formes décimales**
0	0	0000	0
1	1	0001	1
α	α	0010	2
α^2	α^2	0100	4
α^3	α^3	1000	8
α^4	$\alpha + 1$	0011	3
α^5	$\alpha^2 + \alpha$	0110	6
α^6	$\alpha^3 + \alpha^2$	1100	12
α^7	$\alpha^3 + \alpha + 1$	1011	11
α^8	$\alpha^2 + 1$	0101	5
α^9	$\alpha^3 + \alpha$	1010	10
α^{10}	$\alpha^2 + \alpha + 1$	0111	7
α^{11}	$\alpha^3 + \alpha^2 + \alpha$	1110	14
α^{12}	$\alpha^3 + \alpha^2 + \alpha + 1$	1111	15
α^{13}	$\alpha^3 + \alpha^2 + 1$	1101	13
α^{14}	$\alpha^3 + 1$	1001	9

Tableau 1.2 : Éléments de *GF* (2^4).

1.2.6. Dérivation formelle d'un polynôme dans un corps de Galois

La dérivée formelle du polynôme dans un « champ de Galois » *GF* (2^m) est définie avec les puissances paires car les puissances impaires sont précédées de coefficients pairs qui annulent les termes.

Un polynôme d'ordre *m* est défini comme :

$$f(x) = f_m\, x^m + f_{m-1}x^{m-1} + \cdots + f_1 x + f_0.$$

La dérivée formelle de ce polynôme est définie comme :

$$f' = f_1 + 2\,f_2\,x + 3\,f_3 x^2 + \cdots + m\,f_m x^{m-1}$$

$$f' = f_1 + 3\,f_3 x^2 + 5\,f_5 x^4 + \cdots.$$

1.3. Principe des codes de Reed-Solomon

1.3.1. Définition

Les codes de Reed–Solomon sont des codes de détection et de correction des erreurs basés sur les corps de Galois. Ce sont des codes particuliers des codes BCH.

1.3.2. Propriétés des codes Reed-Solomon

Le codeur prend k symboles de donnée (chaque symbole contenant s bits) et calcul les informations de contrôle pour construire n symboles, ce qui donne $n-k$ symboles de contrôle. Le décodeur peut corriger au maximum t symboles, où $2t = n-k$.

La longueur maximale d'un code de Reed–Solomon est définie comme :

$$n = k + 2t = 2^m - 1.$$

Avec :

k : nombre de symboles d'information.

$2t$: nombre de symboles de contrôle.

m : nombre de bits par symbole.

La distance minimale d'un code Reed–Solomon est : $d_{min} = 2t - 1$.

Ces codes présentent les propriétés suivantes :

- Ils sont linéaires.
- Ils sont cycliques, c'est-à-dire, que chaque mot-code décalé engendre un autre mot-code. Tous les codes cycliques peuvent être réduits en gardant la même capacité d'erreur, mais le nouveau code formé n'est alors pas cyclique.
- Ils sont des codes non-binaires. Les codes sont représentés sur des « champs de Galois » de $GF\,(2m\,)$ et non pas sur des champs de $GF\,(2)$.

- Les symboles sont définis comme les coefficients du polynôme et le degré de x indique l'ordre. Ainsi, le symbole avec l'ordre le plus élevé est reçu/envoyé en premier et le dernier symbole reçu/envoyé est celui dont l'ordre est moindre.

1.3.3. Exemple de nombres de symboles corrigeables

Prenons un code de Reed-Solomon, par exemple $RS(n,k) = RS(15,9)$. L'objectif est de découvrir combien de bits sont utilisés pour chaque symbole et combien d'erreurs peut-on corriger.

Le nombre n indique la longueur totale d'un bloc de Reed–Solomon, 15 symboles dans ce cas et le nombre k indique la longueur du bloc d'information, 9 symboles dans cet exemple.

La capacité de correction des erreurs du système est :

$$2t = n - k = 15 - 9 = 6.$$

Donc :

$$t = (n-k)/2 = 3$$

Ce code permettra de corriger 3 symboles.

Le nombre de bits s par symbole est :

$$n = 2^m - 1.$$

Donc :

$$m = 4.$$

Le nombre de bits utilisés pour coder les symboles est donc de 4. Ce qui nous amène à utiliser un « corps de Galois » de GF (2^4).

1.4. Codeur de Reed-Solomon

1.4.1. Théories du codage

L'équation clé [1] définissant le codage systématique de Reed–Solomon (n,k) est :

$$c(x) = i(x)\, x^{n-k} + [i(x)\, x^{n-k}] mod\big(g(x)\big)$$

Avec :

$c(x)$:	Polynôme du mot-code, degré $n-1$.
$i(x)$:	Polynôme d'information, degré $k-1$.
$[i(x)\,x^{n-k}]mod\big(g(x)\big)$:	Polynôme de contrôle, degré $n-k-1$.
$g(x)$:	Polynôme générateur, degré $n-k$.

Le codage systématique signifie que l'information est codée dans le degré élevé du mot code et que les symboles de contrôle sont introduits après les mots d'information.

1.4.2. Polynôme générateur

Le polynôme générateur sert à générer les symboles de contrôle. Tous les codes Reed-Solomon sont valables si et seulement si ils sont divisibles par leur polynôme générateur, *c* (*x*) doit être divisible par *g*(*x*).

Pour générer un code correcteur d'erreurs de t symboles, on devrait avoir un polynôme générateur de puissance α^{2t}. La puissance maximale du polynôme est déterminée grâce la distance minimale qui est $d_{min} = 2t + 1$. On devrait avoir $2t + 1$ termes du polynôme générateur.

Le polynôme générateur est sous la forme :

$$g(x) = (x - \alpha^1)(x - \alpha^2) \dots (x - \alpha^{2t})$$
$$g(x) = g_{2t}x^{2t} + g_{2t-1}x^{2t-1} + \cdots + g_1x + g_0.$$

1.4.3. Exemple de calcul des coefficients du polynôme générateur pour RS(15,9)

On a :

$$g(x) = (x - \alpha^1)(x - \alpha^2) \dots (x - \alpha^{2t})$$
$$g(x) = (x - \alpha^1)(x - \alpha^2) \dots (x - \alpha^6)$$
$$= (x^2 + \alpha^5x + \alpha^3)(x^2 + \alpha^7x + \alpha^7)(x^2 + \alpha^5x + \alpha^7)$$
$$= (x^4 + \alpha^{13}x^3 + \alpha^6x^2 + \alpha^3x + \alpha^{10})(x^2 + \alpha^9x + \alpha^{11}).$$
$$= x^6 + x^5(\alpha^{13} + \alpha^9) + x^4(\alpha^6 + \alpha^7 + \alpha^{11}) + x^3(\alpha^3 + \alpha^9 + 1)$$
$$+ x^2(\alpha^{10} + \alpha^{12} + \alpha^2) + x(\alpha^4 + \alpha^{14}) + \alpha^6$$

$$= x^6 + \alpha^{10}x^5 + \alpha^{14}x^4 + \alpha^4x^3 + \alpha^6x^2 + \alpha^9x + \alpha^6.$$

En prenant l'équivalent en décimale, on trouve :

$$g(x) = x^6 + 7x^5 + 9x^4 + 3x^3 + 12x^2 + 10x + 12$$

1.5. Décodeur de Reed-Solomon

Les étapes nécessaires pour le décodage des codes de Reed-Solomon sont :

- Calcul du syndrome.
- Calcul des polynômes de localisation des erreurs et de d'amplitude.
- Calcul des racines et évaluation des deux polynômes (de localisation des erreurs et de d'amplitude des erreurs).
- Sommation du polynôme constitué et du polynôme reçu pour reconstituer l'information de départ sans erreur.

Le schéma de décodage est montré sur la figure (Figure 1.1)

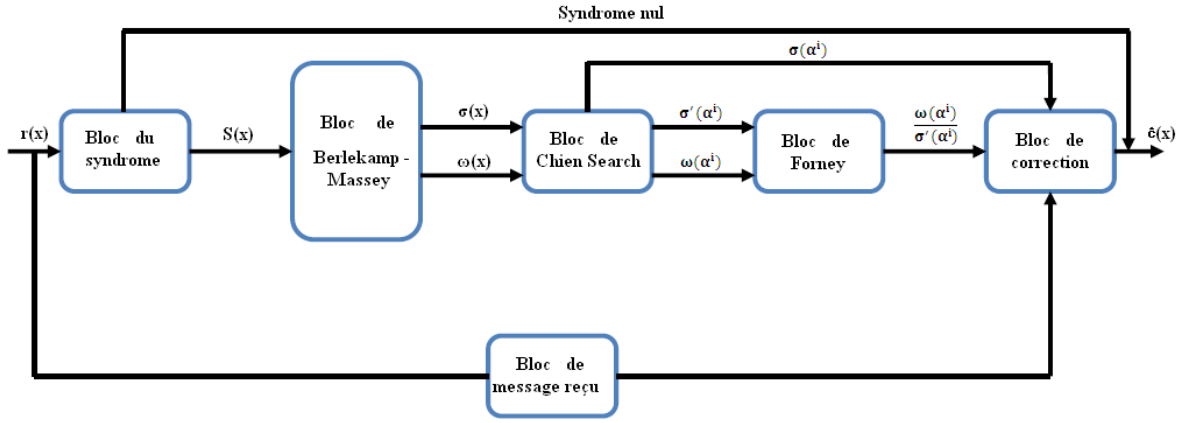


Figure 1.1 : Schéma du décodage.

Avec :

$r(x)$: Mot code reçu,

$S(x)$: Syndrome calculé,

$\omega(x)$: Polynôme d'amplitude des erreurs,

$\sigma(x)$: Polynôme de localisation des erreurs,

$\sigma(\alpha^i)$: polynôme de localisation des erreurs évalué pour tous les éléments compris dans GF(2^m),

$\sigma'(\alpha^i)$: Dérivée du polynôme de localisation des erreurs évalué pour tous les éléments compris dans $GF(2^m)$,

$\omega(\alpha^i)$: polynôme d'amplitude des erreurs évalué pour tous les éléments compris dans $GF(2^m)$,

$\frac{\omega(\alpha^i)}{\sigma'(\alpha^i)}$: Division entre polynôme d'amplitude évalué et la dérivée du polynôme de localisation des erreurs évaluées,

$\hat{c}(x)$: Sortie du décodeur.

1.5.1. Calcul du syndrome

Considérons un code de Reed-Solomon $c(x)$ correspondant au code transmis et soit $r(x)$ le code que l'on reçoit. Le polynome d'erreurs introduit par le canal est définit comme :

$$e(x) = r(x) - c(x) = r(x) + c(x)$$

Le calcul du syndrome est défini comme le reste de la division entre le polynôme reçu $r(x)$ et le polynôme générateur $g(x)$. Le reste indiquera la présence d'erreurs. Il peut être aussi effectué par un processus itératif.

On a :

$$r(x) = c(x) + e(x)$$

On obtient :

$$S_i = r(\alpha^i) = c(\alpha^i) + e(\alpha^i) = e(\alpha^i)$$

Seuls les premiers $2t$ symboles du syndrome qui sont connus.

Ainsi, le syndrome sous forme polynomiale sera :

$$S(x) = S_{2t}x^{2t-1} + \cdots + S_2x + S_1$$

Si le code reçu r (x) n'est pas affecté par des erreurs alors tous les coefficients du syndrome seront nuls $(r(x) = c(x))$.

1.5.2. Calcul des polynômes localisateur et d'amplitude

Le calcul des polynômes de localisation des erreurs et d'amplitude des erreurs est basé sur versions reformulées, soit de l'algorithme du Berlekamp-Massey, ou l'algorithme d'Euclide. Dans ce qui suit, nous présentons quelques algorithmes basés sur les registres à décalage de Berlekamp-Massey

- **<u>Algorithme de Berlekamp-Massey (BM)</u>**

L'algorithme de Berlekamp – Massey est basé sur la synthèse du polynôme de localisation des erreurs en utilisant des registres à décalage pour la résolution des identités des Newton.

En 1969, Massey a développé un algorithme itératif [2] basé sur le principe des registres à décalage de Berlekamp. Ce nouvel algorithme est appelé algorithme de Berlekamp–Massey.

L'identité de Newton à résoudre pour les codes de Reed-Solomon est :

$$S_{v+1} + \sigma_1 S_v + \sigma_2 S_{v-1} + \cdots + \sigma_v S_1 = 0.$$

$$S_{v+2} + \sigma_1 S_{v+1} + \sigma_2 S_v + \cdots + \sigma_v S_2 = 0.$$

... ...

$$S_{2t} + \sigma_1 S_{2t-1} + \sigma_2 S_{2t-2} + \cdots + \sigma_{2t} S_{2t-v} = 0.$$

Le but de l'algorithme de Berlekamp-Massey est de trouver le polynôme $\sigma(x)$ de degré le plus petit possible et satisfaisant l'identité de Newton. Le calcul de $\sigma(x)$ est effectué itérativement avec $2t$ étapes selon le diagramme en flèches de la figure (Figure 1.2).

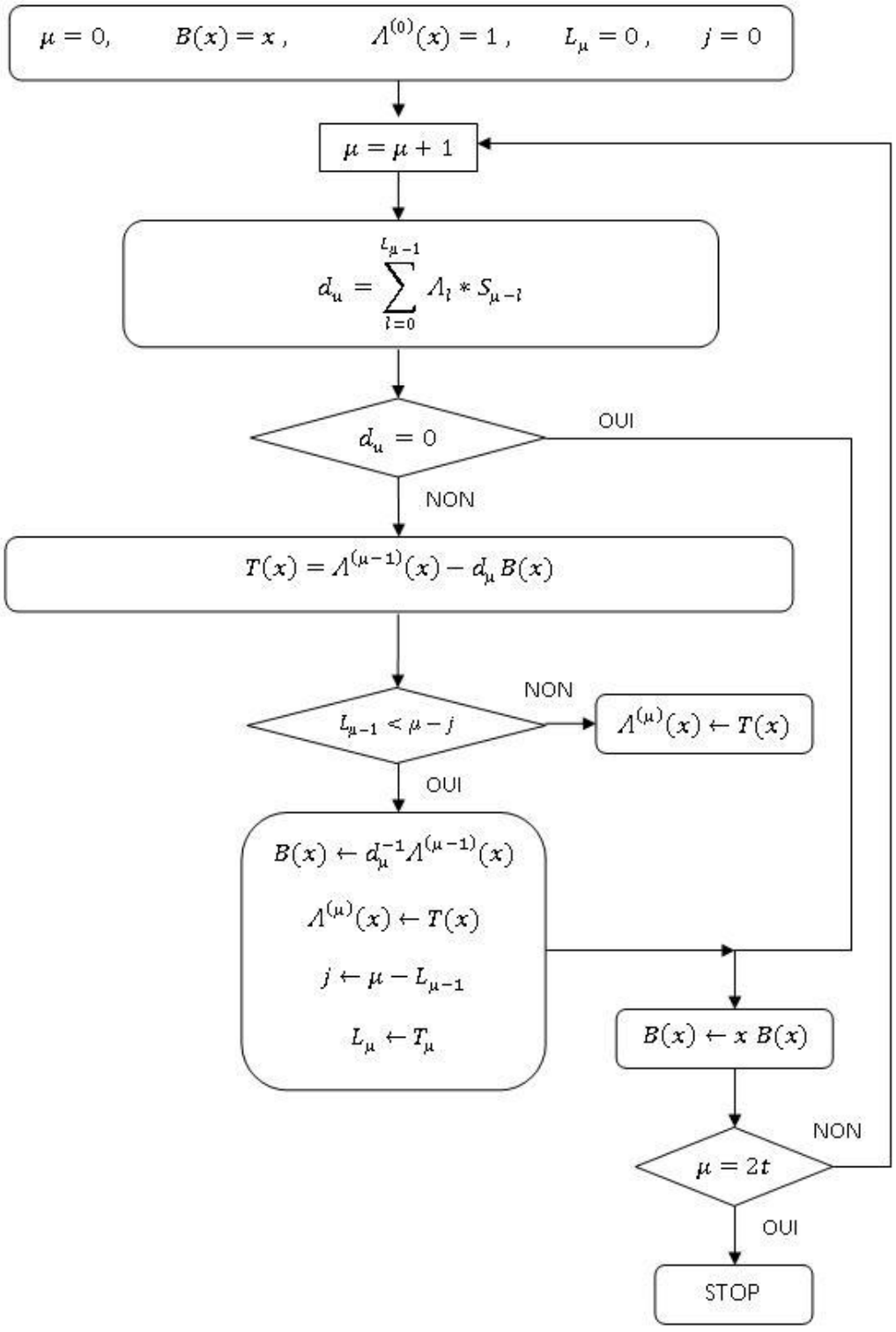


Figure 1.2 : L'organigramme en flèche de l'algorithme de Berlekamp-Massey.

Avec :

$\Lambda^{(\mu)} = \sigma^{(\mu)}(x)$: polynôme de localisation des erreurs après μ étapes.

L_μ : Degré du polynôme de localisation des erreurs après μ étapes.

d_μ: Incohérence après μ étapes.

$T\ (x)$: Polynôme de connexion annulant l'incohérence.

Le polynôme calculé sera sous la forme :

$$\sigma(x) = \sigma_0 + \sigma_1 x + \cdots + \sigma_t x^t$$

Le polynôme d'amplitude sera de degré $t - 1$:

$$\omega(x) = [S(x)\sigma(x)] mod(x^t)$$

$$\omega(x) = [(S_{2t}x^{2t-1} + \cdots + S_2 x + S1)(\sigma_0 + \sigma_1 x + \cdots + \sigma_t x^t)] mod(x^t)$$

$$\omega(x) = (S_t + \sigma_1 S_{t-1} + \cdots)x^{t-1} + \cdots + (S_2 + \sigma_1 S_1)x + S_1$$

- **L'algorithme de Berlekamp-Massey sans inversion (iBM)**

Une amélioration de l'algorithme de Berlekamp-Massey a été faite dans [3] en supprimant la nécessité d'effectuer l'inversion dans le champ de Galois. L'algorithme original de Berlekamp-Massey sans inversion (Inversionless Berlekamp-Massey algorithm « iBM ») calcul le polynôme de localisation des erreurs. Le polynôme d'amplitude des erreurs est calculé après. L'organigramme de cet algorithme est montré sur la figure (Figure 1.3).

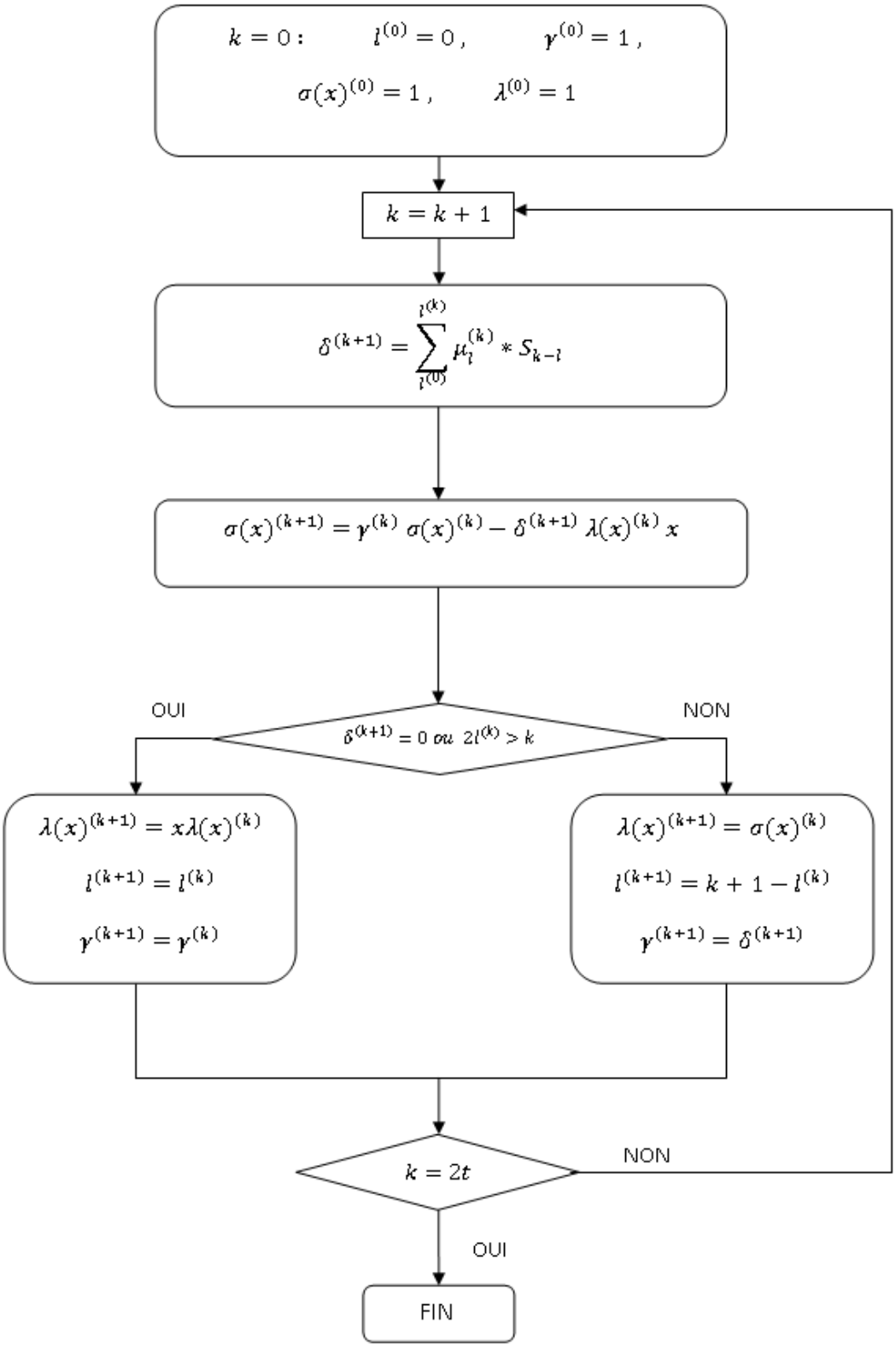


Figure 1.3 : L'organigramme de l'algorithme iBM pour le calcul du polynôme de localisation et le polynôme d'amplitude des erreurs.

Quand l'algorithme se termine, le polynôme de localisation des erreurs est $\sigma(x)$. À ce niveau, le polynôme d'amplitude des erreurs $\omega(x)$ peut être calculé à partir du polynôme de syndrome et le polynôme de localisation des erreurs comme montré dans l'équation :

$$S(x)\sigma(x) = \omega(x) mod(x^{2t})$$

Cette opération s'effectue après que le polynôme de localisation des erreurs est trouvé, et par conséquent, il influe sur la latence de décodeur de Reed-Solomon. La section suivantes décrit un nouveau résultat ; l'algorithme de Berlekamp-Massey sans inversion étendu (extended inversionless Berlekamp-Massey algorithm EiBM). Cet algorithme calcul les deux polynômes en tandem.

- **Algorithme de Berlekamp-Massey sans Inversion Etendu (EIBM)**

L'algorithme EIBM (Extended Inversionless Berlekamp-Massey) permet le calcul de polynôme de localisation des erreurs et le polynôme d'amplitude des erreurs en même temps. Cet algorithme est présenté et prouvé dans [4].

Les étapes de cet algorithme sont illustrées sur l'organigramme de la figure (Figure 1.4).

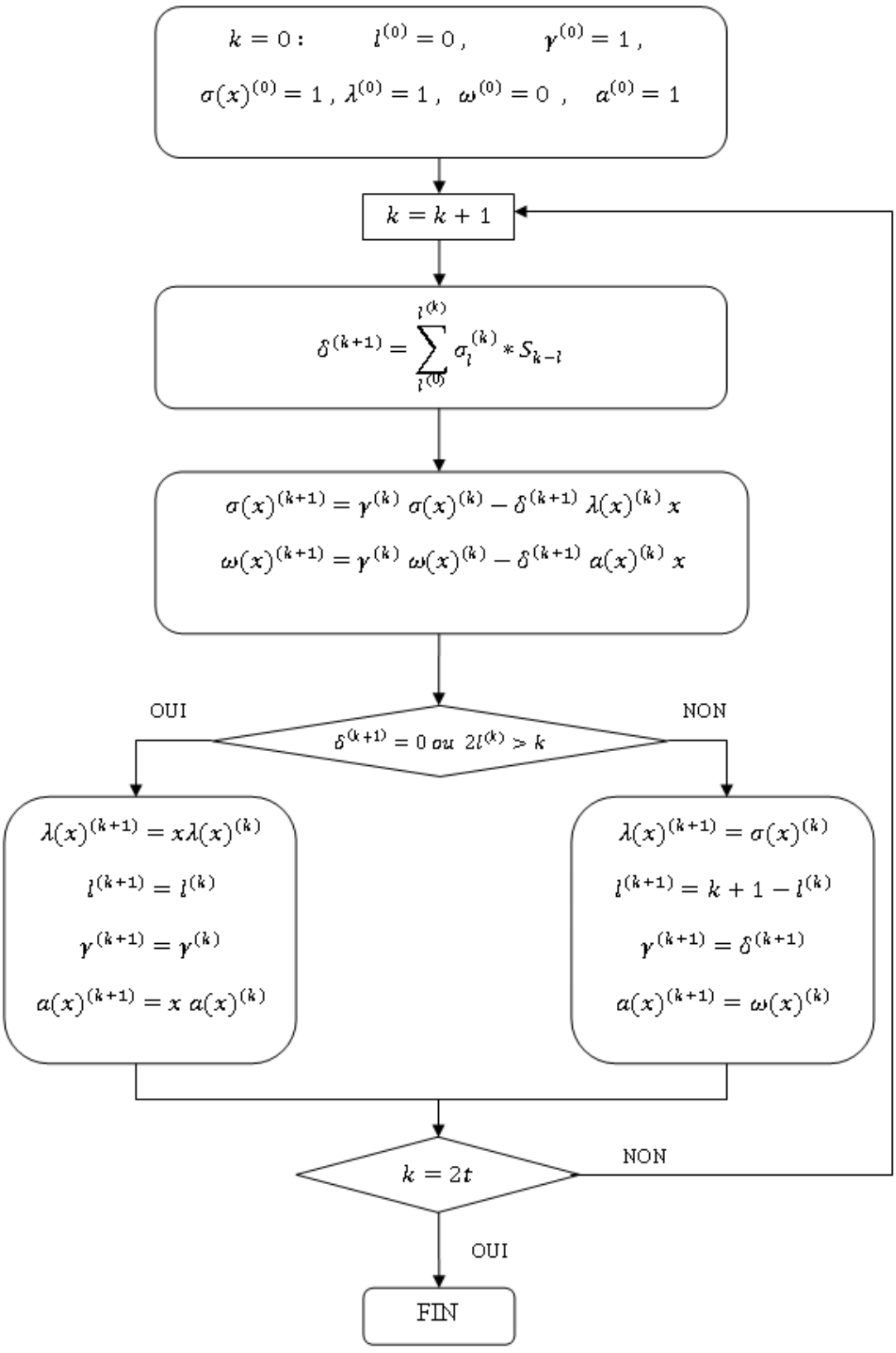


Figure 1.4 : L'organigramme en flèche de l'algorithme EIBM.

1.5.3. Chien Search

Une fois le polynôme de localisation des erreurs calculé, on doit évaluer ses racines et sa dérivée.

L'évaluation des racines s'effectue avec l'algorithme appelé « Chien Search » qui évalue toutes les possibilités. Par exemple pour un RS(15,9), on évalue le polynôme de localisation des erreurs et sa dérivée pour tous les éléments du « corps de Galois » *GF* (2^4), sauf pour l'élément nul.

Cet algorithme permet ainsi l'évaluation du polynôme d'amplitude.

1.5.4. Algorithme de Forney

Cet algorithme permet de construire le polynôme d'erreurs $e\ (x)$ à additionner avec le polynôme reçu $r\ (x)$ pour reconstituer le polynôme $c\ (x)$. Pour le calcul du polynôme d'erreurs $e(x)$, les polynomes $\sigma(\alpha^i)$, $\sigma'(\alpha^i)$ et $\omega(\alpha^i)$ sont nécessaires.

L'algorithme de Forney est défini comme :

$$e_i = \frac{\omega(\alpha^i)}{\sigma'(\alpha^i)}$$

Avec :

$\omega(\alpha^i)$: Polynôme d'amplitude évalué pour les valeurs de *GF* (2^4).

$\sigma'(\alpha^i)$: Dérivée du polynôme de localisation des erreurs pour les valeurs de $GF\ (24)$.

1.6. Conclusion

Dans ce chapitre, on a exposé les bases mathématiques de la théorie de codage et décodage de Reed-Solomon. Nous avons ensuite présenté l'algorithme de décodage RS qui est basé sur les registres à décalage de Berlekamp- Massey. On a mis en évidence l'importance de l'algorithme de Berlekamp-Massey sans inversion étendu (EIBM). Cet algorithme permet le calcul des polynômes de localisation et d'amplitude en même temps.

Chapitre II

Simulation, implémentation et résultats

2.1. Introduction

Nombreuse applications surtout dans le domaine des communications sans fils évoluent vers des systèmes miniaturisés à hautes performances, les circuits reconfigurables fournissent une plateforme flexible et efficace pour répondre aux exigences des conceptions complexes en termes de vitesse d'opération, d'espace occupé, de consommation d'énergie, du coût de conception et de temps de mise sur le marché.

Dans ce chapitre, on donnera dans un premier temps l'architecture de bloc de Berlekamp-Massey pour la résolution de l'équation clé des codes de Reed-Solomon. Cette architecture est basée sur l'algorithme EIBM exposé dans le chapitre 1. On développera ensuite l'implémentation de bloc de Berlekamp-Massey sur FPGA.

2.2. Présentation de la carte FPGA

Les circuits FPGA utilisés sont le XC2V3000-4FG676 de la famille Virtex II de Xilinx. Ce circuit est intégré autour d'une carte multimédia Celoxica RC203E.

2.3. L'outil ISE de Xilinx

ISE Foundation 10.1 est un environnement intégré de développement de systèmes numériques ayant pour but une implémentation matérielle sur FPGA de la compagnie Xilinx. Les designs peuvent être décrits sous trois formes principales :

- schémas.
- langage de description matérielle (HDL) comme VHDL et Verilog.
- diagrammes d'états.

ISE intègre donc différents outils permettant de passer à travers tout le flot de conception d'un système numérique :

- un éditeur de textes, de schémas et de diagrammes d'état.
- un compilateur VHDL/Verilog.
- un outil de simulation.
- des outils pour la gestion des contraintes temporelles.

- des outils pour la synthèse.
- des outils pour la vérification.
- des outils pour l'implémentation sur FPGA.

2.4.Architecture de bloc du Berlekamp-Massey

La cellule élémentaire qui réalise une itération selon l'algorithme EIBM est montrée sur la figure (Figure 2.1).

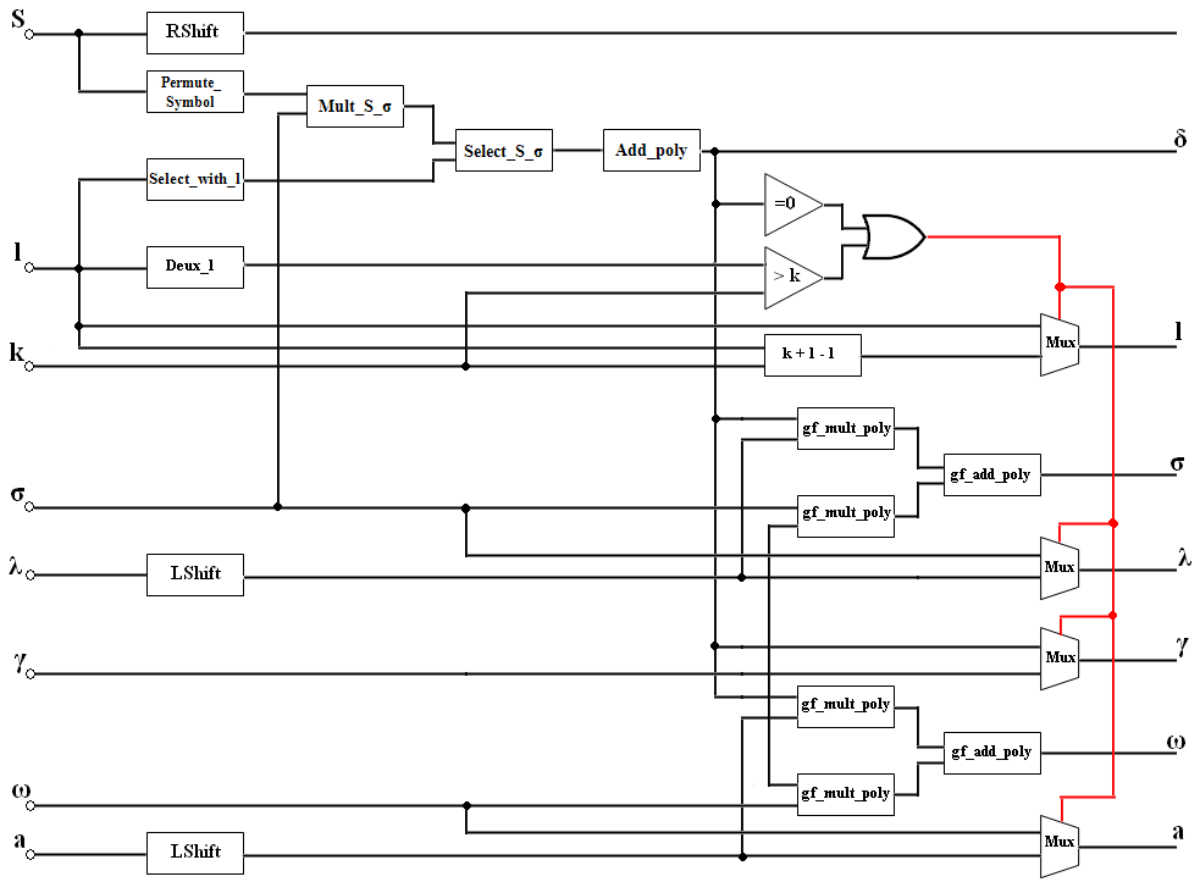


Figure 2.1 : Schéma de la cellule EIBM.

Le bloc « RShift » sert à faire décaler les symboles du syndrome à droite. En effet, l'entrée de ce bloc est un signal S de $(3t)$ symboles, chaque symbole est sur m bits (le signal S est alors sur $(3t) * m$ bits). Initialement, les symboles du syndrome sont placés sur les bits du poids forts du signal S, et les autres bits ($t * m$ bits) sont mis à zéro. À chaque itération, une réalise une opération de décalage à droite du signal S. Le nombre total des itérations est

$(2t)$. Le bloc « Permut_Symbol » a comme entrée les $(t+1)$ premiers symboles de signal S ($(t+1) * m$ bits). Ce bloc réalise une permutation entre les symboles d'entrée ; les symboles du poids forts deviennent ceux du poids faibles et les symboles du poids faibles deviennent ceux du poids forts. On réalise ensuite une multiplication entre la sortie du ce bloc (le bloc « Permut_Symbol ») et le polynôme de localisation des erreurs $\sigma(x)$. Cette opération est effectuée à l'aide du bloc « Mult_S_σ ». Ce bloc réalise une multiplication dans le champ de Galois (GF(2^m))symbole par symbole comme montré sur la figure.

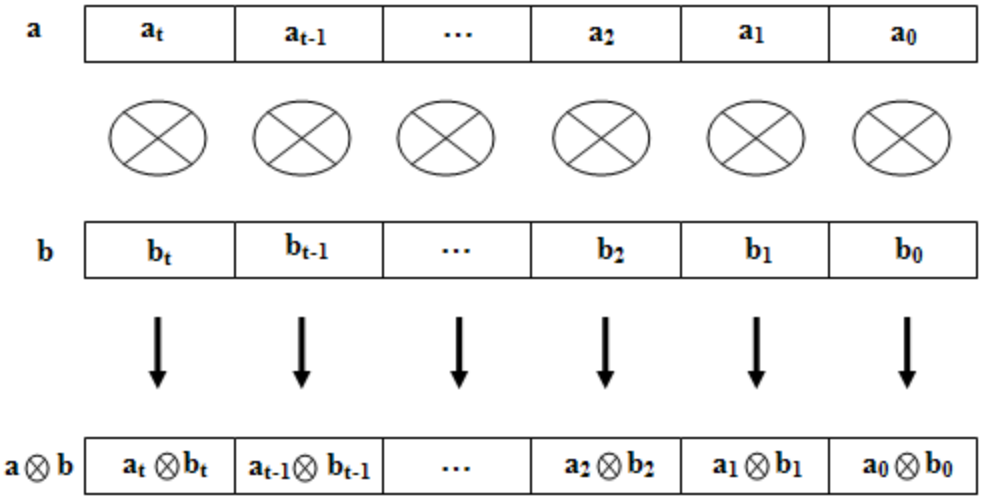


Figure 2.2 : Principe de la multiplication effectuée par le bloc « Mult_S_σ ».

Les blocs « Select_with_l » et « Select_S_σ » servent à sélectionner les symboles de la sortie du bloc « Mult_S_σ » à additionner pour trouver la valeur de δ. Le bloc « Select_with_l » fait sortir un signal sur « m bits » tout dépend du signal de son entrée l. Le bloc « convolution_term_multipl » mis à zéro les symboles qui ne devraient pas entrer dans le calcul de δ et garde les autres inchangés. Le bloc « Add_poly» réalise la somme des symboles de la sortie du bloc « Select_S_σ » pour avoir le signal δ.

Pour le calcul du polynôme de localisation des erreurs $\sigma(x)$ et le polynôme d'amplitude des erreurs $\omega(x)$, différents blocs sont nécessaires. Le bloc « LShift » sert à

réaliser un décalage à gauche du siganl λ ou b. Ces deux polynômes sont représentés par les deux signaux σ et b respectivement. Ces signaux (σ et b) sont sur $((t+1)*m)$ bits (concaténation du $(t+1)$ symboles). Le bloc « gf_mult_poly » réalise une multiplication d'un signal sur $((t+1)*m)$ bits avec un signal sur (m) bits. Le premier signal est la représentation d'un polynôme de degré (t) $((t+1)$ symboles) et le deuxième est un symbole (il peut être δ ou γ). Le bloc « gf_add_poly » effectue une addition dans le champ de Galois symbole par symbole entre deux polynômes. Comme l'addition dans le champ de Galois (GF(2)) est réalisée avec une porte XOR, l'addition effectuée par le bloc « gf_add_poly » peut être effectuée bit par bit.

Le test $(\delta = 0)$ est effectué par le bloc « is_zero ». La sortie de ce bloc est un signal de niveau logique haut quand la condition est vérifiée. Le test $(2l > k)$ est réalisé avec le bloc « Comparateur ». Ce bloc compare la sortie du bloc « Deux_l », qui comme son nom indique, calcul la valeur 2l, avec le signal k. Le signal k est valeur de l'itération en cours. Il est généré par un compteur, et prend des valeurs de 0 jusqu'au $2t$. La sortie du bloc « Comparateur » est mis à 1 quand $2l > k$. Les sorites du bloc « is_zero » et « Comparateur » sont les entrées d'une porte OU. La sortie cette porte sert comme une entrée de sélection pour les multiplexeurs « Mux ». Ainsi, ces multiplexeurs sélectionne les bonnes valeurs des signaux l, λ, γ et b.

Un registre est placé à la fin de la cellule pour la sauvegarde des différents résultats et assurer le calcul de l'itération suivante. Quand le nombre d'itération est égal à $2t$, l'algorithme s'arrête. Le polynôme de localisation des erreurs est alors obtenu sur le signal σ tandis que le polynôme d'amplitude des erreurs est obtenu en prenant le signal ω sans le symbole du poids faible (sans les premiers m bits).

2.5. Implémentation de bloc du Berlekamp-Massey

2.5.1. Structure de bloc Berlekamp-Massey

Le circuit de bloc du Berlekamp-Massey est composé d'un seul fichier qui contient les connexions entre les différents blocs déclarés comme composant. Ces derniers sont décrits

dans des fichiers indépendants. Ce circuit est commandé par une machine d'état représenté sur la figure (Figure 2.4). La structure de bloc du Berlekamp-Massey est celle représentée sur la figure (Figure 2.3).

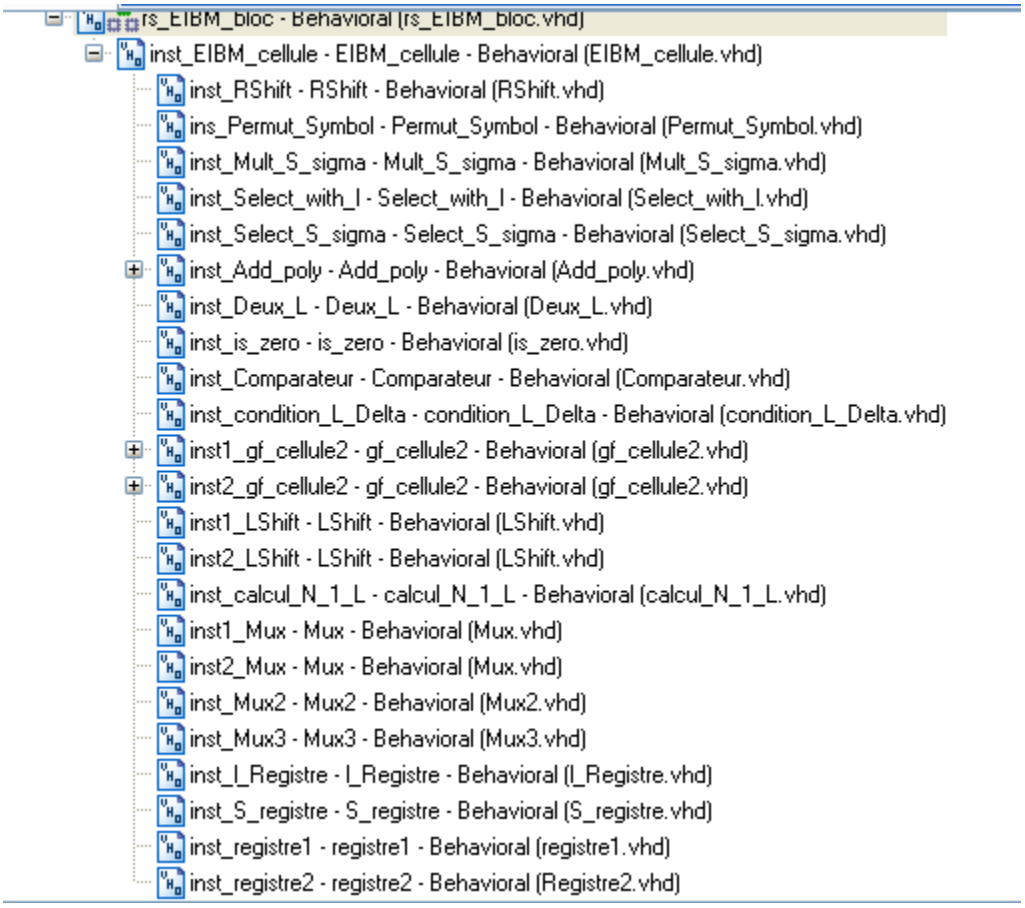


Figure 2.3 : Structure de bloc du Berlekamp-Massey.

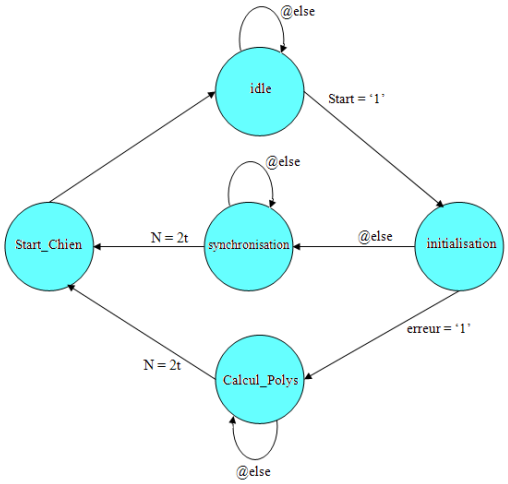


Figure 2.4 : Machine d'état de module du Berlekamp-Massey.

La cellule élémentaire pour le calcul de bloc du Berlekamp-Massey « EIBM » est représentée sur la figure (Figure 2.5).

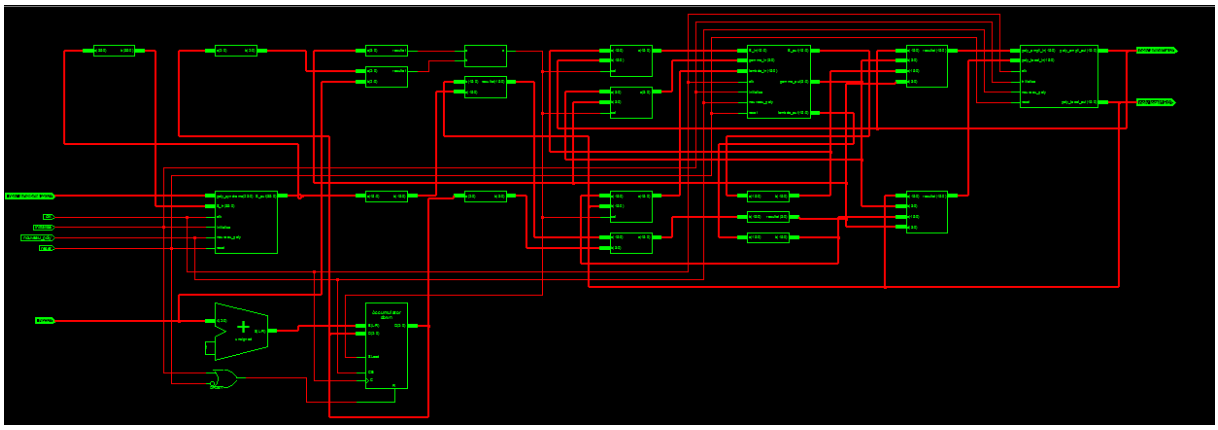


Figure 2.5 : Cellule élémentaire pour le calcul de bloc du Berlekamp-Massey.

Le bloc qui calcule le polynôme d'amplitude et de localisation des erreurs suivant l'algorithme de Berlekamp-Massey « EIBM » est :

Figure 2.6 : Schéma de bloc du Berlekamp-Massey.

2.5.2. Signaux de commande et d'entrée/sortie de bloc du Berlekamp-Massey

Les signaux d'entrée/sortie sont :

- clk : signal d'horloge (std_logic)
- start_EIBM : signal qui permet d'activer le bloc du berlekamp-massey (std_logic), c'est une impulsion d'un cycle d'horloge qui précède le polynôme de syndrome.
- reset : signal permettant la remise à zéro du syndrome (std_logic).
- erreurs : signal qui permet de détecter la présence d'erreur.Il est à 1 lorsque qu'il y a des erreurs (std_logic).
- poly_syndrome : signal d'entrée des symboles de polynôme de syndrome. Cette entrée est sur 24 bits pour le RS(15,9) et sur 128 bits pour le RS(255,239).
- poly_amplitude : signal de sortie de polynôme d'amplitude des erreurs. Cette sortie est sur 12 bits pour le RS(15,9) et sur 64 bits pour le RS(255,239).
- poly_localisation : signal de sortie de polynôme de localisation des erreurs. Cette sortie est sur 16 bits pour le RS(15,9) et sur 72 bits pour le RS(255,239).
- Start_chienSearch : signal de sortie qui permet l'activation du bloc de chiensearch (std_logic).

2.5.3. Simulation de bloc du Berlekamp-Massey

La figure (Figure 2.7) représente un exemple de simulation de bloc du berlekamp-massey RS(15,9)

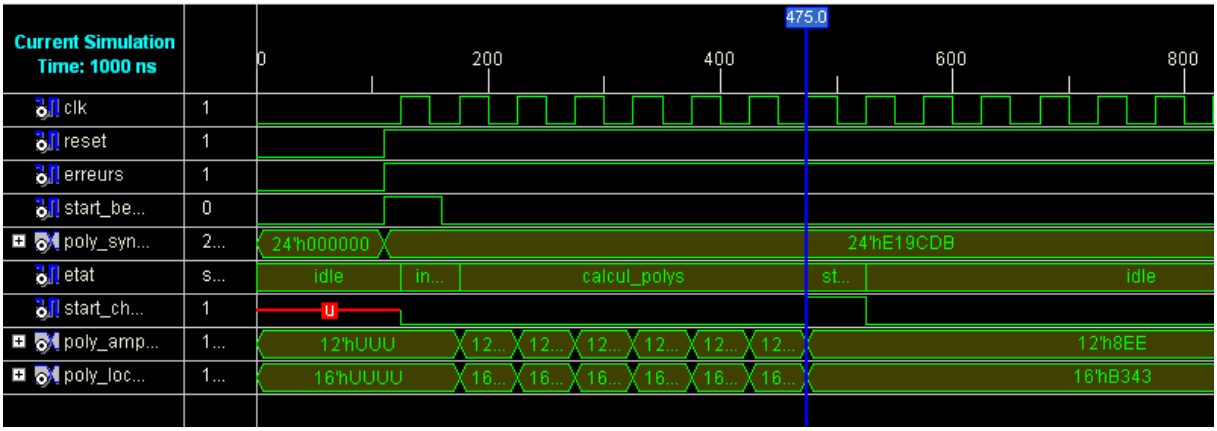


Figure 2.7 : Exemple de simulation de bloc du Berlekamp_Massey.

2.5.4. Rapport de synthèse

Les ressources hardware utilisées et les performances temporelles pour le bloc du Berlekamp-Massey pour le code RS(15,9) et RS(255,239) après placement et routage sous Xilinx Virtex 2xc2v3000-4bf676 sont représentées respectivement dans les tableaux et :

Fpga	**XC2V3000-4BF676**			
	RS(15,9)		**RS(255,239)**	
Type de ressources	**Utilisé**	**taux**	**Utilisé**	**taux**
Slice Flip Flop	109 de 28672	1%	507 de 28672	1%
4 inputs LUTs	339 de 28672	1%	2880 de 28672	10%
Bonded IOBs	57 de 484	11%	269 de 484	55 %
Total occuped slices	180 de 14336	1%	1520 de 14336	10 %

Tableau 2.1 : Ressource hardware utilisé sous Xilinx Virtex 2 xc2v3000-4bf676 pour le bloc du Berlekamp-Massey.

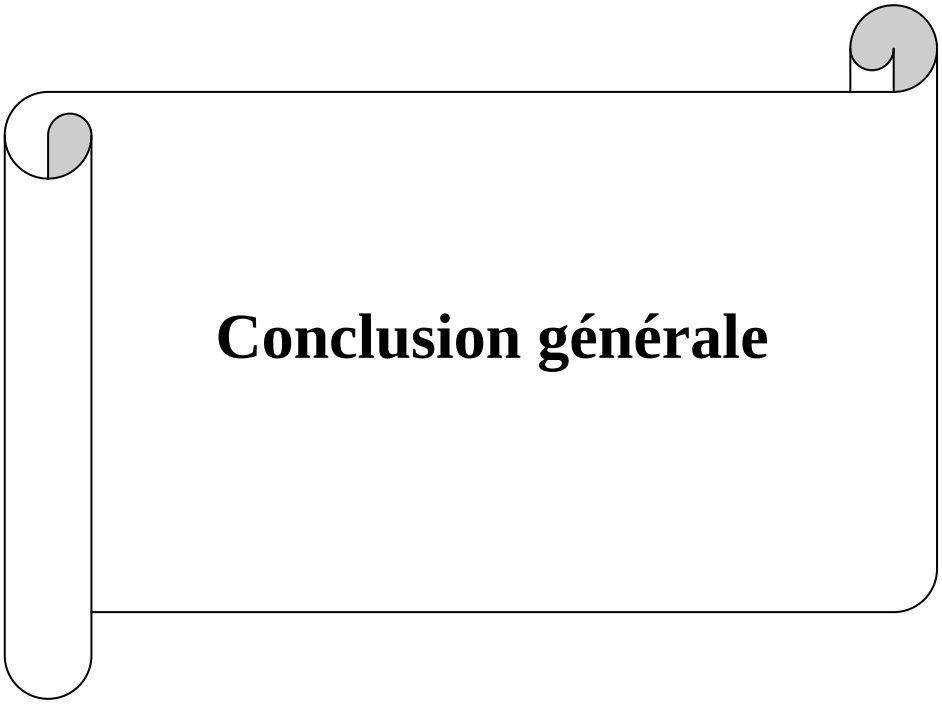
FPGA device	XC2V3000-4BF676	
Le code RS	**RS(15,9)**	**RS(255,239)**
Vitesse	-4	-4
Période minimum	8,676 ns	12.266 ns
Fréquence maximale	115,260 Mhz	81,527MHz

Tableau 2.2 : Performance temporelle pour le bloc du Berlekamp-Massey.

2.6. Conclusion

Dans ce chapitre, on a présenté d'abord l'architecture proposée pour le bloc de Berlekamp-Massey en choisissant l'algorithme (EIBM). Ensuite, on a montré les résultats de l'implémentation de ce bloc pour les codes RS(15,9) et RS(255,239).

Les résultats de l'implémentation montre que les ressources hardware requises augmente avec l'augmentation deb la taille de code RS.



Conclusion générale

L'objectif de ce mémoire est l'étude et l'implémentation sur circuit reconfigurable de bloc de Berlekamp-Massey pour le décodage des codes de Reed-Solomon. On a décrit le principe de fonctionnement de codeur et décodeur RS, ainsi que l'architecture adoptée pour la résolution de l'équation clé.

Le décodage de Reed-Solomon présente la partie la plus complexe tant au niveau théorique qu'au niveau hardware. Le bloc principal de décodage est celui qui résout résolution de l'équation clé permettant de calculer le polynôme de localisation des erreurs et d'amplitude des erreurs. Plusieurs algorithmes existent, et nous avons choisi l'algorithme de Berlekamp-Massey sans inversion étendu (EIBM).

On a développé les codes en VHDL du bloc de Berlekamp-Massey pour les codes RS(15,9) et RS(255,239) en suivant l'architecture proposée. Ce bloc est simulé et synthétisé.

À l'avenir, il serait pertinent d'apporte des modifications sur l'architecture proposée pour le bloc de résolution de l'équation clé. Cela pourrait permettre de diminuer les ressources hardware à utiliser et d'améliorer les performances temporelles. Il serait pertinent aussi d'implémenter tous les blocs du décodeur de Reed-Solomon pour la correction aussi bien des erreurs et des effacements.

Bibliographie

[1] Samuele Dietler, « Implémentation de codes de Reed-Solomon sur FPGA pour communications spatiales », Projet de diplôme, Haute École d'Ingénierie et de Gestion du Canton de Vaud, 2005.

[2] J. L. Massey, "Shift-Register Synthesis and BCH Decoding", IEEE Transactions on Information Theory, pp. 122-127, 1969.

[3] I.S. Reed, M.T. Shih, T.K. Truong, « VLSI design of Inverse-free Berlekamp-Massey algorithm », IEE Proceedings-E, Vol. 138, No.5, Septembre 1991.

[4] V. Glavac, « A VHDL code generator for Reed-Solomon encoders and decoders », mémoire pour l'obtention du grade de Master des Sciences Appliquées, Université de Concordia, 2003.