

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 d'Ingénieur d'État en Électronique**

Intitulé :

**Implémentation de l'Algorithme de Codage par
Prédiction Linéaire sur un Circuit FPGA**

Réalisé par :

M. GOUICEM Imad-Eddine

M. SOLTANA Samir

Proposé et dirigé par :

Pr D. BERKANI

Promotion : Juin 2008

ملخص

في هذا العمل قمنا بتمثيل خوارزمية التعرف الخطي على دارة قابلة للبرمجة بالحقل , لهذا درسنا إنتاج و بعض مميزات الصوت. ثم درسنا بعض تقنيات التشفير و اخترنا خوارزمية التعرف, الخطي هذا ما جعلنا ندرس الدارة القابلة للبرمجة بالحقل و لغة الوصف المطورة للدارة , وأخيرا قمنا بوصفنا دارة للإخراج عوامل الارتباط الذاتي وبرمجنا خوارزمية ليفنسون داربان

كلمات مفتاحية: التمثيل على دارة FPGA، تشفير تنبئي، Levinson-Durbin، شعاع الارتباط الذاتي , دارات الحساب

Résumé

Dans ce travail, nous avons implémenté une partie de codage par la prédiction linéaire sur FPGA, pour cela on a étudié la production et quelques propriétés de la parole, puis on a fait une vision sur les techniques de codage et parmi ces technique nous avons choisi à implémenter la prédiction linéaire, ce qui nécessite une étude de l'architecture de la carte VIRTEX II et le langage de description VHDL, et enfin nous avons fait une conception du circuit de calcul pour l'autocorrélation et un programmes pour l'algorithmme de Levinson Durbin.

Mots clés : implémentation sur circuit FPGA, codage prédictif, Levinson-Durbin, vecteur autocorrélation, circuits de calculs.

Abstract

In this work we implemented the linear prediction on FPGA. For that, we have studied the production, some properties of the speech, then we have printed a vision on the techniques of coding and among these techniques we chose to implement the linear prediction, which requires to study's the architecture of VIRTEX II and the language of description VHDL. Finally we made a design of channel computation for the autocorrelation and programs for the algorithm of Levinson Durbin.

Key word: implementation on FPGA, predictive coding, Levinson-Durbin, autocorrélation vector, calculating circuits.

Remerciements

Nous remercions le bon Dieu de nous avoir donné la volonté et la patience qui nous ont permis de mener à bien ce travail.

Nous tenons à exprimer nos vifs remerciements à notre promoteur Monsieur **Daoued BERKANI** Professeur au département de l'électronique de l'École Nationale Polytechnique pour nous avoir encadré durant notre projet de fin d'études et nous conseillé tout le long de notre travail.

Nous remercions également Monsieur **R. SAADOUN**, directeur du centre de calcul de l'École Nationale Polytechnique.

Nous remercions également Madame **L. HAMMAMI**, Maitre de conférences à l'École Nationale Polytechnique.

Nous remercions également Monsieur **C. LARBESSE** , Maitre de conférences à l'École Nationale Polytechnique.

Nous remercions chaleureusement les membres du jury pour l'honneur qu'ils nous ont fait en acceptant d'évaluer notre travail.

Enfin, nous aimerions adresser nos plus fervents remerciements à nos parents, car nul autres qu'eux se sont plus sacrifiés pour notre bien et l'accomplissement de nos projets. Ils ont fait de nous ce que nous sommes aujourd'hui, et pour cela, nous leurs dédions ce mémoire.

Que tous ceux qui ont contribué de près ou de loin à la réalisation de ce modeste travail « **GEULLAL, DIBE, AMINA, IMANE,...**» trouvent ici l'expression de notre sincère gratitude.

SOMMAIRE

INTRODUCTION GENERALE	4
<i>Prédiction Linéaire</i>	5
Introduction.....	5
1.1 La Production de la Voix	5
1.1.1. Quelques propriétés importantes de la parole	6
1.1.2. Propriétés des Sons Voisés et des Sons Non Voisés.....	6
1.1.3. Modélisation du Processus de Production de la Parole	8
1.2. La Prédiction Linéaire.....	9
1.2.1. Principes.....	9
1.2.2. Minimisation d'Erreur de Prédiction	9
1.2.3. Algorithme de Levinson-Durbin.....	10
1.3. Analyse de Fourier à Court Terme.....	13
1.4. Prédiction Linéaire rétrograde (Backward) et Filtre en Treillis.....	14
1.5. Représentation des Coefficients de Prédiction	17
1.5.1. Les Coefficients de Réflexions	17
1.5.2. Les Coefficients (LARs) et (ASRC).....	17
1.5.3. Les Fréquences de Raies Spectrales	18
1.6. Conversion LSP-LPC.....	20
1.7. Détecteur de Voisement.....	21
1.7.1. Mesure d'Aplatissement du Spectre	22
1.7.2. Énergie et Taux de Passage par Zéro	22
1.8. Détection du pitch	22
Conclusion	22
<i>Techniques de codage de la parole</i>	23
Introduction.....	23
2.1. Mesure de L'information et Entropie de la Source	23
2.2. La Quantification Scalaire	24
2.3. Codage de Formes d'ondes.....	26
2.3.1. Le Codage PCM.....	26
2.3.2. Le Codage Différentiel DPCM, DM et ADM	28
2.4. Le Codage Fréquentiel.....	29
2.4.1. Le Codage en Sous-Bande	29
2.4.2. Codage Par Transformée.....	30
2.5. La Quantification Vectorielle	32

2.5.1. Principes.....	32
2.5.2. Quantification Par Split.....	34
2.6. Principe d'analyse Par Synthèse.....	35
2.7. Les Codeurs à Bas Débit et à Très Bas Débit	36
2.7.1. Codage à Excitation Multi-Pulse et Regular-Pulse.....	36
2.7.2. Le Codeur CELP.....	37
2.7.3. Les Codeurs LPC à Excitation Mixte ou MELP.....	38
2.7.4. Codeur à Excitation Multibande (MBE).....	39
2.8. Critères Relatifs au Codage	39
2.8.1. Débit de Transmission	40
2.8.2. Délai de Codage.....	40
2.8.3. Qualité de Parole.....	40
2.8.4. Mesures Subjective de La Qualité de Parole	40
2.8.5. Mesures objectives.....	40
2.9. La décomposition en valeurs singulières SVD	42
2.9.1. Principes.....	42
2.9.2. Interprétation Géométrique de la SVD	43
Conclusion	44
<i>Introduction aux FPGA</i>	<i>45</i>
Introduction.....	45
3.1. Architecture adoptée par Xilinx.....	45
3.2. Les circuits configurables	46
<i>Le langage VHDL</i>	<i>53</i>
Introduction.....	53
4.1. La structure d'une description VHDL	54
4.2. Déclaration des bibliothèques.....	55
4.3. Déclaration de l'entité et des entrées/sorties.....	55
4.4. Déclaration de l'architecture – description du fonctionnement.....	56
4.5. Les instructions de base de la logique combinatoire	57
4.6. Les instructions du mode séquentiel	59
4.7. Les types	60
Conclusion	62
<i>La Carte Virtex-II de Xilinx</i>	<i>63</i>
5.1. La carte système Virtex-II	63
5.1.1. Description de la carte système Virtex-II	63
5.1.2. La mémoire DDR.....	64
5.1.3. Génération de l'horloge	65

5.1.4. Le circuit Reset	65
5.1.5. Le port RS 232	66
5.1.6. Le port JTAG	67
5.1.7. Le voltage du banc d'entrée/sortie	68
5.1.8. ISP PROM	68
5.2. Chargement des conceptions.....	69
<i>Programmation et implémentation</i>	<i>70</i>
Introduction.....	70
6.1. Les différentes architectures proposées	70
6.2. Les blocs élémentaires utilisés pour l'implémentation.....	72
6.2.1. Le registre FIFO.....	72
6.2.2. Le diviseur	74
6.2.3. Le multiplieur.....	76
6.2.4. Le multiplieur/accumulateur	77
6.2.5. Le multiplexeur	78
6.3. Les blocs fonctionnels.....	79
6.3.1. Le bloc d'autocorrélation.....	79
6.3.2. Le bloc de Levinson-Durbin	81
CONCLUSION GENERALE.....	88
BIBLIOGRAPHIE.....	90

INTRODUCTION GENERALE

Les techniques de codage de parole ont connu des développements importants dans les dernières décennies. Plusieurs codeurs ont été proposés pour fournir des débits de plus en plus faibles pour une qualité accrue et pour une large gamme d'applications.

D'une manière générale, le but de codage est de réduire le nombre de bits utilisés pour représenter la forme d'onde discrétisée est de maintenir une qualité acceptable pour des raisons de transmission et de stockage. Habituellement le codage avec perte signifie que les formes d'ondes ne peuvent pas être complètement reproduites par le décodeur.

L'objectif principal dans le codage à bas débit n'est pas reproduire un signal qui est physiquement identique au signal parole original mais de reproduire un son qui semble identique, seules les informations utiles à un auditeur humain sont retenues [1].

La plupart des codeurs de parole sont basés sur la prédiction linéaire, leurs efficacités viennent du modèle simple sur lequel ils se basent. Les méthodes de codage prédictif nécessitent une représentation efficace des coefficients du filtre LPC et de son excitation.

Bien que le progrès important qui a été fait dans le codage des paramètres LPC, ce n'est pas encore possible de coder l'excitation à bas débit et maintenir la haute qualité de la voix dans le signal parole reconstituée.

Les transformées orthogonales offrent des méthodes efficaces de réduction du débit nécessaire pour le codage et la transmission du signal. L'efficacité d'un système de codage par transformation dépend du type de transformation et l'allocation de bits au cours du codage. La plupart des systèmes pratiques sont basés sur les approches sous-optimales de la transformation ainsi que l'allocation de bits.

Dans le Chapitre I nous avons étudié la prédiction linéaire et la modélisation de la parole. Ensuite, dans le chapitre II, nous avons étudié les différentes techniques de codage.

Ensuite, dans les chapitres III et IV, à la présentation des FPGA et du leur langage de programmation le VHDL en l'occurrence. Dans le chapitre V, nous avons étudié la carte Xilinx Virtex-II que nous avons utilisée pour implémenter notre algorithme.

Dans le chapitre IV, on a expliquer les différentes étapes de conception, différentes simulations de blocs réalisés, les différentes propositions pour l'implémentation de notre algorithme et les problèmes rencontrés.

Prédiction Linéaire

Introduction

Pour réaliser une analyse efficace du signal parole au niveau acoustique, c'est avantageux d'exploiter la connaissance du processus de la production de parole, cette connaissance est utile pour sélectionner un modèle paramétrique convenable pour la production de parole, une fois le modèle est sélectionné, le rôle des techniques d'analyse de la parole est d'estimer correctement et efficacement les paramètres de ce modèle.

1.1 La Production de la Voix

La parole apparaît physiquement comme une variation de la pression de l'air causée est émise par le système articulatoire. Quand une personne parle, sous le contrôle du système nerveux central qui reçoit en permanence des informations par rétroaction auditive et par les sensations kinesthésiques, les poumons jouent le rôle d'un générateur du système de production de la parole, ils fournissent l'énergie nécessaire à la production du son, en poussant de l'air à travers la trachée-artère, au sommet de celle-ci se trouve le larynx où la pression de l'air est modulée avant d'être appliquée au conduit vocal, composé des cavités pharyngienne et buccale pour la plupart des sons.

La glotte fournit l'entrée avec certaine fréquence du pitch (fondamentale) (F_0). Le conduit vocal travail comme un instrument de musique produisant un son. En fait, les différentes formes du conduit vocal produisent des sons différents. La cavité buccale joue le rôle majeur pour former les différentes formes du conduit vocal [2].

Pour produire des sons nasaux, la cavité nasale est souvent incluse dans le conduit vocal. La cavité nasale est connectée en parallèle avec la cavité buccale. Le conduit vocal simplifié est montré dans la Figure 1.1.

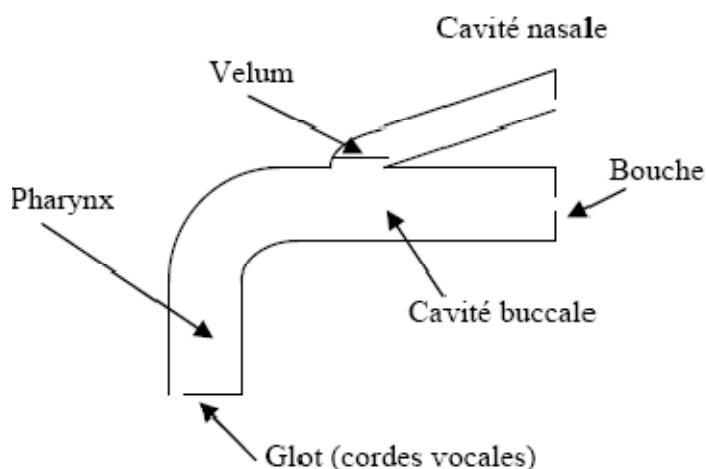


Fig.1.1. Schéma simplifié du conduit vocal

1.1.1. Quelques propriétés importantes de la parole

- Les Fricatives • (s, sh, f, th) sont produites quand le conduit vocal est resserré à quelques emplacements et l'air est forcé à travers ces resserrements.
- Les Plosives • (p, k, t) sont produites quand la fin du conduit vocal est resserré ou est fermée momentanément au moment où la pression atmosphérique a développé, ensuite la pression est libérée brusquement.
- Il y'a approximativement 40 phonèmes (éléments de sons) en anglais (16 voyelles, 24 consonnes).
- Dans la parole normale, 10 à 15 phonèmes sont parlés en une seconde [2].

1.1.2. Propriétés des Sons Voisés et des Sons Non Voisés

- Les sons voisés (ou *sonores*), tels que les voyelles, sont produits par le passage de l'air des poumons à travers la trachée qui met en vibration les cordes vocales. Ce mode, qui représente 80% du temps de phonation, est caractérisé en général par une quasi-périodicité et une énergie élevée.
- Les sons non-voisés (apériodique), comme des consonnes, sont obtenus par resserrement du conduit vocal, et ont habituellement une énergie inférieure aux sons voisés. Les cordes vocales sont écartées et n'entrent pas en vibration. Ces sons sont

considérés comme ayant les mêmes caractéristiques que le bruit. Ceux-ci sont montrés dans Figure 1.2.

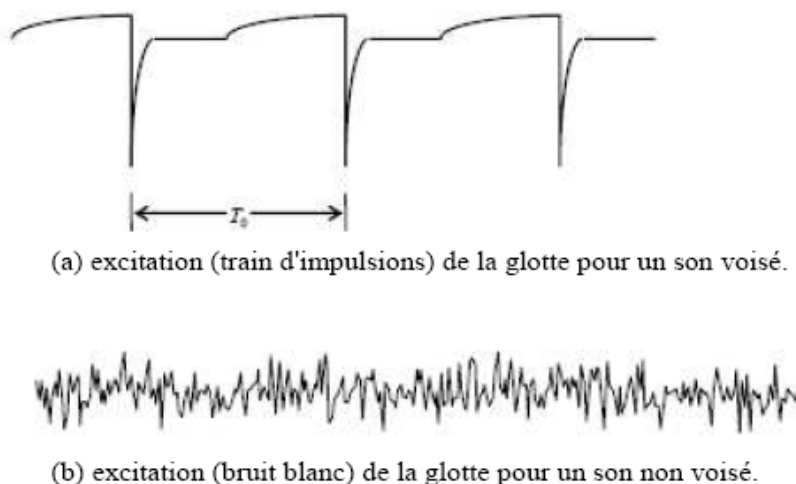


Fig.1.2. Deux types d'excitation pour générer le son (T_0 :période du pitch).

La fréquence de pitch F_0 ($1/T_0$) varie d'une personne à l'autre, pour un petit enfant elle peut aller au maximum à 400 Hz, elle s'étend approximativement de 70 à 250 Hz chez les hommes, de 150 à 400 Hz chez les femmes, elle évolue lentement dans le temps.

Cette pulsation de glotte excite la cavité du conduit vocale et produit une voyelle (son voisé). Quelques caractéristiques d'un son vocalique typique sont montrées ci-dessous [3].

Comme montré dans la Figure 1.3 (a), il y'a au moins quatre sommets (pics) résonnants visibles. Les sommets résonnants se produisent à des fréquences dites formants.

Les fréquences formants F_1 et F_2 sont très distinctes par contre les Formants F_3 et F_4 ne sont pas tout à fait distinctes, ces fréquences sont souvent utilisées pour la reconnaissance de la parole. En observant le spectre de la parole, on peut voir qu'il y'a beaucoup d'harmoniques de F_0 , fréquence du pitch, à cause de la richesse en sommets harmonieux il n'est pas facile d'extraire les formants du spectre. D'un autre côté, en entrant un bruit - comme signal d'excitation au conduit vocal, les sons non voisés tels que les Plosives et les Fricatives sont produits [3].

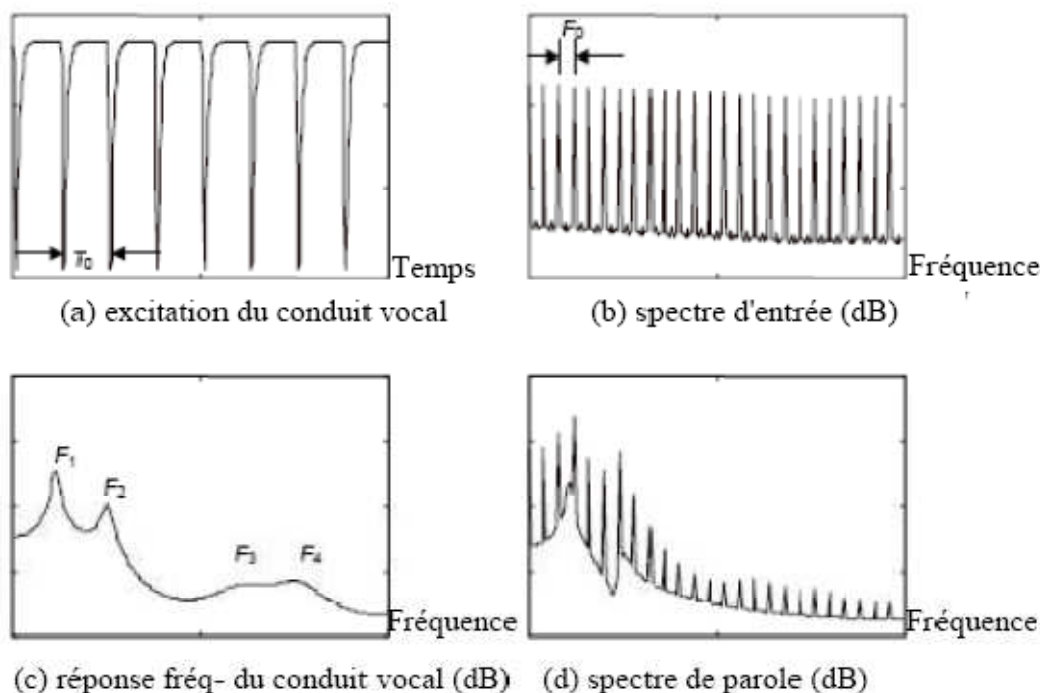


Fig1.3. Caractéristiques d'un son vocalique typique

1.1.3. Modélisation du Processus de Production de la Parole

Le processus de production de la parole peut être modélisé par le système montré dans la Figure 1.4. Un signal voisé peut être modélisé par le passage d'un train d'impulsions $e(n)$ à travers un filtre numérique récuratif de type *tous pôles*. On montre que cette modélisation reste valable dans le cas du son non-voisé, à condition que $e(n)$ soit cette fois un bruit blanc. Il est souvent appelé *modèle autorégressif*. Les paramètres du modèle AR sont : la période du train d'impulsions (sons voisés uniquement), la décision Voisé/Non Voisé (V/NV), le gain G , et les coefficients du filtre $1/A(z)$, appelé *filtre de synthèse* [4].

Dans les deux cas (sons voisés ou non voisés), le facteur du gain contrôle l'intensité de l'excitation. Dans le domaine temporel le signal parole est la convolution de l'excitation et la réponse impulsionnelle du système linéaire de génération de parole. Dans le domaine spectral, le spectre de la parole généré est le produit simple des spectres de l'excitation et du système linéaire.

Les différents sons sont produits par ce modèle en changeant la source de l'excitation et les configurations du système linéaires.

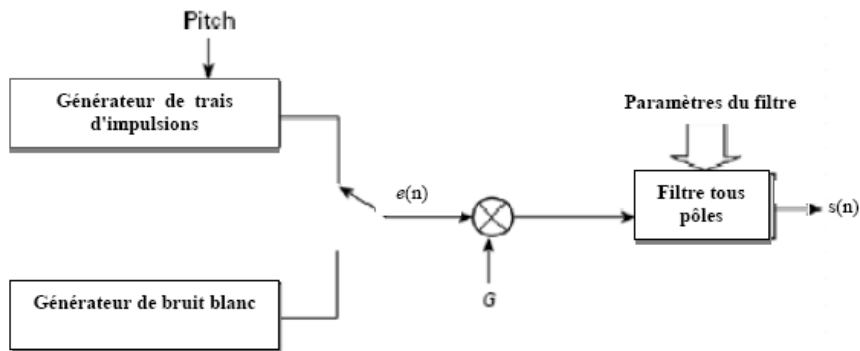


Fig. 1.4. Modèle simplifié de production de la parole: G est le gain, $e(n)$ l'excitation, $s(n)$ le signal parole.

1.2. La Prédiction Linéaire

1.2.1. Principes

La prédiction linéaire est l'une des techniques fondamentales pour enlever la redondance d'un signal. Elle consiste à prédire la valeur d'un échantillon $s(n)$ à partir d'une combinaison linéaire de M échantillons passés $s(n - i)$. La séquence estimée $\hat{s}(n)$ est donnée par [2] [5]:

$$\hat{s}(n) = a_1 s(n-1) + a_2 s(n-2) + \dots + a_M s(n-M) = \sum_{i=1}^M a_i s(n-i) \quad (1.1)$$

La différence entre l'échantillon actuel et l'échantillon prédit est appelée erreur de prédiction, elle est exprimée par :

$$e(n) = s(n) - \hat{s}(n) = s(n) - \sum_{i=1}^M a_i s(n-i) \quad (1.2)$$

Le problème revient à déterminer les coefficients de prédiction (a_i).

1.2.2. Minimisation d'Erreur de Prédiction

Les "meilleurs" coefficients de prédiction sont obtenus par la minimisation de la valeur quadratique moyenne de l'erreur de prédiction. On cherche donc les coefficients a_i qui minimisent la puissance de l'erreur de prédiction définie par:

$$E = \sum_n e^2(n) = \left(s(n) - \sum_{i=1}^M a_i s(n-i) \right)^2 \quad (1.3)$$

Si le signal $s(n)$ est supposé égal à zéro pour $n < 0$ et $n > N$ (par exemple, en le multipliant par une fenêtre de durée finie), la minimisation de l'erreur ($\frac{\partial E}{\partial a_i} = 0, 1 \leq i \leq M$), conduit au système d'équations linéaires de Yule-Walker [5]:

$$\sum_{k=1}^M a_k R(|i-k|) = R(i) \quad 1 \leq i \leq M \quad M : \text{ordre de prédiction} \quad (1.4)$$

Où

$$R(k) = \sum_{n=0}^{N-1-k} s(n)s(n+k), \text{ est la fonction d'autocorrélation du signal parole } s(n).$$

L'équation (4) correspond à l'équation matricielle :

$$\begin{bmatrix} R_0 & R_1 & R_2 & \cdots & R_{M-1} \\ R_1 & R_0 & R_1 & & R_{M-2} \\ R_2 & R_1 & R_0 & & R_{M-3} \\ \vdots & & & \ddots & \vdots \\ R_{M-1} & R_{M-2} & R_{M-3} & \cdots & R_0 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ \vdots \\ a_M \end{bmatrix} = \begin{bmatrix} R_1 \\ R_2 \\ R_3 \\ \vdots \\ R_M \end{bmatrix} \quad (1.5)$$

La matrice d'autocorrélation est une matrice de Toeplitz symétrique. La résolution de ce système est couramment réalisée par l'algorithme de Levinson-Durbin. C'est un algorithme qui résout le système en un nombre restreint d'opérations sans calculer l'inverse de la matrice d'autocorrélation [2].

1.2.3. Algorithme de Levinson-Durbin

Les valeurs initiales

$$E_0 = r(0)$$

$$a_{11} = \kappa_1 = r(1) / E_0$$

$$E_1 = E_0(1 - \kappa_1)^2.$$

avec $m \geq 2$, la récursion suivante est exécutée

$$(i) \quad q_m = r(m) - \sum_{i=1}^{M-1} a_{i(m-1)} r(m-i)$$

$$(ii) \quad k_m = q_m / E_{m-1}$$

$$(iii) \quad a_{mm} = k_m$$

$$(iv) \quad a_{im} = a_{i(m-1)} - k_m a_{(m-i)(m-1)} \quad \text{pour } i = 1, \dots, m-1$$

$$(v) \quad E_m = E_{m-1}(1 - \kappa_m)^2$$

(vi) si $m < M$, augmenter m à $(m+1)$ et aller à (i).

si $m = M$, arrêter.

Une fois les coefficients de prédiction sont calculés, l'équation (1.2) peut être utilisée pour trouver la séquence d'erreurs Figure 1.5.

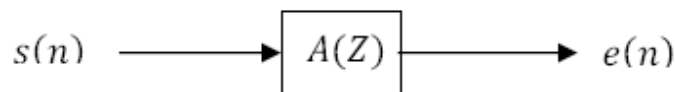


Fig.1.5. Filtre d'analyse

La fonction de transfert du filtre d'analyse (Filtre inverse) est donnée par :

$$A(z) = 1 - \sum_{i=1}^M a_i z^{-i} \quad (1.6)$$

La Figure 1.6 représente un exemple d'erreur de prédiction, aussi nommée excitation ou signal résiduel, calculée à partir du filtre d'analyse.

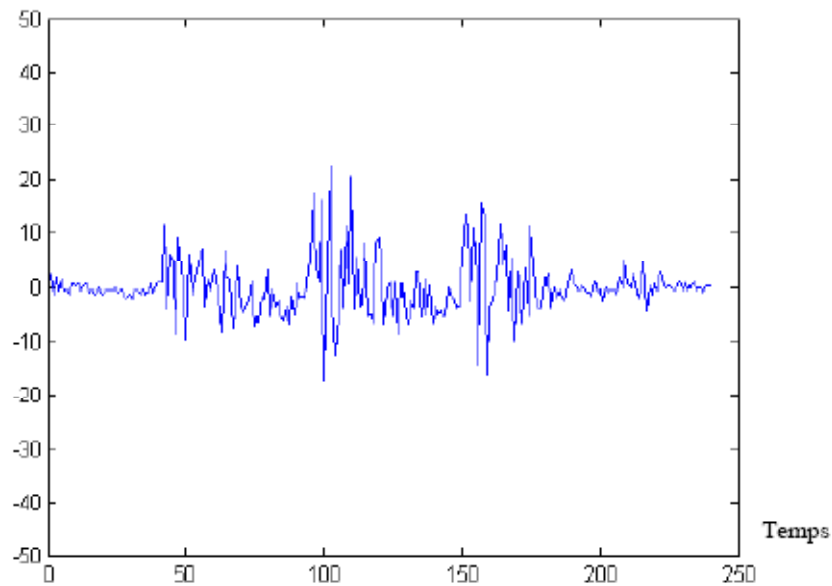


Fig.1.6. La séquence du signal résiduel

Les échantillons $s(n)$ ayant une variance plus faible que le signal parole, ils nécessitent moins de bits pour être représentés.

Le signal parole peut être reconstitué à partir des coefficients de prédiction et du signal résiduel en utilisant le filtre de synthèse qui modélise le conduit vocal du locuteur Figure 1.7.

On peut réécrire l'équation (1.2) sous la forme :

$$s(n) = \sum_{i=1}^M a_i s(n-i) + e(n) \quad (1.7)$$

L'entrée et la sortie de ce filtre sont respectivement $s(n)$ et $e(n)$.

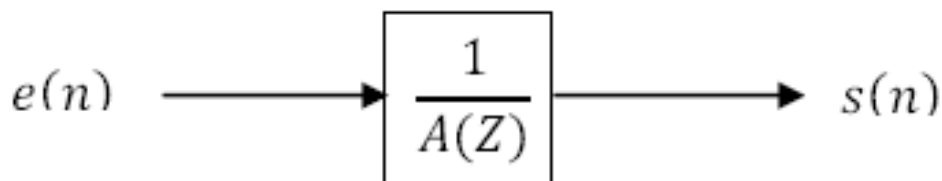


Fig.1.7. Filtre de synthèse

La fonction de transfert du filtre de synthèse (Filtre LPC) est donnée par :

$$H(z) = \frac{1}{1 - \sum_{i=1}^M a_i z^{-i}} \quad (1.8)$$

La Figure 1.8 représente l'enveloppe spectrale de 30ms de la voyelle / uh /, calculé par l'utilisation de la méthode d'autocorrélation avec $M=10$, la fenêtre de Hamming est utilisée.

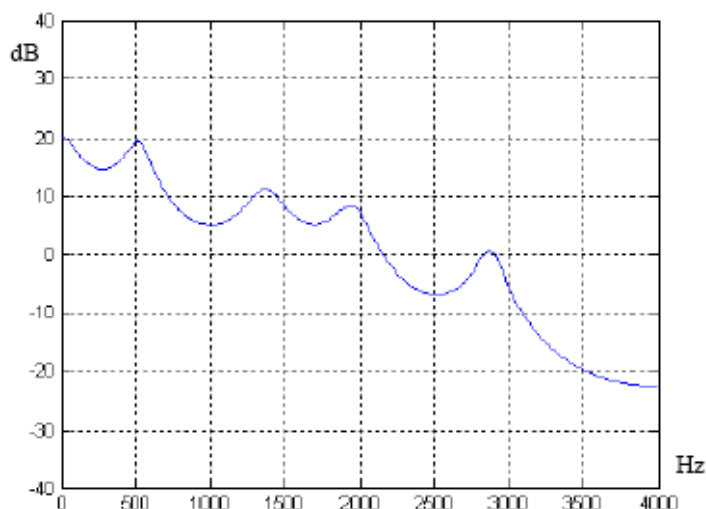


Fig.1.8. Spectre d'amplitude du filtre LPC

Le choix de l'ordre de prédiction, M , résulte d'un compromis. Il doit être suffisamment élevé pour reproduire correctement la structure "formantique" du signal de parole: un ordre de 8 est nécessaire pour créer 4 pics dans le spectre et on a vu que le signal de parole comporte généralement 4 "formants". Inversement, l'ordre doit être le plus faible possible pour économiser le débit. On choisit donc M compris entre 8 et 16.

1.3. Analyse de Fourier à Court Terme

L'analyse de Fourier est la technique traditionnelle pour calculer le spectre d'amplitude et de phase du signal parole. La transformée de Fourier standard exige que le signal soit disponible pour tout le temps (c-à-d de moins l'infini à plus l'infini). Puisque le signal parole est de nature non stationnaire, il devient nécessaire d'utiliser une analyse de Fourier à court terme, on pondérant les échantillons de parole par une fenêtre de pondération (souvent une fenêtre de Hamming) Figure 1.9, et en effectuant une transformée de Fourier sur ces échantillons, la DFT (transformée de Fourier discrète) est calculée par l'algorithme (FFT) [2].

$$w(n) = \begin{cases} 0,54 - 0,46 \cos\left(\frac{2\pi n}{N}\right), & 0 \leq n \leq N \\ 0 & \text{ailleurs} \end{cases} \quad (1.9)$$

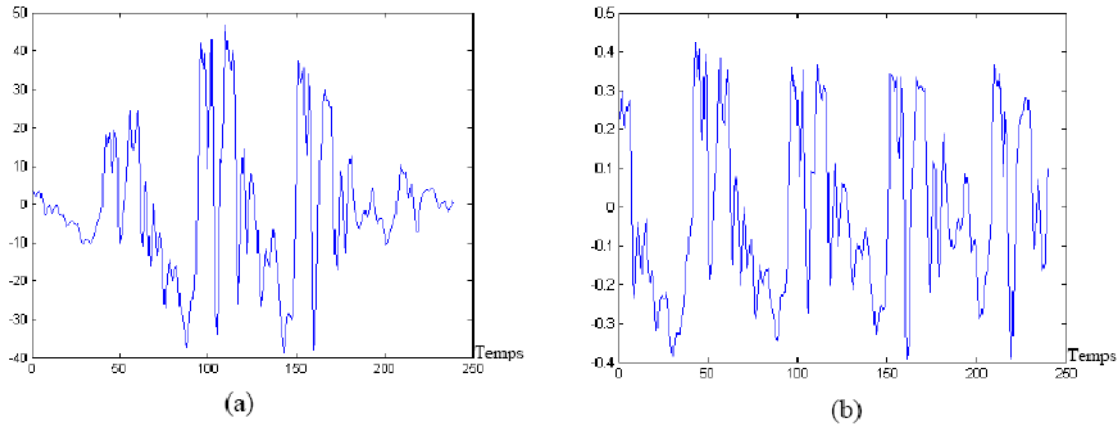


Fig.1.9. 240 échantillons de signal parole "uh ", (a) : fenêtre de Hamming, (b) : fenêtre carrée.

1.4. Prédiction Linéaire rétrograde (Backward) et Filtre en Treillis

La prédiction linéaire peut aussi être appliquée en Backward. Dans ce cas $s(n-M)$ est prédit à partir de la séquence : $s(n) s(n-1) \dots s(n-M+1)$:

$$\hat{s}(n-M) = \sum_{i=1}^M b_i s(n-i+1) \quad (1.10)$$

De la même façon que (1.2), l'erreur de prédiction Backward est définie comme suit :

$$\beta(n) = s(n-M) - \sum_{i=1}^M b_i s(n-i+1) \quad (1.11)$$

La minimisation de cette erreur par rapport aux coefficients de prédiction donne le même ensemble d'équations (1.5).

Le filtre en treillis est une forme de représentation très utile dans le traitement numérique de parole, cette forme d'implémentation (Algorithme équivalent à l'algorithme de Levinson-Durbin) est basée sur les coefficients de réflexion, aussi nommés coefficients PARCOR (corrélation partielle) Figure 1.10, [2].

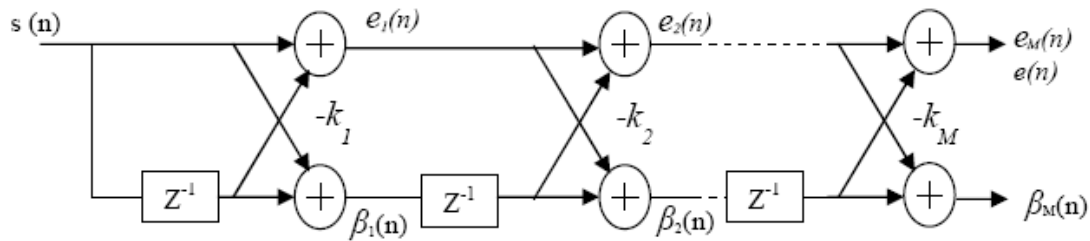


Fig.1.10. Implémentation en treillis par l'utilisation de coefficients de réflexions

Cette structure correspond à :

$$\begin{aligned}
 e_m(n) &= e_{m-1}(n) - k_m \beta_{m-1}(n-1) \quad \text{pour } m = 1, \dots, M \\
 \beta_m(n) &= -k_m e_{m-1}(n) + \beta_{m-1}(n-1) \\
 \text{avec : } e_0(n) &= \beta_0(n) = s(n)
 \end{aligned}
 \tag{1.12}$$

Si nous évaluons ces équations récursives jusqu'à l'ordre M , nous trouverons que la sortie de la cellule M du filtre en treillis est identique à la sortie M du filtre FIR de la forme directe. Pour illustrer ceci on considère un exemple où M égal à deux.

Nous évaluons les équations (10) deux fois, on obtient:

$$\begin{aligned}
 e_2(n) &= s(n) - (k_1 - k_1 k_2) s(n-1) - k_2 s(n-2) \\
 \beta_2(n) &= -k_2 s(n) - (k_1 - k_1 k_2) s(n-1) + s(n-2)
 \end{aligned}
 \tag{1.13}$$

On constate qu'un filtre en treillis à deux cellules est équivalent à un filtre FIR d'ordre deux de forme directe avec :

$$a_2(1) = (k_1 - k_1 k_2), \quad a_2(2) = k_2 \quad \text{et} \quad b_2(0) = k_2, \quad b_2(1) = (k_1 - k_1 k_2)$$

Les coefficients dans le prédicteur linéaire Backward sont identiques aux coefficients du prédicteur linéaire Forward dans un ordre renversé, $b_M(i) = a_M(M - i) \quad 1 \leq i \leq M$.

Aussi de la relation $a_m(m) = k_m \quad \text{pour } m = 1, 2, \dots, M$ (Algorithme de Levinson -Durbin), on peut montrer que :

$$B_M(Z) = Z^{-(M+1)} A_M(Z^{-1}) \tag{1.14}$$

Cette relation implique que les zéros de $B_M(z)$ sont simplement les conjugués réciproques des zéros de $A_M(z)$. Si on prend la transformée en Z des équations récursives (1.13) et diviser les deux résultats par $s(z)$ on obtient :

$$\begin{aligned} A_m(Z) &= A_{m-1}(Z) - k_m Z^{-1} B_{m-1}(Z) & 1 \leq m \leq M \\ B_m(Z) &= -k_m A_{m-1}(Z) + Z^{-1} B_{m-1}(Z) & 1 \leq m \leq M \end{aligned} \quad (1.15)$$

Avec : $A_0(Z) = B_0(Z) = 1$

Donc ce filtre en treillis est complètement décrit par l'équation matricielle :

$$\begin{bmatrix} A_m(Z) \\ B_m(Z) \end{bmatrix} = \begin{bmatrix} 1 & -k_m Z^{-1} \\ -k_m & Z^{-1} \end{bmatrix} \begin{bmatrix} A_{m-1}(Z) \\ B_{m-1}(Z) \end{bmatrix} \quad (1.16)$$

Cette relation nous permet de calculer les coefficients LPC à partir des coefficients de réflexion k_i et vice versa. L'algorithme récursif pour calculer les coefficients LPC à partir des coefficients de réflexion est donné par :

Pour $m = 1$ à M faire

$$\begin{cases} a_m(m) = k_m \\ \text{Pour } i = 1 \text{ à } m-1 \text{ faire} \\ a_m(i) = a_{m-1}(i) - k_m a_{m-1}(m-i) \end{cases} \quad (1.17)$$

De la même façon, on peut mettre l'équation (1.15) sous la forme :

$$\begin{bmatrix} A_{m-1}(Z) \\ B_{m-1}(Z) \end{bmatrix} = \frac{1}{1 - k_m^2} \begin{bmatrix} 1 & k_m \\ k_m Z & Z \end{bmatrix} \begin{bmatrix} A_m(Z) \\ B_m(Z) \end{bmatrix} \quad (1.18)$$

Alors, une formule pour déterminer les coefficients a_i peut être dérivée de la relation polynomiale d'équation (1.14)

$$A_{m-1}(Z) = \frac{A_m(Z) + k_m Z^{-(m+1)} A_m(Z^{-1})}{1 - k_m^2} \quad m = M, \dots, 1 \quad (1.19)$$

Les coefficients de réflexions peuvent être calculés à partir des paramètres LPC par l'algorithme suivant :

Pour $m = M$ à 1 faire

$$\begin{cases} a_m(m) = k_m \\ \text{Pour } i = 1 \text{ à } m-1 \text{ faire} \\ a_{m-1}(i) = \frac{a_m(i) + k_m a_m(m-i+1)}{1 - k_m^2} \end{cases} \quad (1.20)$$

1.5. Représentation des Coefficients de Prédiction

Dans les applications de codage de parole, il est nécessaire de quantifier les paramètres LPC avec un minimum de distorsion. Aussi, il est exigé que le filtre tout pôles reste stable après la quantification de ces paramètres. La quantification directe des coefficients LPC n'est pas conseillée parce que les petites erreurs de quantification dans ces coefficients peuvent produire des erreurs spectrales relativement grandes, et peuvent causer aussi une instabilité du filtre $H(z)$. Par conséquent, c'est nécessaire d'utiliser un grand nombre de bits pour accomplir une bonne quantification des paramètres LPC eux-mêmes. Par l'utilisation de 6 bits/coefficient (c.-à-d., 60 bits/trame) pour une quantification scalaire des coefficients LPC dans la base de données FM, 25.5% des filtres sont instables, et la distorsion moyenne spectrale est de 1.83 dB [6].

1.5.1. Les Coefficients de Réflexions

Les coefficients de réflexions (RCs) peuvent être obtenus des coefficients LPC par l'utilisation de l'algorithme de Levinson -Durbin. Ces coefficients ont deux avantages majeurs sur les coefficients LPC :

- i. ils sont moins sensibles spectralement à la quantification, et
- ii. la stabilité du filtre tout pôles peut être assurée en gardant chaque coefficient dans la gamme de -1 à +1 pendant le processus de quantification [6].

1.5.2. Les Coefficients (LARs) et (ASRC)

Bien que les coefficients de réflexion sont spectralement moins sensibles à la distorsion de quantification que Les coefficients LPC, la distribution statistique de ces coefficients ne ressemble pas à une distribution uniforme. Cependant, cet inconvénient peut être vaincu par l'usage d'une transformation non-linéaire appropriée qui étend la région proche de $k_i = 1$, où k_i est le $i^{\text{ème}}$ coefficient de réflexion :

- LAR (Log Area Ratio), ou rapport d'aires logarithmiques, sont déduits des coefficients RCs après transformation non-linéaire, et sont utilisés pour leurs bonnes propriétés de quantification linéaire, ils ont rapport avec les fonctions des surfaces du tube vocale.

$$LAR_i = \ln\left(\frac{1+k_i}{1-k_i}\right) \quad 1 \leq i \leq M$$

(1.21)

- Une deuxième transformation est la transformation en sinus inverse (ASRC)

$$J_i = \sin^{-1} k_i \quad (1.22)$$

Les représentations LAR et ASRC exigent approximativement 32 bits/trame pour fournir une distorsion spectrale moyenne de 1 dB [6].

1.5.3. Les Fréquences de Raies Spectrales

Parmi les représentations équivalentes aux coefficients de prédiction, Les Fréquences de Raies Spectrales (Paramètres LSF) ont les meilleures caractéristiques de quantification. Une interprétation physique des paramètres LSF permet d'établir un lien avec les formants.

A partir du polynôme $A(z)$ d'ordre M , on construit deux nouveaux polynômes d'ordre $M+1$.

$$\begin{aligned} P(Z) &= A(Z) + Z^{-(M+1)} A(Z^{-1}) \\ &= 1 + (a_1 - a_M)Z^{-1} + (a_2 - a_{M-1})Z^{-2} + \dots + (a_M - a_1)Z^{-M} + Z^{-M+1} \end{aligned} \quad (1.23)$$

$$\begin{aligned} Q(Z) &= A(Z) - Z^{-(M+1)} A(Z^{-1}) \\ &= 1 + (a_1 + a_M)Z^{-1} + (a_2 + a_{M-1})Z^{-2} + \dots + (a_M + a_1)Z^{-M} + Z^{-M+1} \end{aligned} \quad (1.24)$$

Donc :

$$A(Z) = \frac{P(Z) + Q(Z)}{2} \quad (1.25)$$

On peut montrer que ces deux polynômes ont les propriétés suivantes:

- Le polynôme $P(z)$ est un polynôme symétrique. Le polynôme $Q(z)$ est un polynôme antisymétrique.
- Si toutes les racines de $A(z)$ sont à l'intérieur du cercle unité toutes les racines de $P(z)$ et de $Q(z)$ sont sur le cercle unité.
- Les racines de $P(z)$ et de $Q(z)$ apparaissent de façon alternée sur le cercle unité.

Si M est paire, $P(z)$ a pour racine évidente -1 et $Q(z)$ a pour racine évidente $+1$.

L'expression des paires de racines complexes conjuguées des polynômes $P(z)$ et $Q(z)$ est donnée par :

$$(1 - e^{j\theta_i} Z^{-1})(1 - e^{-j\theta_i} Z^{-1}) = 1 - 2Z^{-1} \cos \theta_i + Z^{-2} \quad (1.26)$$

Avec θ_i étant la fréquence de la raie spectrale. $P(z)$ et $Q(z)$ peuvent être factorisés comme suit [3] :

$$\left\{ \begin{array}{l} P(Z) = (1 + Z^{-1})P'(Z) \\ \quad = (1 + Z^{-1}) \prod_{i=2,4,\dots,M} (1 - 2Z^{-1} \cos \theta_i + Z^{-2}) \\ \\ Q(Z) = (1 - Z^{-1})Q'(Z) \\ \quad = (1 - Z^{-1}) \prod_{i=2,4,\dots,M} (1 - 2Z^{-1} \cos \theta_i + Z^{-2}) \end{array} \right. \quad (1.27)$$

Les paramètres θ_i sont rangés en ordre décroissant :

$$0 < \theta_1 < \theta_2 < \dots < \theta_{M-1} < \theta_M < \pi$$

Le carré de la réponse en amplitude du filtre de synthèse est donné par:

$$\begin{aligned} |H(\theta)|^2 &= \frac{1}{|A(\theta)|^2} \\ &= 2^2 |P(\theta) + Q(\theta)|^{-2} \\ &= 2^{-M} \left[\cos^2 \frac{\theta}{2} \prod_{i=1,3,\dots,M-1} (\cos \theta - \cos \theta_i)^2 + \sin^2 \frac{\theta}{2} \prod_{i=2,4,\dots,M} (\cos \theta - \cos \theta_i)^2 \right]^{-1} \end{aligned} \quad (1.28)$$

Le premier terme à l'intérieur des parenthèses dans l'équation (1.28) se rapproche de 0 quand θ ou un des θ_i ($i = 1, 3, \dots, M-1$) se rapproche de π . Le deuxième terme se rapproche de 0 quand θ ou un des θ_i ($i = 2, 4, \dots, M$) se rapproche de 0. Par conséquent, lorsque deux paramètres LSP θ_i et θ_j se rapprochent, le gain de $H(z)$ devient grand et une résonance se produit. C'est pourquoi, le spectre de parole est relié directement avec les paramètres LSF [3]. Le spectre d'amplitude des filtres $P(z)$ et $Q(z)$ est montré sur la Figure 1.11.

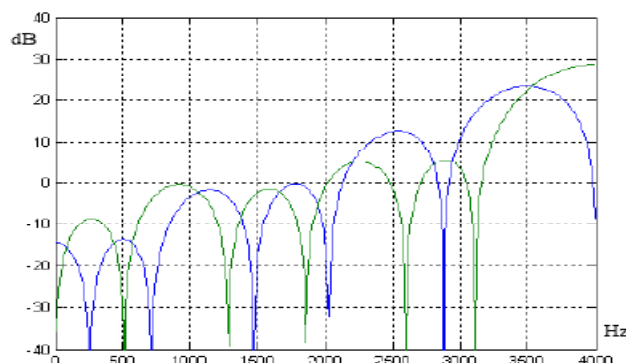


Fig.1.11. Le spectre d'amplitude de $P(z)$ et $Q(z)$ avec $M=10$.

S'il y'a une erreur causée par la quantification d'un paramètres LSF, l'erreur est localisée.

On peut transformer les LSP (positions angulaires) en LSF dans le domaine des fréquences normalisées $f_i (0 \leq f_i \leq 0.5)$, $f_i = \frac{\theta_i}{2\pi}$, ou en fréquences réelles $F = \frac{\theta_i}{2\pi} F_e$ où F_e est la fréquence d'échantillonnage ($0 \leq F \leq F_e / 2$).

• Algorithme de Calcul

Les paramètres LSF peuvent être trouvés par la procédure suivante [3] :

1. Trouver les coefficients de prédiction linéaires LPC.
2. Former $P(z)$ et $Q(z)$.
3. Estimer la réponse en amplitude de $P(z)$ et $Q(z)$. FFT ou DFT peut être utilisée pour ceci.
4. Les fréquences dans lesquelles les minima locaux se produisent sont les paramètres LSF.

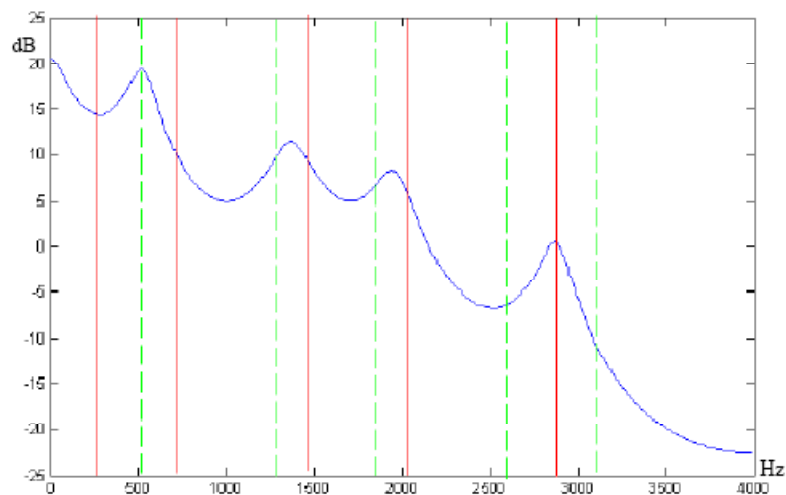


Fig.1.12. Spectre du filtre LPC et les positions des paramètres LSF correspondants, pour une trame de 30 ms de la voyelle /uh/. Les lignes continues représentent les zéros de $P'(z)$, les lignes en points représentent les zéros de $Q'(z)$

La propriété $0 < \theta_1 < \theta_2 < \dots < \theta_{M-1} < \theta_M < \pi$, permet de réduire considérablement la complexité de recherche des paramètres LSF.

1.6. Conversion LSP-LPC

Une fois quantifiés, les coefficients LSF sont reconvertis en coefficients LPC, a_i . Pour cela, on détermine les coefficients des polynômes $P'(z)$ et $Q'(z)$ par expansion des équations (1.32), sur la base des coefficients LSF quantifiés [7].

$$P'(Z) = \frac{P(Z)}{1+Z^{-1}}, \quad Q'(Z) = \frac{Q(Z)}{1-Z^{-1}} \quad (1.31)$$

A partir des coefficients q_i , on calcule les coefficients $p'(i)$, $i=1, \dots, M/2$, par la relation de récurrence suivante :

Pour $i = 1$ à 5

$$p'(i) = -2q_{i-1} p'(i-1) + 2p'(i-1) \quad \text{avec } q_i = \cos \theta_i$$

Pour $j = i-1$ à 1

$$p'(j) = p'(j) - 2q_{2i-1} p'(j-1) + p'(j-1) \quad (1.32)$$

Fin i

Fin j

Avec les conditions initiales $p'(0) = 1$ et $p'(-1) = 0$. Les coefficients $q'(i)$ sont calculés de la même manière, avec remplacement des q_{2i-1} par q_{2i} .

Une fois les coefficients $p'(i)$ et $q'(i)$ calculés, on multiplie les polynômes $P'(z)$ et $Q'(z)$ par $(1+Z^{-1})$ et $(1-Z^{-1})$ respectivement, pour obtenir les polynômes $P(z)$ et $Q(z)$, qui donnent les coefficients suivants :

$$\begin{aligned} p(i) &= p'(i) + p'(i-1) & i &= 1, \dots, 5 \\ q(i) &= q'(i) + q'(i-1) & i &= 1, \dots, 5 \end{aligned} \quad (1.33)$$

Finalement, on trouve les coefficients LPC à partir des coefficients $p(i)$ et $q(i)$ comme suit :

$$a_i = \begin{cases} 0.5 p(i) + 0.5 q(i) & i = 1, \dots, 5 \\ 0.5 p(11-i) + 0.5 q(11-i) & i = 6, \dots, 10 \end{cases} \quad (1.34)$$

Cette équation est directement issue de la relation : $A(Z) = \frac{P(Z) + Q(Z)}{2}$.

1.7. Détecteur de Voisement

Le rôle de détecteur de voisement est de classer les trames de parole voisée /non voisée. La mesure de l'aplatissement du spectre, l'énergie, et le taux de passage par zéro sont les méthodes les plus utilisées.

1.7.1. Mesure d'Aplatissement du Spectre

Une des méthodes pour décider si la trame de parole est voisée ou non est la mesure d'aplatissement du spectre qui fait usage de la propriété que le spectre de bruit pur est supposé être plat. En d'autres termes, le spectre du segment non voisé est plat et le spectre du segment voisé est moins plat, cette mesure (SFM) est donné par:

$$SFM = \frac{\left(\prod_{k=0}^{N-1} S_j(k) \right)^{\frac{1}{N}}}{\frac{1}{N} \prod_{k=0}^{N-1} S_j(k)} ; S(k) \text{ est TF de la séquence } s(n) \quad (1.35)$$

Cette mesure varie de 0.9 pour un bruit blanc à 0.1 pour un signal sonore. Le seuil est habituellement choisi entre 0.35 à 0.48 [2].

1.7.2. Énergie et Taux de Passage par Zéro

L'énergie de la $j^{\text{ème}}$ trame du signal parole est $E = \sum_{n=0}^{N-1} s_j^2(n)$ où $s_j(n)$ est l'échantillon n dans la trame j . Habituellement l'énergie d'une trame voisée est plus grande que celle d'une trame non voisée.

Le taux de passage par zéro est obtenu en comptant le changement de signe dans les échantillons successifs de la parole. Le ZCR (Zero-crossing rate) du son voisé est inférieur à celui du son non voisé.

1.8. Détection du pitch

La détection du pitch a été pour longtemps une partie active de recherche à cause de son importance dans la synthèse de parole. Il y'a plusieurs algorithmes de détection de pitch, qui ont leurs propres avantages et inconvénients. En général, quatre classes de méthodes sont utilisées communément. Elles sont les méthodes du cepstre, méthodes de corrélation, méthodes temporelles et méthodes fréquentielles [2].

Conclusion

Dans ce chapitre les fondements théoriques et pratiques pour la modélisation du système de production de la parole, et principalement l'estimation des paramètres LPC, les différentes représentations de ces derniers a été faite, les techniques de codage de la parole font l'objet du chapitre suivant.

Techniques de codage de la parole

Introduction

Dans ce chapitre nous allons présenter brièvement les notions de base de la théorie de l'information relatives au codage source et les techniques du codage de la parole. Le nombre de codeurs différents étant très vaste, nous ne tenterons pas de récapituler tous les codeurs existants ou de les présenter en détail. Nous allons nous borner aux techniques les plus répandues. On conclura le chapitre par les différents critères couramment utilisés pour juger et classer les méthodes de codage.

2.1. Mesure de L'information et Entropie de la Source

Le rôle du codeur source est d'effectuer la compression ou la réduction du débit binaire en enlevant la redondance. Le codage de la source nécessite une description quantitative de la source discrète et en particulier, du contenu et du débit de l'information à transmettre. Ces renseignements sont en général, fournis respectivement par l'entropie de la source et par le débit d'entropie. Considérons une source discrète possédant un alphabet Q et M symboles différents α_j ($j = 1, 2, \dots, M$) de probabilité d'apparition $p(\alpha_j)$. Cette probabilité est appelée probabilité a-priori afin de la distinguer de la probabilité a-posteriori qui correspond à la probabilité que le message reçu au récepteur soit correct. On a donc :

$$p(\alpha_j) \geq 0 \quad j = 1, 2, \dots, M \quad \text{et} \quad \sum_{j=1}^M p(\alpha_j) = 1 \quad (2.1)$$

Plus la probabilité d'apparition d'un symbole est faible, moins on s'attend à son apparition. Ce symbole contient donc plus d'informations que lorsqu'il s'agit d'un symbole couramment émis. Cet argument ainsi que d'autres du même type conduisent à la définition usuelle d'auto-information d'une source discrète, valant pour chaque symbole α_j :

$$I(\alpha_j) = \log\left(\frac{1}{p(\alpha_j)}\right) \quad (2.2)$$

Ce résultat correspond à l'information transmise sous l'hypothèse qu'à la sortie de la source, le symbole α_j soit fixé dans le temps. L'unité de cette information dépend de la base; on quantifie donc l'information en bits [8].

Tant qu'il s'agit de systèmes d'information, il est plus utile de décrire la source par l'information moyenne qu'elle génère par symbole plutôt que par l'information instantanée par symbole. Si l'on suppose une source sans mémoire (i.e. tous les symboles sont statiquement indépendants), l'information moyenne par symbole est facile à calculer : le $j^{\text{ème}}$ symbole apparaît environ $Np(\alpha_j)$ fois dans un long message de N symboles et l'information totale vaut approximativement :

$$I(Q)_{\text{totale}} = \sum_{j=1}^M Np(\alpha_j) \log \left(\frac{1}{p(\alpha_j)} \right) \text{ bits} \quad (2.3)$$

L'entropie H de la source est l'information moyenne statistique par symbole :

$$H(Q) = \sum_{j=1}^M p(\alpha_j) \log \left(\frac{1}{p(\alpha_j)} \right) \frac{\text{bits}}{\text{symbole}} \quad (2.4)$$

Cependant, pour une source dont les symboles sont équiprobables ($p(\alpha_j) = 1/M$), montre qu'on obtient dans ce cas l'information moyenne par symbole :

$$H_{\text{max}} = \sum_{j=1}^M p(\alpha_j) \log(M) = \log(M) \quad (2.5)$$

2.2. La Quantification Scalaire

La quantification est une opération qui consiste à arrondir une valeur d'entrée en lui associant une autre valeur parmi un nombre fini de valeurs possibles. Cette opération est indispensable si l'on veut transmettre l'entrée sous forme numérique. Très souvent, on inclut dans la quantification le codage qui associe un indice à chacune des valeurs possibles de l'entrée à quantifier. La quantification établit une relation subjective entre l'ensemble où la grandeur à quantifier prend ses valeurs et une partie de l'ensemble des entiers naturels. Donc à chaque valeur x de l'entrée à quantifier on fait correspondre un indice $i = I(x)$ dont la valeur en binaire sera transmise à la place de x . A la réception, on estime la valeur x par $Ci = Q(x)$ qui ne dépend que de i , $Q(x)$ étant la valeur quantifiée de x .

Soit x une quantité scalaire continue, un coefficient transformé par exemple. Pour représenter x avec un nombre fini de bits, on doit utiliser un nombre limité de niveaux de reconstruction ou de quantification L .

Le processus d'attribution d'une valeur spécifique de x à un seul des L niveaux est appelé quantification d'amplitude, ou quantification tout court. Si chaque scalaire représentant une amplitude est quantifié indépendamment, la procédure est dite quantification scalaire.

Si plusieurs scalaires sont quantifiés conjointement, la procédure est appelée quantification vectorielle [8].

Soit la valeur quantifiée de \hat{x} , on peut exprimer \hat{x} par :

$$\hat{x} = Q(x) = r_i \quad q_{i-1} \leq x \leq q_i \quad (2.6)$$

où Q représente l'opération de quantification, r_i pour $1 < i < L$ représente L niveaux reconstruits, et q_i pour $0 \leq i \leq L$ représente $L+1$ niveaux de décision.

On peut aussi exprimer \hat{x} par $\hat{x} = Q(x) = x + e_Q$ où $e_Q = x - \hat{x}$ est l'erreur de quantification. e_Q est aussi appelée bruit de quantification. La quantité e_Q^2 peut être considérée comme un cas particulier de la mesure de distorsion $d(x, \hat{x})$.

Les niveaux de décision sont souvent déterminés en minimisant un critère d'erreur basé sur $d(x, \hat{x})$, telle que la moyenne de distorsion D donnée par :

$$D = E [d(x, \hat{x})] = \int_{x_0=-\infty}^{\infty} d(x_0, \hat{x}) p_x(x_0) dx_0 \quad (2.7)$$

$$= \sum_{i=1}^L \int_{x_0=q_{i-1}}^{q_i} d(x_0, \hat{x}) p_x(x_0) dx_0$$

où \hat{x} = signal quantifié
 $p_x(x_0)$ = densité de probabilité de x
 L = nombre de niveaux de quantification
 q_i = bornes de quantification
 $d(x, \hat{x})$ = mesure de distorsion

Supposons que nous avons N scalaires x_i pour $1 \leq i \leq N$, et chaque scalaire x_i est quantifié à L_i niveaux de reconstruction. Si L_i peut être exprimé en puissance de deux et que chaque niveau est codé avec le même nombre de bits, L_i sera exprimé par un nombre nécessaire de bits R_i par :

$$L_i = 2^{R_i} \quad (2.8)$$

$$R_i = \log_2 L_i$$

Le nombre total de bits R nécessaire pour coder N scalaires est :

$$R = \sum_{i=1}^N R_i \quad (2.9)$$

À partir de ces équations, on peut calculer le nombre total de niveaux de reconstruction :

$$L = \prod_{i=1}^N L_i = 2^R \quad (2.10)$$

Il est important de noter, à partir des équations, que le nombre total de bits R est la somme des R_i , tandis que le nombre de niveaux de reconstruction L est le produit des L_i .

Si nous avons un nombre de bits fixe pour coder toutes les valeurs N en utilisant la quantification scalaire, l'optimisation du nombre de bits alloués à chaque valeur dépend du critère d'erreur utilisé et de la densité de probabilité des scalaires à coder. Une optimisation typique consiste à allouer plus de bits aux scalaires qui ont une grande variance, et un nombre plus petit de bits pour ceux qui ont une petite variance.

2.3. Codage de Formes d'ondes

Le codage de la forme d'onde consiste à reproduire le signal vocal par une modélisation numérique de la forme d'onde. Ces techniques peuvent accomplir la haute qualité avec une basse complexité et un bas délai. Cependant, le taux de bits nécessaire dans le codage de forme d'onde est aussi haut.

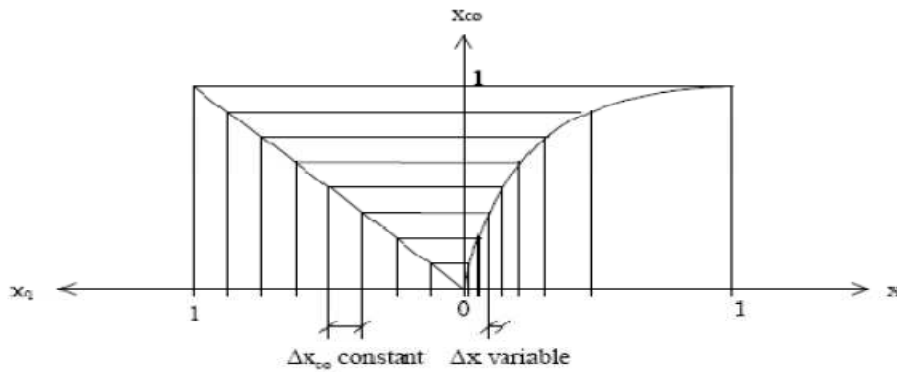
2.3.1. Le Codage PCM

Dans ce type de codage, chaque valeur codée et transmise correspond à la valeur instantanée de l'échantillon de parole, chaque échantillon est quantifié à un nombre fini de niveaux de reconstruction, et à chaque niveau est assigné une séquence unique d'éléments binaires, c'est cette séquence qui est transmise au récepteur [4].

La forme d'onde de parole est prélevée et alors logarithmiquement quantifiée à des niveaux numériques. Il y'a deux méthodes de PCM, qui produisent des débits binaires de 64kb/s (selon la loi A et selon la loi u). Dans ces méthodes les pas de quantification les plus larges sont utilisés pour les échantillons de grandes amplitudes et les pas de quantifications les plus petits sont utilisés pour les échantillons de petites amplitudes. Avec ce type de codage, les échantillons de 13 bits peuvent être comprimés à 8 bits tout en gardant la qualité originale de la parole. La loi A est le standard d'Europe ($A=87.56$) tandis que les Etats-Unis utilise la loi u ($u=255$).

Ces méthodes sont très populaires en raison de leur basse complexité et leurs délais de codage, la Figure 2.1 illustre ce principe de compression avec :

- x : signal original
- x_{co} : signal compressé.
- x_q : signal compressé échantillonné



Loi A (europe)	Loi μ (USA, Japon)
$x_{co} = \begin{cases} \frac{A}{1 + \ln(A)} \cdot x & \text{si } x \leq \frac{1}{A} \\ \frac{1 + \ln(A \cdot x)}{1 + \ln(A)} & \text{si } x > \frac{1}{A} \end{cases}$	$x_{co} = \frac{\ln(1 + \mu \cdot x)}{\ln(1 + \mu)}$
<p>A = 87.6, donc :</p> $x_{co} = \begin{cases} 16 \cdot x & \text{si } x \leq 1.14 \cdot 10^{-2} \\ \frac{1 + \ln(87.6 \cdot x)}{5.473} & \text{si } x > 1.14 \cdot 10^{-2} \end{cases}$	<p>$\mu = 255$, donc :</p> $x_{co} = \frac{\ln(1 + 255 \cdot x)}{5.545}$

Fig.2.1. Lois de compressions

La Figure 2.2 représente un exemple de quantification avec la loi u :

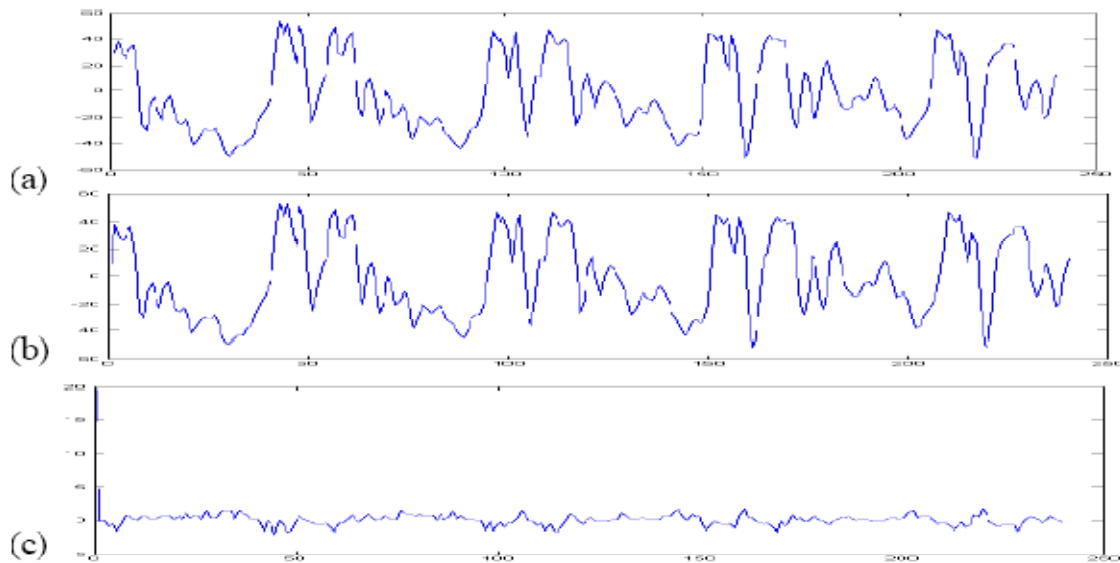


Fig.2.2 Exemple de quantification (loi u)

(a) signal original (b) signal quantifié (c) erreur de quantification

2.3.2. Le Codage Différentiel DPCM, DM et ADM

Il y'a une forte corrélation entre les échantillons adjacents dans le signal parole. Par conséquent, il est plus rentable de coder non pas les échantillons eux même mais la différence entre des échantillons successifs. Cette technique est connue sous le nom de DPCM. Le signal transmis aura une gamme dynamique beaucoup plus inférieure que le signal original, donc il peut être efficacement quantifié en utilisant un quantificateur avec moins de niveaux de reconstruction. Les échantillons précédents sont employés pour prédire la valeur de l'échantillon actuel [4].

La technique est montrée sur la Figure 2.3, où l'erreur de prédiction $e(n)$ — obtenue en soustrayant le signal d'entrée $s(n)$ du signal prédit $s_p(n)$ — est quantifiée. Les indices à la sortie du quantificateur du codeur représentent le segment de bits de la DPCM. L'entrée quantifiée est obtenue en rajoutant la prédiction de l'entrée $s_p(n)$ à la différence quantifiée $\hat{e}(n)$. La prédiction de l'entrée $s_p(n)$ est obtenue à partir de m valeurs quantifiées précédentes de l'entrée. Le décodeur reconstruit l'entrée quantifiée $\hat{s}(n)$ en rajoutant la prédiction de l'entrée $s_p(n)$ à la différence quantifiée $\hat{e}(n)$. La prédiction $s_p(n)$ est obtenue à partir de m valeurs quantifiées ; déjà décodées.

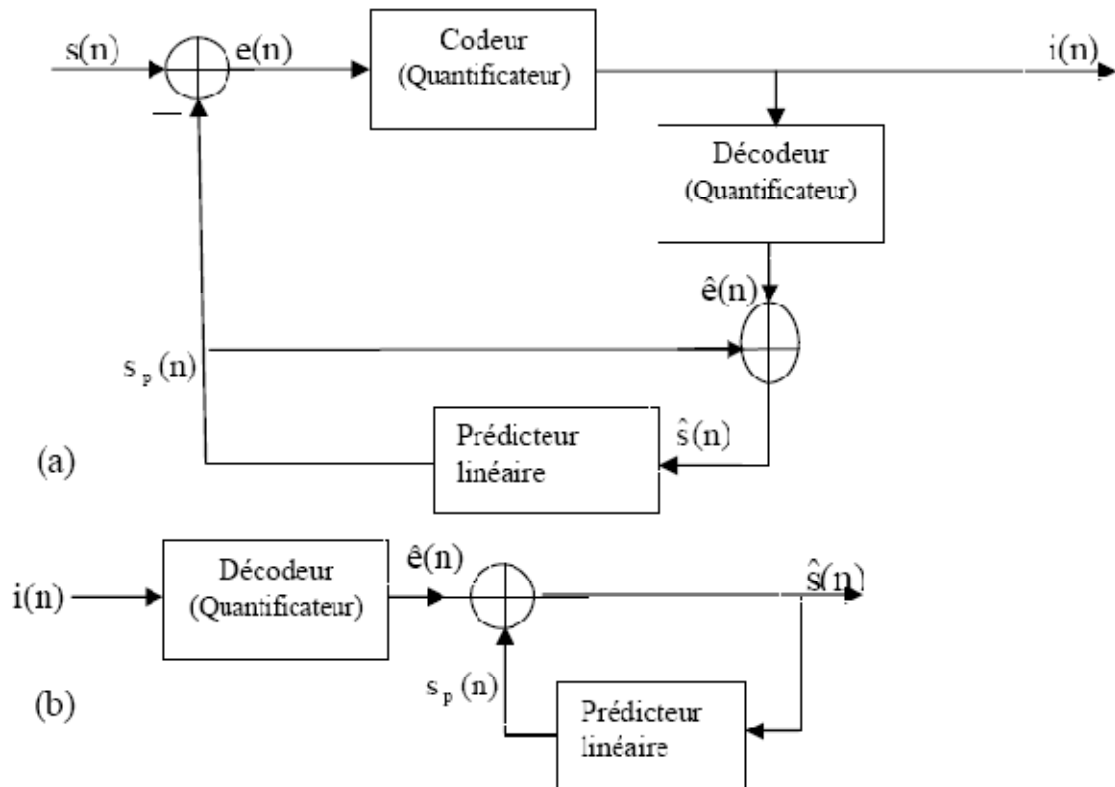


Fig.2.3 Un système DPCM : (a) Codeur et (b) Décodeur

La modulation delta est une sous-classe de la DPCM dans laquelle l'erreur de prédiction est codée avec un seul bit. La modulation delta opère avec un taux d'échantillonnage plus élevé que celui utilisé de la DPCM. Le pas de quantification peut être adaptatif (ADM). Les codeurs DM et DPCM sont de bas -à- moyenne complexité.

Il est possible de réduire le débit binaire en préservant la qualité du message en combinant le principe de la modulation différentielle avec une adaptation dynamique du pas de quantification. Ainsi, l'UIT-T a défini un procédé de modulation par impulsion et codage différentiel adaptatif ADPCM qui fait l'objet de la recommandation G 721.

2.4. Le Codage Fréquentiel

Les codeurs fréquentiels de formes d'ondes divisent le signal en plusieurs composantes séparées de fréquence et les codent indépendamment, le nombre de bits utilisé pour coder chaque composante de fréquence peut varier dynamiquement.

2.4.1. Le Codage en Sous-Bande

C'est le plus simple des techniques du codage fréquentiel. Le signal dans un codeur en sous bandes passe à travers un banque de filtres passe bandes, comme montrée sur la Figure 2.4

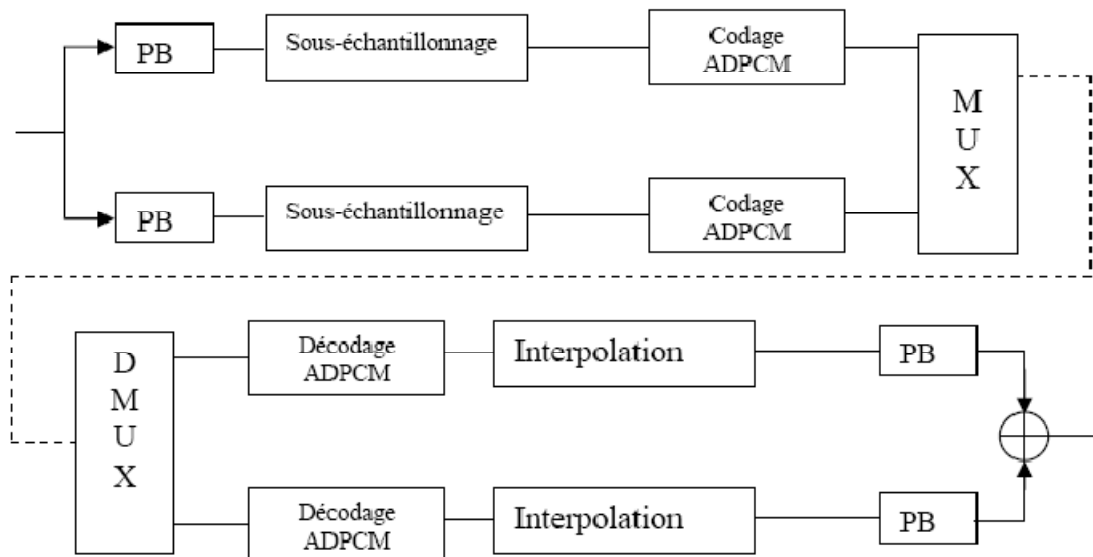


Fig.2.4. Principe de codage en sous bande (PB= filtre passe-bande)

La sortie de chaque filtre est échantillonnée, puis une des techniques du domaine temporel décrites au-dessus est utilisée pour coder chaque sous bande, chaque bande a un nombre de bits assigné, lequel peut être varié d'après l'importance perceptuelle de la bande. Le procédé inverse reproduit le signal original [4].

L'avantage le plus important du codage en sous bande est que le bruit de quantification produit dans une bande est restreint à cette bande. Ceci empêche le bruit de quantification de masquer les composants de fréquence dans d'autres bandes.

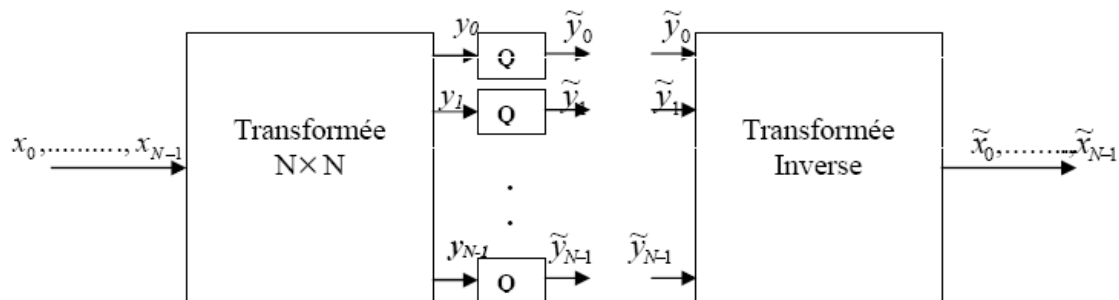
2.4.2. Codage Par Transformée

A : Principes

L'idée est de transformer le signal original dans un autre domaine, afin de décorréler ce dernier, la partie majeure d'information (énergie) dans le nouveau domaine est localisée dans un nombre réduit de coefficients transformés à l'aide d'une transformation réversible.

Dans le codage par transformée (TC), des blocs de N échantillons d'entrée sont transformés en N coefficients, qui sont alors quantifiés et transmis, Figure 2.5.

Au décodeur, une transformation inverse est appliquée aux coefficients, rapportant une reconstruction de la forme d'onde originale. En concevant différents quantificateurs selon les statistiques de leurs entrées, c'est possible d'allouer les bits d'une manière plus optimale, par exemple, codant les coefficients les "plus importants" à un débit binaire plus élevé. Les coefficients qui ne contiennent pas beaucoup d'informations « énergie faible » peuvent être abandonnés, de cette manière nous pouvons obtenir un taux élevé de compression, donc la stratégie de quantification utilisée dépendra de trois facteurs principaux : le débit binaire moyen désiré, les statistiques des différents éléments de la séquence transformée, l'effet de la distorsion des coefficients transformés sur la séquence reconstruite.



B : Transformations Orthogonales

Une Transformation orthogonale $N \times N$, est une opération linéaire à valeurs réelles prenant N échantillons d'entrée aux N échantillons de sortie, ou "coefficients transformés".

Cette opération peut toujours être écrite sous forme matricielle.

$$y(m) = Tx(m) \quad T \in R^{N \times N} \quad (2.11)$$

où $x(m)$ et $y(m)$ sont des vecteurs qui représentent des blocs de dimensions $N \times 1$ d'éléments d'entrée / sortie. Puisque le coefficient $y_k = t_k^T x$ peut être considéré comme résultat d'une "comparaison" entre le $k^{\text{ème}}$ vecteur de base et le vecteur d'entrée x , ces comparaisons sont définies par le produit scalaire $\langle t_k, x \rangle = t_k^T x$.

Plusieurs transformations, comme la transformée de Fourier discrète (DFT), la transformée de Karhunen-Loève (KLT), peuvent être employées, La KLT est une transformation optimale, malheureusement il n'existe pas d'algorithmes rapides pour le calcul de cette transformation.

En pratique on utilise des transformations sous-optimales qui convergent asymptotiquement vers la transformée de Karhunen-Loève mais peuvent être calculées à l'aide des algorithmes rapides. On peut mentionner parmi ces transformations: la transformée en cosinus discrète, DCT, qui est définie comme suit :

$$X_c(k) = \sum_{n=0}^{N-1} x(n)g(k) \cos\left(\frac{(2n+1)k\pi}{2N}\right), k = 0, 1, \dots, N-1$$

Où (2.12)

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X_c(k)g(k) \cos\left(\frac{(1n+1)k\pi}{1N}\right), n = 0, 1, \dots, N-1$$

Avec $g(0)=1$ et $g(k)=\sqrt{2}, k=1, 2, \dots, N-1$

Le codage par transformée est largement utilisé dans le codage audio en large bande et dans la compression d'image, il n'est pas utilisé dans le codage de parole à cause de sa complexité.

2.5. La Quantification Vectorielle

2.5.1. Principes

Dans la quantification vectorielle (QV), les données sont quantifiées conjointement, au lieu de les quantifier individuellement comme dans le cas de la quantification scalaire. Par conséquent, la QV requière moins de bits par rapport à la quantification scalaire pour le même critère de distorsion. Pendant la quantification, un vecteur d'entrée de longueur N est comparé au contenu d'un dictionnaire (codebook), appelé code vecteurs. Le code vecteur le plus proche au vecteur d'entrée d'après un critère spécifié est choisi comme la version quantifiée du vecteur d'entrée [9].

Soit y_n un vecteur d'entrée de dimension N . La version quantifiée de y_n est alors :

$$\begin{aligned} \hat{y}_n &= Q(y_n) \\ &= c_i \end{aligned} \quad (2.13)$$

où $Q(y_n)$ représente l'opération de quantification, c_i est le code vecteur i dans le codebook C , N est la longueur de chaque code vecteur, $L=2^R$ est le nombre de code vecteurs dans C , et R est le nombre de bits utilisé pour représenter l'indice de C . Les vecteurs y_n et c_i ont la relation suivante :

$$d(y_n, c_i) \leq d(y_n, c_k); k = 0, 1, 2, \dots, L-1; k \neq i \quad (2.14)$$

où $d(y_n, c_i)$ représente la fonction de distance.

Dans la plupart des applications de la QV, tel que dans la QV des paramètres LSF, la densité de probabilité de l'entrée à quantifier est inconnue. D'où, le codebook est habituellement dérivé à partir d'un grand nombre de vecteurs d'apprentissage. La séquence d'apprentissage est découpée en L régions dans \mathfrak{R}^N , où la moyenne (centroïde) de chaque région forme le codebook. La région à associer avec un code vecteur c_i est appelée la région Voronoi, (V_i) définie par :

$$V_i = \{y \in \mathfrak{R}^N ; \|y - c_i\| < \|y - c_k\|; k = 0, 1, 2, \dots, L-1, k \neq i\} \quad (2.15)$$

Tout $y_i \in V$ sera quantifié à c_i , un exemple de quantificateur vectoriel à deux dimensions et quatre points ($N=2, L=4$) est illustré par la Figure 2.6 :

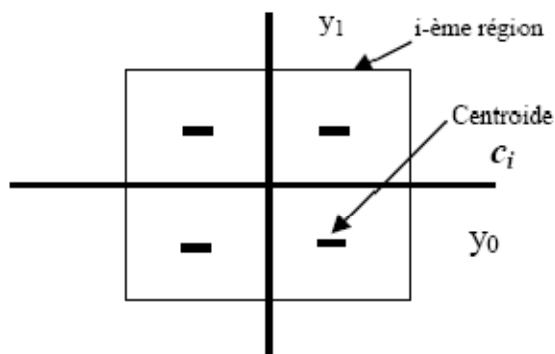


Fig.2.6. Illustration d'un QV à deux dimensions et 4 points.

Une fois que les vecteurs dans V_i sont connus, le code vecteur correspondant y_i est trouvé pour minimiser la distorsion moyenne D_i tel que :

$$D_i = \frac{1}{M_i} \sum_{y_k \in V_i} d(y_k, c_i) \quad (2.16)$$

où M_i est le nombre de vecteurs dans V_i . La distorsion globale D est :

$$D = \sum_{i=1}^N \frac{M_i}{M} D_i \quad (2.17)$$

L'expression de C_i est donnée par :

$$C_i = \frac{1}{M_i} \sum_{y_k \in V_i} y_k \quad (2.18)$$

La méthodologie globale pour obtenir un codebook de taille N est :

1. Commencer avec un codebook initial et calculer la distorsion moyenne.
2. Trouver V_i .
3. Résoudre pour y_i .
4. Calculer la distorsion moyenne.
5. Si la distorsion moyenne diminue moins d'un seuil donné, arrêter. Autrement, aller à l'étape 2.

Si N est une puissance de 2 (nécessaire pour coder), un algorithme croissant qui commence avec un codebook de dimension 1 est formulé comme suit:

1. Trouver un codebook de dimension 1.
2. Trouver un codebook initial de double dimension en faisant une division binaire de chaque code vecteur. Pour une division binaire, un code vecteur est divisé en deux par des petites perturbations.
3. Appliquer la méthodologie itérative présentée plus tôt pour trouver les régions Voronoi et les codes vecteurs pour obtenir le codebook optimal.
4. Si le codebook de la dimension désirée est obtenu, arrêter. Autrement, aller à l'étape 2 dans laquelle la dimension du codebook est doublée.

2.5.2. Quantification Par Split

La méthode la plus simple pour réduire la complexité de recherche et de mémoire, lors de quantification des vecteurs de grande dimension est tout simplement de diviser ces vecteurs en plusieurs sous vecteurs [10]:

Le vecteur original : $x = [x_1, x_2, \dots, x_p]^T$ est partitionnée en R sous vecteurs de plus petite dimension, $x = [x^{(1)}, x^{(2)}, \dots, x^{(R)}]^T$, le $i^{\text{ème}}$ sous vecteur $x^{(i)}$, à la dimension d_i . Par conséquent, $p = d_1 + d_2 + \dots + d_R$, spécifiquement :

$$x^{(1)} = [x_1, x_2 \dots \dots \dots x_{d_1}]^T$$

$$x^{(2)} = [x_{d_1+1}, x_{d_1+2} \dots \dots \dots x_{d_2}]^T$$

$$x^{(3)} = [x_{d_1+d_2+1}, x_{d_1+d_2+2} \dots \dots \dots x_{d_1+d_2+d_3}]^T$$

et ainsi de suite.

Il y'a R quantificateurs, un pour chaque sous vecteur. Les sous vecteur $x^{(i)}$ sont quantifiés individuellement à $y^{(i)}$, donc le vecteur complet x est quantifié à :

$$y = \left[y_k^{(1)} y_k^{(2)} \dots \dots \dots y_k^{(R)} \right]^T \in \mathbb{R}^P$$

Les quantificateurs sont conçus par l'utilisation des sous vecteurs appropriés.

Le cas extrême d'une QV par split est lorsque $R=p$. Alors $d_1 = d_2 = \dots = d_p = 1$ et nous obtenons un quantificateur scalaire.

La réduction dans l'exigence de la mémoire et la complexité de la recherche est illustrée par un exemple.

On suppose que la dimension du vecteur à coder est $p=10$. Une pleine QV de 30 bits aura un seul codebook de 230 codes vecteurs. Un quantificateur par split équivalent de $R=3$ emploie des sous vecteurs de dimensions $d_1 = 3, d_2 = 3, \text{ et } d_3 = 4$, respectivement. Pour chaque sous vecteur, il y aura un codebook ayant 210 codes vecteurs, chacun est codé sur 10-bits.

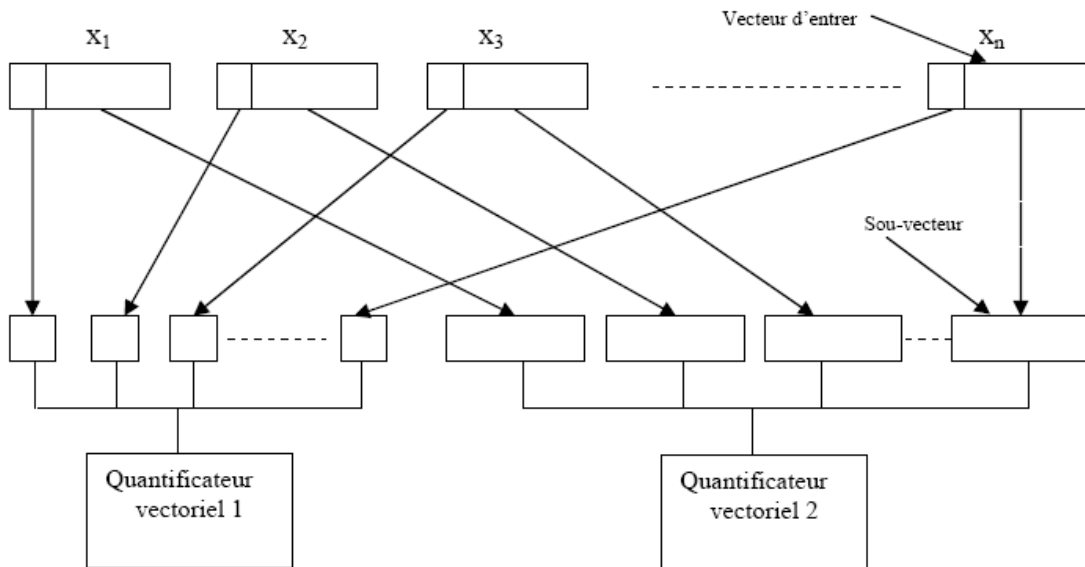


Fig.2.7. Diagramme d'un quantificateur par split (deux sous vecteurs)

2.6. Principe d'analyse Par Synthèse

Dans un système de codage en boucle ouverte (open-loop), les paramètres sont calculés à partir du signal d'entrée, quantifiés et plus tard utilisés pour la synthèse (reconstruction de parole). C'est le principe dans plusieurs codeurs. Une méthode plus efficace est

d'employer les paramètres pour synthétiser le signal parole pendant le codage et les régler pour produire la reconstruction la plus précise. Conceptuellement, c'est un procédé en boucle fermée (closed-loop) d'optimisation, où le but est de choisir les meilleurs paramètres afin de rapprocher autant que possible la parole reconstituée avec la parole originale. La Figure 2.8 illustre le schéma fonctionnel d'un codeur avec l'approche en boucle fermée [8].

Pour une quantification scalaire d'erreur de prédiction au moins un bit par échantillon est nécessaire. Par conséquent, la majorité des bits est utilisée pour la quantification du signal résiduel. Le taux de bit pour la quantification du signal résiduel peut être baissé considérablement en codant ce signal par blocs, c-à-d., appliquer une forme de QV.

La dimension du vecteur ne doit pas être trop grande, pour que la complexité ne devienne pas un problème.

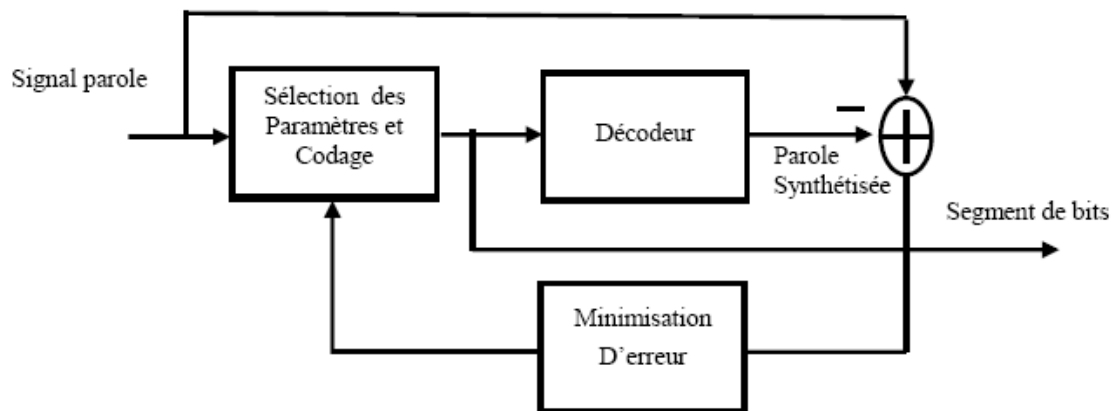


Fig.2.8. Principe d'analyse par synthèse

2.7. Les Codeurs à Bas Débit et à Très Bas Débit

2.7.1. Codage à Excitation Multi-Pulse et Regular-Pulse

Dans les codeurs multi-pulse, l'excitation est représentée par un nombre restreint d'impulsions à des intervalles non réguliers. Les amplitudes et les positions d'impulsions doivent être codées. Trouver la combinaison optimale des endroits d'impulsions et leurs amplitudes par l'Analyse par Synthèse est un problème très complexe. Par conséquent, des procédures sous optimales sont souvent employées là où les endroits et les amplitudes d'impulsion sont trouvés une par une. Les codeurs Multi-pulse opèrent à un taux de bits d'environ 16 kbps [4].

Dans le codeur à excitation regular-pulse, les impulsions sont uniformément espacées, dans ce cas, seulement la position de la première impulsion et les amplitudes de toutes les impulsions doivent être codées. Une version du codeur d'excitation regular-pulse est recommandée par "mobile de Groupe Spéciale (GM/M)" pour la radio cellulaire numérique en Europe.

2.7.2. Le Codeur CELP

Une technique réussie de codage de parole à bas débit est le codage CELP (codeexcited linear prediction) [Schroeder et Atal 1985]. Il est très efficace pour les débits moyens de 4.8 kbit/s à 16 kbit/s, comme en témoignent les nombreuses normes qui l'utilisent. La Figure 2.9 représente le principe du codage CELP [4][11].

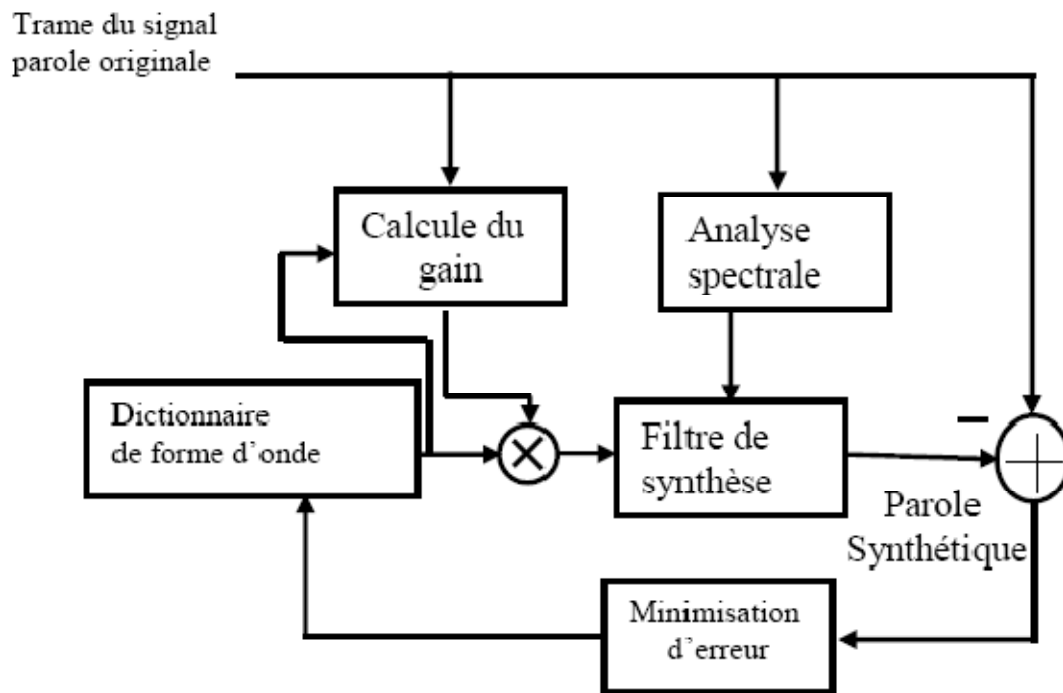


Fig.2.9. Schéma simplifié d'un codeur CELP

Dans chaque trame, une analyse spectrale par prédiction linéaire détermine le filtre de synthèse $1/A(z)$. On découpe chaque trame en sous-trames plus courtes (durée typique 5 ms) sur lesquelles on effectue une quantification vectorielle du signal par une technique d'analyse par synthèse. On compare à l'aide d'un critère dit « perceptuel » de type moindres carrés pondérés, le signal de parole original avec tous les signaux synthétiques possibles obtenus après quantification vectorielle. Ces signaux synthétiques sont générés en filtrant par le filtre de synthèse, un signal d'excitation choisi dans un dictionnaire de séquences d'excitation (on ajoute parfois la sortie de plusieurs dictionnaires) et en ajustant le signal résultant par le gain optimal. Le codeur transmet le ou les index des

segments qui minimisent le critère ainsi que le ou les gains associés, les paramètres spectraux et le pitch fractionnaire. Le critère perceptuel prend en compte la propriété de masquage du bruit de quantification par les formants en pondérant plus fortement l'erreur de quantification dans les zones de faible amplitude du spectre et plus faiblement dans les zones de formants. Cette pondération s'effectue en filtrant le signal d'erreur par un filtre de type $A(z)/A(z/\gamma)$ où γ est compris entre 0 et 1 (typiquement $\gamma = 0.85$). Les dictionnaires utilisés sont appelés stochastiques ou adaptatifs selon qu'ils contiennent des séquences fixes de bruit ou bien les séquences d'excitation de trames précédentes. Le dictionnaire adaptatif permet de prendre en compte la redondance introduite par la quasi-périodicité des sons voisés. La qualité subjective des codeurs CELP décroît rapidement lorsque le débit descend en dessous de 4 kbit/s.

2.7.3. Les Codeurs LPC à Excitation Mixte ou MELP

Le standard MELP à 2400 bits/s est un codeur LPC à excitation mixte (MELP=Mixed Excitation Linear Prediction). Il utilise une excitation mixte c'est-à-dire formée de la somme d'une composante impulsionnelle et d'une composante de bruit. La composante impulsionnelle est formée d'un train d'impulsions périodique ou non. Cette excitation est une excitation multibande avec une intensité de voisement définie pour chaque bande de fréquence. Le codeur fait une première estimation de la fréquence fondamentale, puis il calcule l'intensité de voisement dans cinq bandes de fréquences adjacentes. L'intensité de voisement est déterminée dans chaque bande par la valeur de l'autocorrélation normalisée par la valeur de la période de pitch. Dans la norme, cette intensité est codée sur 1 bit, chaque bande est donc classée voisée ou non-voisée. Après analyse le codeur peut positionner un indicateur appelé indicateur d'apériodicité (aperiodic flag) pour indiquer au décodeur que la composante impulsionnelle doit être apériodique. Le codeur effectue par ailleurs une analyse spectrale par prédiction linéaire et calcule les amplitudes des 10 premières harmoniques du pitch sur la transformée de Fourier du signal résiduel. Ces amplitudes sont quantifiées de manière vectorielle. Les paramètres transmis par le codeur sont finalement: La période fondamentale, le drapeau d'apériodicité, les cinq intensités de voisement, deux gains (correspondant aux énergies de demi-trames), les paramètres spectraux et les 10 amplitudes d'harmoniques du pitch codés par quantification vectorielle [11].

Le synthétiseur interpole linéairement les différents paramètres de manière synchrone au pitch. La composante impulsionnelle est obtenue sur une période de pitch par transformée de Fourier inverse sur les 10 amplitudes de Fourier. Pour les sons non-voisés ou lorsque l'indicateur d'apériodicité est positionné, une perturbation aléatoire (jitter) est appliquée à la valeur de la période fondamentale. Cette possibilité d'excitation impulsionnelle non périodique est particulièrement intéressante pour les zones de transitions entre sons. La composante impulsionnelle et la composante de bruit sont filtrées puis ajoutées. Le filtrage appliqué à la composante impulsionnelle a pour réponse impulsionnelle la somme de toutes les réponses impulsionnelles des filtres passe-bande pour les bandes voisées. Le

filtrage de la composante de bruit est déterminé de la même façon à partir des bandes non-voisées.

L'excitation globale est ensuite filtrée par un filtre adaptatif de renforcement des formants et par le filtre de synthèse LPC. Le signal synthétique résultant est mis à l'échelle en fonction de l'énergie de la trame originale et passé dans un filtre dont le but est d'étaler l'énergie des impulsions sur une période de pitch.

2.7.4. Codeur à Excitation Multibande (MBE)

Dans les codeurs MBE le signal est analysé dans plusieurs bandes de fréquences adjacentes et est déclaré voisé ou non dans chaque bande. Le nombre de bandes est de l'ordre d'harmoniques de F_0 entre 0 et $F_c/2$. La figure 2.10 représente l'amplitude de la transformée de Fourier discrète d'une trame de signal parole avec ces zones harmoniques (Voisées) ou non harmonique (Non voisées) représentées par les signes V et UV. Les paramètres transmis par le codeur sont : la fréquence fondamentale, l'information de voisement pour chaque bande, et les paramètres décrivant l'enveloppe spectrale [11].

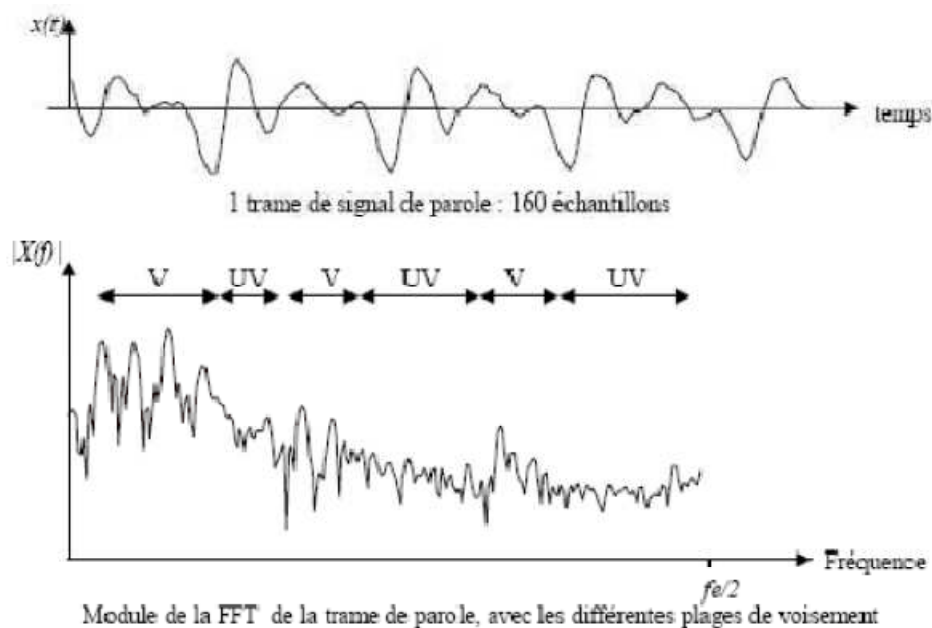


Fig.2.10. Zones voisées (V) ou non voisées (UV) du spectre d'une trame de parole

2.8. Critères Relatifs au Codage

Pour conclure le chapitre, nous allons énumérer les différents critères couramment utilisés pour juger et classer les méthodes de codage. D'autres critères peuvent être importants, selon le type d'application [12].

2.8.1. Débit de Transmission

Le débit de transmission détermine le nombre de bits par seconde alloué au codeur pour la représentation de l'information. L'objectif d'un algorithme de codage est de réduire ce débit en maintenant la bonne qualité du signal. La largeur de bande disponible dans le système de communication détermine la limite supérieure du débit envisageable du codeur de parole. Lors de la conception d'un système de codage, un choix sera effectué parmi les codeurs à débit de transmission fixe ou variable.

2.8.2. Délai de Codage

Le délai de codage est très important dans les transmissions en temps réel. Le délai global est engendré par le temps de traitement du codage et du décodage mais aussi par le délai de transmission qui dépend du canal utilisé et les différents temps d'attente liés à l'application choisie. De ce temps de restitution du signal dépendra la qualité d'écoute de la communication.

2.8.3. Qualité de Parole

Une considération importante dans tout codage de parole est la qualité du signal reconstruit. Les recherches sur les différents types de codage essaient toujours de trouver un bon compromis entre la qualité du signal de parole restitué et le débit de transmission. Pour un débit fixé, le critère de qualité pourra alors être employé pour évaluer un système de codage. Deux types de mesures, objective et subjective, peuvent permettre l'évaluation de la qualité de parole.

2.8.4. Mesures Subjective de La Qualité de Parole

La qualité du signal reconstitué peut être déterminée par des tests d'écoute du signal codé et décodé dans des conditions désirées, où des auditeurs jugent subjectivement la qualité globale et l'intelligibilité de la parole. Pour ce type d'étude, un grand nombre de personnes est nécessaire pour effectuer une analyse statistique de leur opinion moyenne (MOS : Mean Opinion Score). Une telle expérimentation est très contraignante à réaliser mais reste incontournable. Un échantillon de résultats de test simple, comme par exemple comparer le signal synthétisé au signal original, peut apporter de premières informations significatives sur les forces et les faiblesses de la procédure de codage.

2.8.5. Mesures objectives

Les mesures objectives utilisent des fonctions ou des critères mathématiques pour comparer les formes d'ondes, les spectres ou les cepstres codés et originaux.

Certaines mesures donnent des informations utiles selon le type de codage testé. Par exemple, le Rapport Signal sur Bruit (SNR : Signal to Noise Ratio) est représentatif pour les codeurs temporels et certains codeurs hybrides, tels que les codeurs de type CELP, qui incorporent des mécanismes de modélisation de forme d'onde. D'autres évaluent certains éléments des algorithmes de codage, tels que les distorsions cepstrales ou spectrales, employés pour calculer la déformation introduite par la quantification des paramètres LPC. Idéalement, les mesures objectives recourent les résultats obtenus par notre perception subjective de la parole. Toutefois, les tests subjectifs et objectifs peuvent produire des résultats légèrement différents.

A : Rapport signal à bruit (RSB)

Le rapport signal à bruit représente le rapport de la puissance du signal à la puissance du bruit. Le RSB s'exprime souvent en décibel suivant la relation :

$$RSB = 10 \log_{10} \left(\frac{\sum_n s^2(n)}{\sum_n (s(n) - \hat{s}(n))^2} \right)$$

Le RSB n'est pas approprié pour mesurer la distorsion des signaux non-stationnaires. C'est pour cela que l'on a introduit le RSB segmental défini comme la moyenne des RSB calculés sur des trames courtes.

B : RSB Segmental (RSBS)

C'est une amélioration en ce qui concerne la mesure conventionnelle du RSB, faite pour manipuler la nature dynamique des signaux non stationnaires tels que la parole. La RSBS est définie par :

$$RSB_{seg} = \frac{1}{N} \sum_{m=1}^N SNR_m$$

N est le nombre de trames.

C: Mesure de la distorsion spectrale

Pour déterminer la qualité d'une enveloppe spectrale estimée, la distorsion spectrale (SD) est calculée sur le spectre de puissance dans le plan fréquentiel comme une mesure objective.

$$SD = \sqrt{\frac{1}{N} \sum_{i=0}^{N-1} \left(10 \log \log_{10} \frac{P_i}{\hat{P}_i} \right)^2}$$

Où P_i et \hat{P}_i sont les spectres original et estimé respectivement, cette mesure est effectuée chaque trame d'analyse LPC.

2.9. La décomposition en valeurs singulières SVD

Les premiers codeurs de parole utilisent un modèle très simple d'excitation qui capture seulement ses caractéristiques minimales et un tel modèle est incapable de produire une parole de résonance naturelle que celui produit par le système vocal humain.

Plusieurs travaux ont été effectués pour trouver une représentation adéquate de l'excitation. Récemment, l'analyse par synthèse est devenue une approche populaire pour reconstruire la parole avec une erreur perceptuelle minimale en optimisant les positions d'impulsions et les amplitudes du signal d'excitation comme dans les codeurs multi-pulse, ou on cherche à trouver le meilleur vecteur d'excitation dans un codebook comme dans le codeur CELP.

Des représentations plus complexes de l'excitation sont incluses dans les codeurs MBE qui —comme les premiers vocodeurs LPC— classifient l'excitation voisée/non voisée mais pour des bandes de fréquence différentes à travers les spectres.

B.S. Atal a proposé une représentation paramétrique du signal résiduel basée sur la décomposition en valeurs singulières (SVD) de la matrice d'autocorrélation de la réponse impulsionnelle du filtre LPC. Dans cette représentation, la forme d'onde d'excitation est exprimée comme une combinaison linéaire des vecteurs propres de la matrice d'autocorrélation de la réponse impulsionnelle du filtre LPC.

La décomposition en valeurs singulières (SVD), ses caractéristiques et son application à la modélisation du signal d'excitation font l'objet ce chapitre.

2.9.1. Principes

La Décomposition en Valeurs Singulières (SVD) est l'une des méthodes les plus fondamentales et les plus importantes d'algèbre linéaire, elle porte quelque ressemblance avec la Décomposition en Valeurs Propres (ED), mais elle est plus générale.

Habituellement, l'ED est d'intérêt sur les matrices carrés et symétriques seulement, mais la SVD peut être appliqué à toute matrice.

La (SVD) décompose une matrice arbitraire A de dimension $(m \times n)$ dans une matrice diagonale S , de la même dimension de A et avec éléments diagonaux réels, non négatifs dans un ordre décroissant, et des matrices orthogonales U et V de dimension $(m \times m)$ et $(n \times n)$, respectivement [13].

- Valeurs singulières : sont les racines des valeurs propres de $A^T A$
- La matrice orthogonale V est la matrice des vecteurs propres de $A^T A$
- La matrice orthogonale U est la matrice des vecteurs propres de $A A^T$

$A^T A$ et $A A^T$ sont des matrices carrées symétriques :

- Les valeurs propres d'une matrice symétrique sont réelles
- Leurs vecteurs propres sont orthogonaux
- Diagonalisables

Les valeurs singulières et les vecteurs singuliers d'une matrice arbitraire A , sont : un scalaire non négatif s et deux vecteurs non nuls u et v tels que :

$$AV = SV$$

$$A^H V = SV \quad (H \text{ est l'opérateur hermitien})$$

$$A = USV^T \tag{3.1}$$

2.9.2. Interprétation Géométrique de la SVD

Les scalaires réels s_i sont appelés valeurs singulières de la matrice complexe A . Ce ne sont rien d'autre que les racines carrées des valeurs propres de la matrice $A^T A$, c'est-à-dire $\sqrt{(A A^T)}$.

La plus grande valeur singulière s_1 définit une norme sur la matrice A . Le nombre de valeurs singulières de A non nulles indique le rang de la matrice A .

Par suite, la matrice U (resp. V) a ses colonnes orthonormales : elles correspondent à des matrices associées à des rotations : la matrice U (resp. V) à une rotation d'angle φ_1 (resp. φ_2).

Si on applique la matrice A à un vecteur x , celui-ci subit d'abord une rotation définie par la matrice V^T puis chacune de ces composantes est dilatée par s_i avant de subir une nouvelle rotation définie par U .

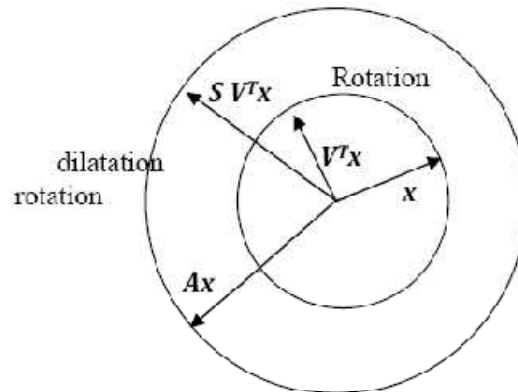


Fig. 3.1. Transformation d'un vecteur x par A quand $s_1 = 2$ et $s_2 = 1$

Conclusion

Dans ce chapitre une présentation des différents codeurs de parole a été faite, on s'est intéressé surtout au rôle central de l'excitation dans un codeur. Le codage multi-pulse, regular-pulse, l'utilisation du codebook fixe et du codebook adaptatif dans le codage CELP sont les représentations d'excitation les plus répandues, mais elles restent incapables de baisser le débit au dessous d'une certaine limite en gardant en même temps une qualité convenable de la parole reconstituée, d'où la nécessité de chercher à trouver une représentation paramétrique de cette dernière.

Introduction aux FPGA

Introduction

Les FPGA (*Field Programmable Gate Array*) sont des composants entièrement reconfigurables, ce qui permet de les reprogrammer à volonté afin d'accélérer notablement certaines phases de calcul. Ces circuits ont été inventés par la société Xilinx en 1985.

L'avantage de ce genre de circuits est sa grande souplesse qui permet de le réutiliser plusieurs fois dans des algorithmes différents en un temps très court (quelques millisecondes ou parfois moins selon la complexité de l'algorithme et les capacités du FPGA).

Le progrès de ces technologies a permis de faire des composants toujours plus rapides et à plus haute intégration, ce qui permet d'implémenter des applications complexes et à faible coût.

Elles sont utilisées pour de nombreuses applications. En voici quelques unes :

- Prototypage de nouveaux circuits.
- Fabrication de composants spéciaux en petite série.
- Adaptation aux besoins rencontrés lors de l'utilisation.
- Systèmes de commande temps réel.
- Imagerie médicale.

Les circuits FPGA possèdent une structure matricielle de deux types de blocs (ou cellules). Des blocs d'entrée/sortie et des blocs logiques programmables. Le passage d'un bloc à un autre se fera par un routage programmable. Certains intègrent également des mémoires RAM, des multiplieurs, des convertisseurs et même des noyaux de processeur.

Actuellement, deux leaders mondiaux se disputent le marché des FPGA : Xilinx et Altera. Nous allons étudier les FPGA de Xilinx qui sont utilisés dans notre application.

3.1. Architecture adoptée par Xilinx

L'architecture retenue par Xilinx se présente sous forme de deux couches :

- **Circuit configurable** :

La couche dite circuit configurable est constituée d'une matrice de blocs logiques configurables (CLB) permettant de réaliser des fonctions combinatoires et des fonctions séquentielles. Tout autour de ces blocs logiques configurables, nous trouvons des blocs d'entrée/sortie (IOB) dont le rôle est de gérer les entrées/sorties réalisant l'interface avec les modules extérieurs. La programmation des FPGA appelés aussi LCA (*logic cells arrays*) consistera, par le biais de l'application d'un potentiel adéquat sur la grille de certains transistors à effet de champ, à interconnecter les éléments des CLB et des IOB afin de réaliser les fonctions souhaitées et d'assurer la propagation des signaux.

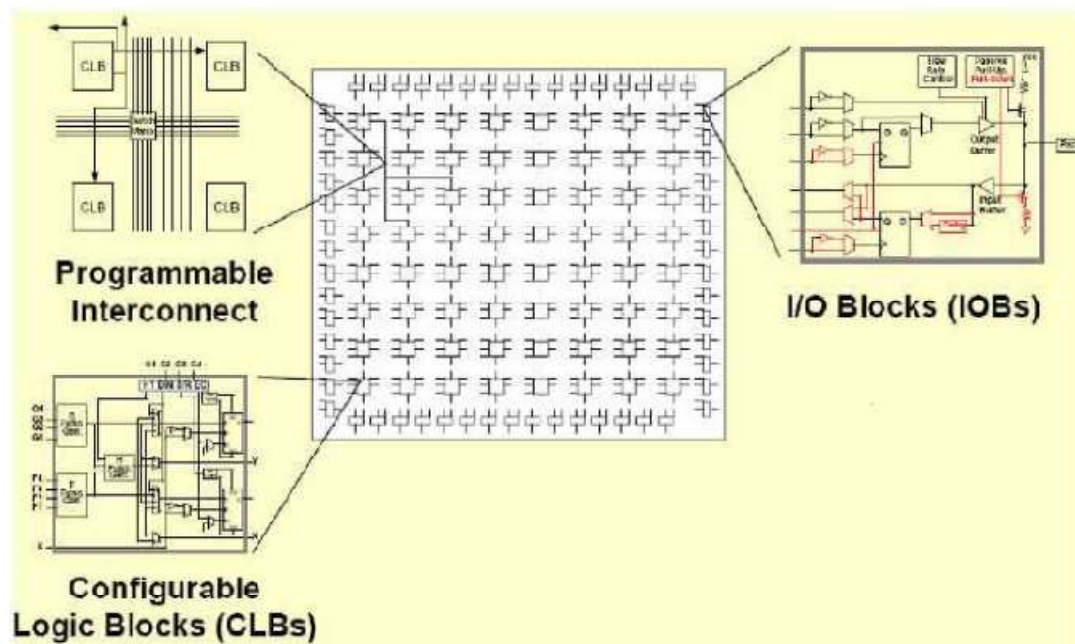


Figure 4.1 Architecture interne des FPGA 1 de Xilinx

➤ Le réseau mémoire :

La configuration du circuit est mémorisée sur la couche réseau SRAM et stockée dans une ROM externe. Un dispositif interne permet à chaque mise sous tension de charger la SRAM interne à partir de la ROM. Ainsi, on réalise aisément qu'un même circuit puisse être exploité successivement avec des ROM différentes puisque sa configuration interne n'est jamais définitive.

3.2. Les circuits configurables

La première couche de circuit configurable est constituée de trois sortes de circuits : les cellules logiques CLB (*configurable logic bloc*), les cellules d'entrée/sortie IOB (*input*

output bloc) qui sont connectées entre elles par un réseau d'interconnexions configurables.

➤ **CLB (configurable logic bloc) :**

Les blocs logiques configurables sont les éléments les plus déterminants des performances du FPGA. Chaque bloc est composé d'un bloc de logique combinatoire composé de deux générateurs de fonctions à quatre entrées et d'un bloc de mémorisation et de synchronisation composé de deux bascules D. Quatre autres entrées permettent d'effectuer les connexions internes entre les différents éléments du CLB.

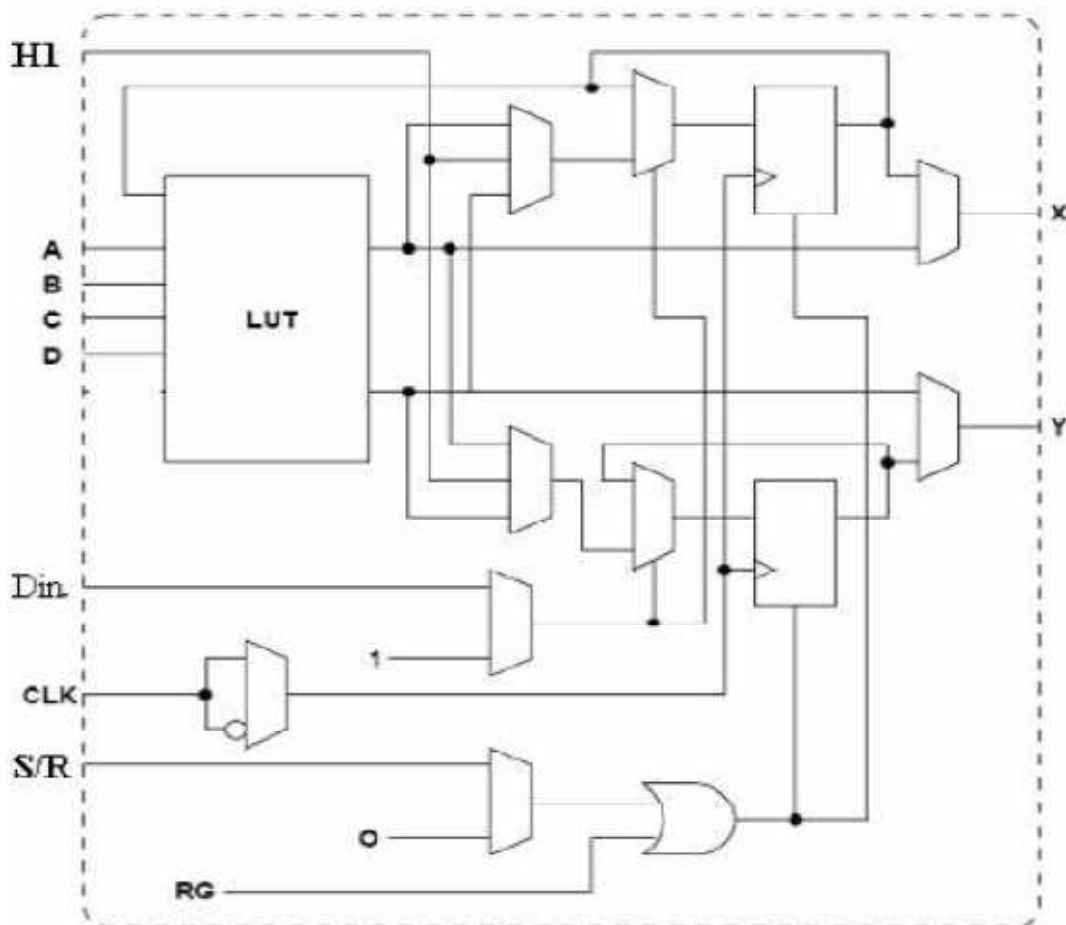


Figure 1.2 Bloc logique configurable (CLB)

Voyons d'abord le bloc logique combinatoire. Il possède deux générateurs de fonctions F' et G' à quatre entrées indépendantes ($F1...F4$; $G1...G4$), lesquelles offrent aux concepteurs une flexibilité de développement importante car la majorité des fonctions aléatoires à concevoir n'excède pas quatre variables. Les deux fonctions sont générées à partir d'une table de vérité câblée inscrite dans une zone mémoire, rendant ainsi les

détails de propagation pour chaque générateur de fonction indépendants de celle à réaliser. Une troisième fonction H' est réalisée à partir des sorties F' et G' et d'une troisième variable d'entrée H1 sortant d'un bloc composé de quatre signaux de contrôle H1, Din, S/R, Ec.

Les signaux des générateurs de fonctions peuvent sortir du CLB soit par la sortie X pour les fonctions F' et G', soit par la sortie Y pour les fonctions G' et H'. Ainsi un CLB peut être utilisé pour réaliser :

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

L'intégration de fonctions à nombreuses variables diminue le nombre de CLB nécessaires, et par conséquent les délais de propagation des signaux augmentent la densité et la vitesse du circuit. Les sorties de ces blocs logiques peuvent être appliquées à des bascules au nombre de deux ou directement à la sortie du CLB (sorties X et Y). Chaque bascule présente deux modes de fonctionnement : un mode 'flip-flop' avec comme donnée à mémoriser, soit l'une des fonctions F', G', H' soit l'entrée directe DIN. La donnée peut être mémorisée sur un front montant ou descendant de l'horloge (CLK). Les sorties de ces deux bascules correspondent aux sorties des CLB X_Q et Y_Q. Un mode dit de "verrouillage" exploite une entrée S/R qui peut être programmée soit en mode SET, mise à 1 de la bascule ; soit en mode RESET, mise à 0 de la bascule. Ces deux entrées coexistent avec une autre appelée le global Set/Reset. Cette entrée initialise le circuit FPGA à chaque mise sous tension, à chaque configuration, en commandant toutes les bascules au même instant soit à '1', soit à '0'. Elle agit également lors d'un niveau actif sur le fil RESET lequel peut être connecté à n'importe quelle entrée du circuit FPGA.

➤ **IOB (input output bloc) :**

La figure représente la structure de ce bloc. Ces blocs entrée/sortie permettent l'interfaçage entre les broches du composant FPGA et la logique interne développée à l'intérieur du composant. Ils sont présents sur toute la périphérie du circuit FPGA. Chaque bloc IOB contrôle une broche du composant et il peut être défini en entrée, en sortie, en signaux bidirectionnels ou être inutilisée (haute impédance).

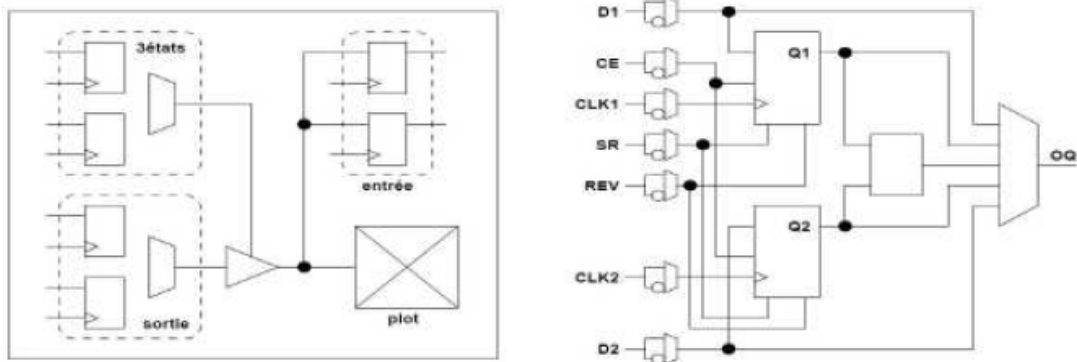


Figure 4.3 Bloc d'entrée / sortie (IOB)

○ Configuration en entrée

Premièrement, le signal d'entrée traverse le buffer qui selon sa programmation peut détecter soit des seuils TTL ou soit des seuils CMOS. Il peut être routé directement sur une entrée directe de la logique du circuit FPGA ou sur une entrée synchronisée. Cette synchronisation est réalisée à l'aide d'une bascule D ; le changement d'état peut se faire sur un front montant ou descendant. De plus, cette entrée peut être retardée de quelques nanosecondes pour compenser le retard pris par le signal d'horloge lors de son passage par l'amplificateur. Le choix de la configuration de l'entrée s'effectue grâce à un multiplexeur (*program controlled multiplexer*). Un bit positionné dans une case mémoire commande ce dernier.

○ Configuration en sortie

Nous distinguons les possibilités suivantes :

- ✓ Inversion ou non du signal avant son application à l'IOB.
- ✓ Synchronisation du signal sur des fronts montants ou descendants de l'horloge.
- ✓ Mise en place d'un "pull-up" ou "pull-down" dans le but de limiter la consommation des entrées/sorties inutilisées.
- ✓ Signaux en logique trois états ou deux états. Le contrôle de mise en haute impédance et la réalisation des lignes bidirectionnelles sont commandés par le signal de commande Out Enable lequel peut être inversé ou non. Chaque sortie peut délivrer un courant de 12 mA. Ainsi toutes ces possibilités permettent au concepteur de connecter au mieux une architecture avec les périphériques extérieurs.

➤ Les différents types d'interconnexions :

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

- **Les interconnexions à usage général**

Ce système fonctionne en grille de cinq segments métalliques verticaux et quatre segments horizontaux positionnés entre les rangées et les colonnes du CLB et de l'IOB.

Des aiguilleurs appelés aussi matrices de commutation sont situés à chaque intersection. Leur rôle est de raccorder les segments entre eux selon diverses configurations, ils assurent ainsi des la communication des signaux d'une voie sur l'autre. Ces interconnexions sont utilisées pour relier un CLB à n'importe quel autre. Pour éviter que les signaux traversent les grandes lignes ne soient affaiblies, nous trouvons généralement des buffers implantés en haut et à droite de chaque matrice de commutation.

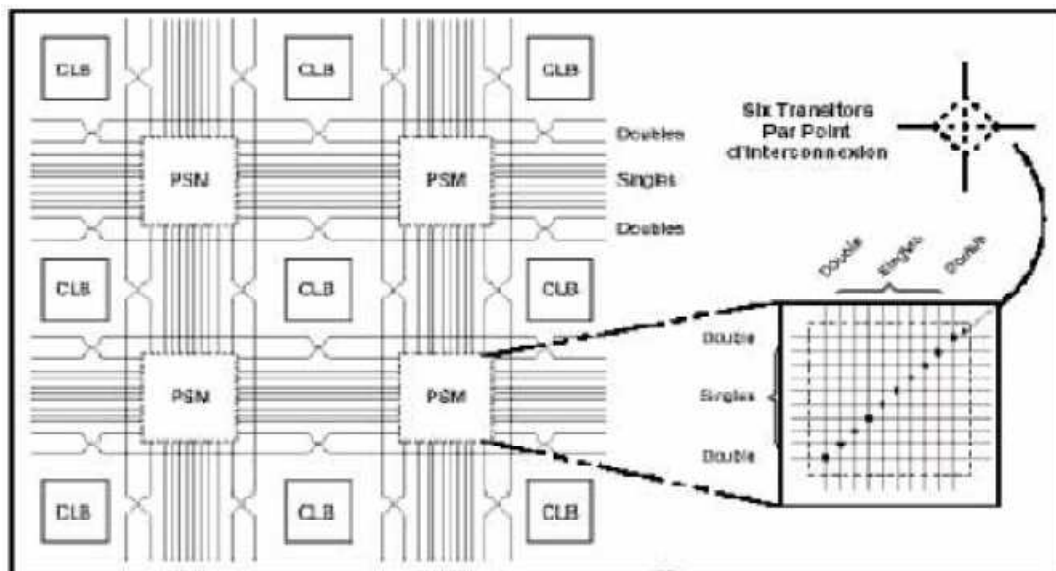


Figure 4.4 Connexions à usage général et détail d'une matrice de commutation

- **Les interconnexions directes**

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

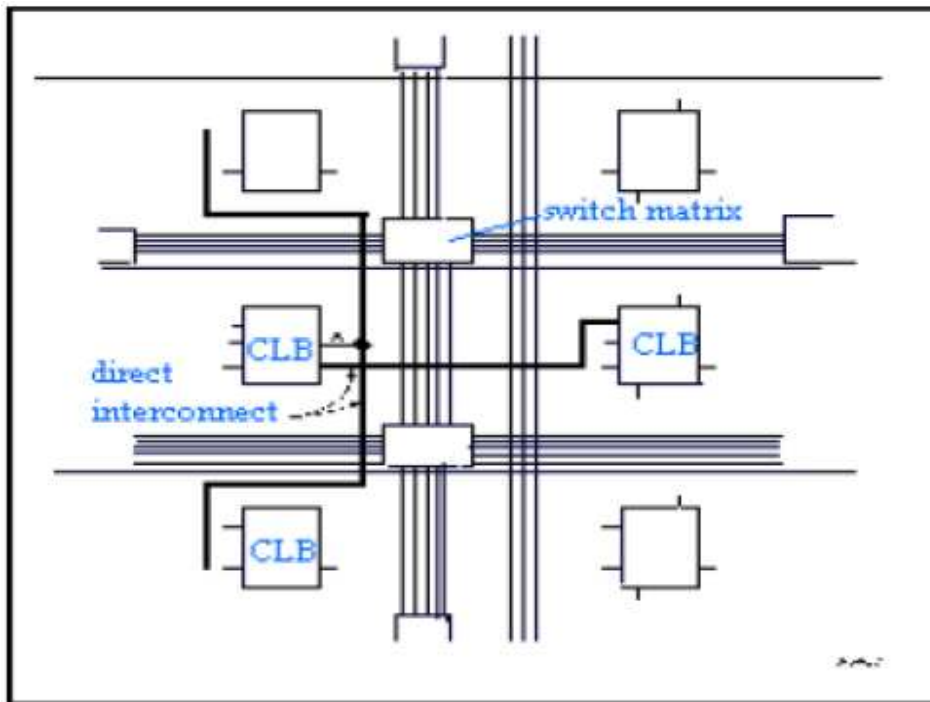


Figure 4.5 Les interconnexions directes

Pour chaque bloc logique configurable, la sortie X peut être connectée directement aux entrées C ou D du CLB situés au-dessus. Quant à la sortie Y, elle peut être connectée à l'entrée B du CLB placée immédiatement à sa droite. Pour chaque bloc logique adjacent à un bloc entrée/sortie, les connexions sont possibles avec les entrées I ou les sorties O suivant leur position sur le circuit.

- **Les longues lignes**

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

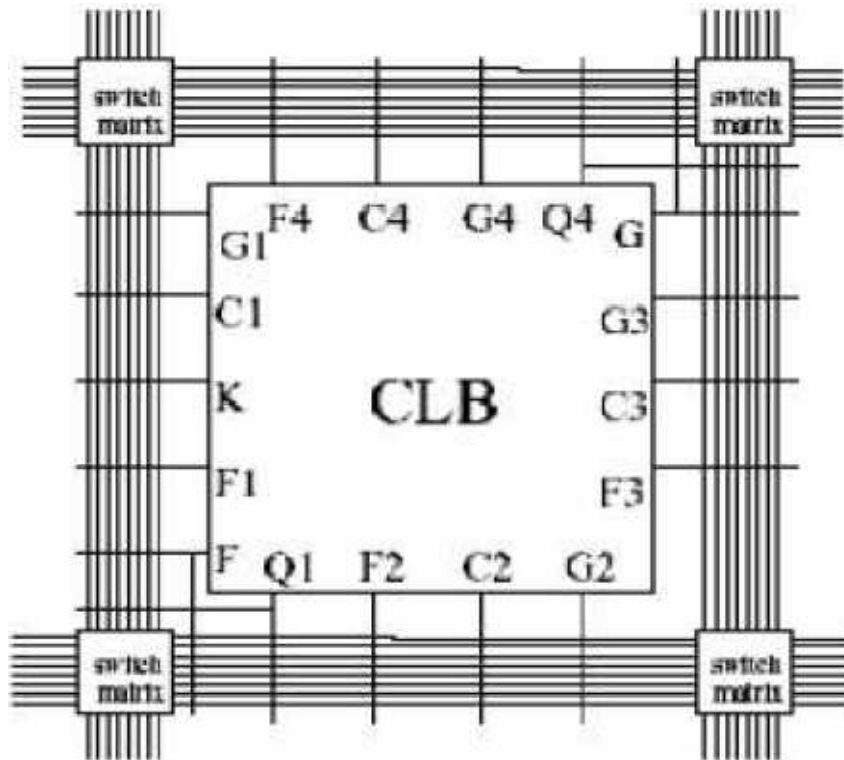


Figure 4.6 Les longues lignes

- Performances des interconnexions

Les performances des interconnexions dépendent du type de connexions utilisées. Pour les interconnexions à usage général, les délais générés dépendent du nombre de segments et de la quantité d'aiguilleurs employés. Le délai de propagation de signaux utilisant les connexions directes est minimum pour une connectique de bloc à bloc. Quant aux segments utilisés pour les longues lignes, ils possèdent une faible résistance mais une capacité importante. De plus, si on utilise un aiguilleur, sa résistance s'ajoute à celle existante.

Le langage VHDL

Introduction

Le langage VHDL a été créé pour décrire des systèmes numérisés destinés à une implantation dans des circuits logiques programmables (PLD, FPGA, ASIC). Il remplace la saisie de schéma traditionnelle. La plupart des outils de développement autorisant les deux types de saisie.

Le langage VHDL est en principe standardisé. Chaque outil de développement dispose cependant de bibliothèques spécifiques.

Pour développer une application de logique programmable, il faudra suivre la démarche ci-dessous :



- L'étape de vérification permet de valider la syntaxe du programme. La « netlist » est un fichier contenant la description de l'application sous forme d'équations logiques.

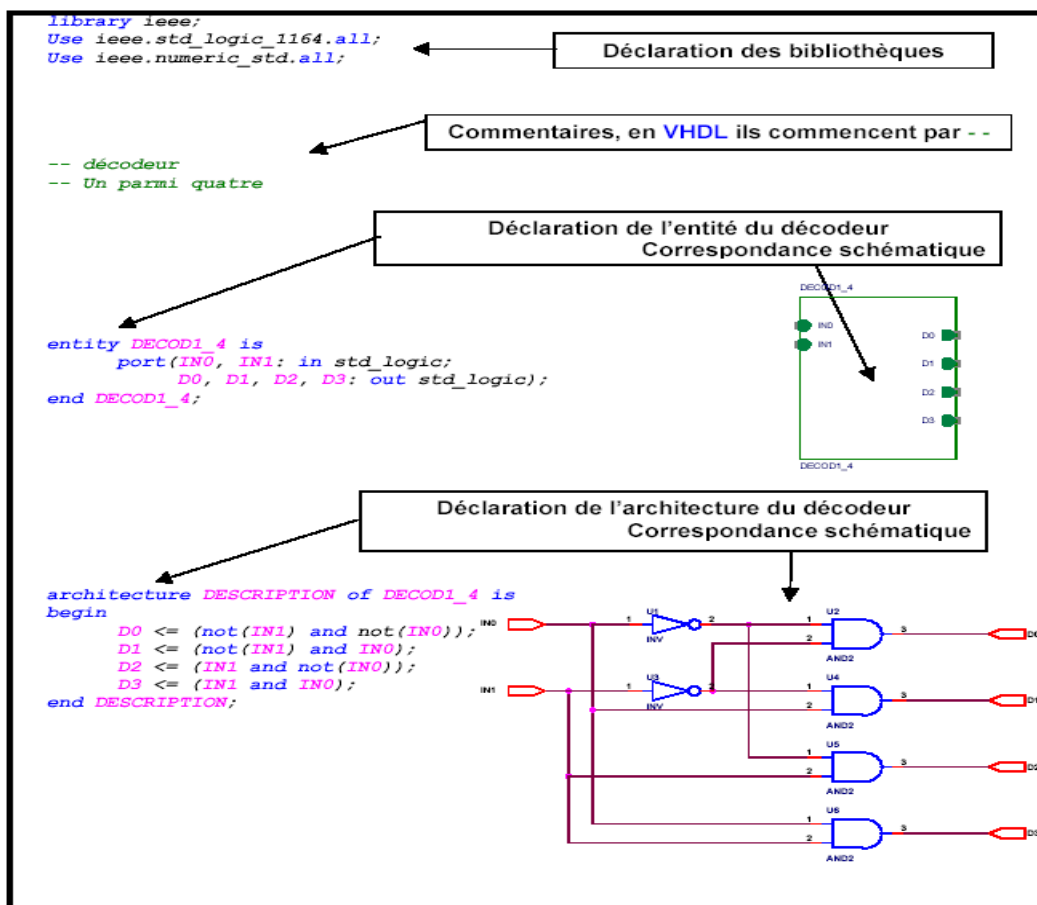
- Lors de l'étape de simulation, on valide l'application, indépendamment de l'architecture et des temps de propagation du futur circuit cible.
- L'étape de routage génère les informations permettant d'intégrer l'application dans le circuit choisi. Une « métro annotation » est effectuée dans la « netlist » : les temps de propagation du composant cible y sont pris en compte.
- Lors de la simulation temporelle, on peut évaluer les performances de l'application générée.

4.1. La structure d'une description VHDL

Une description VHDL est composée de deux parties indissociables à savoir :

- L'entité (ENTITY), elle définit les entrées et sorties.
- L'architecture (ARCHITECTURE), elle contient les instructions VHDL permettant de réaliser le fonctionnement attendu ; donc on y trouve des descriptions procédurales, fonctionnelles et structurelles. Plusieurs architectures peuvent être associées à une seule entité.

Nous illustrerons cette structure dans un exemple de programmation d'un décodeur (1 parmi 4) comme suit :



4.2. Déclaration des bibliothèques

Toute description VHDL utilisée pour la synthèse a besoin de bibliothèques. L'IEEE (Institut of Electrical and Electronics Engineers) les a normalisées et plus particulièrement la bibliothèque IEEE1164. Elles contiennent les définitions des types de signaux électroniques, des fonctions et sous programmes permettant de réaliser des opérations arithmétiques et logiques,...

La directive *Use* permet de sélectionner les bibliothèques à utiliser.

```
Library ieee;
Use ieee.std_logic_1164.all;
Use ieee.numeric_std.all;
Use ieee.std_logic_unsigned.all;
```

4.3. Déclaration de l'entité et des entrées/sorties

Elle permet de définir le NOM de la description VHDL ainsi que les entrées et sorties utilisées. L'instruction port permet de définir les entrées/sorties. On doit définir pour chaque signal : le nom du signal, le sens et le type. Sa syntaxe sera donc :
 NOM_DU_SIGNAL : sens type;

Le NOM_DU_SIGNAL : Il est composé de caractères, le premier caractère doit être une lettre, sa longueur est quelconque, mais elle ne doit pas dépasser une ligne de code. Le VHDL n'est pas sensible à la « casse », c'est-à-dire qu'il ne fait pas la distinction entre les majuscules et les minuscules.

Le SENS du signal : peut être soit **in** pour un signal en entrée, soit **out** pour un signal en sortie, soit **inout** pour un signal en entrée/sortie ou alors **buffer** pour un signal en sortie mais utilisé comme entrée dans la description.

Le TYPE : utilisé pour les signaux d'entrées / sorties est :

- Le **std_logic** pour un signal.
- Le **std_logic_vector** pour un bus composé de plusieurs signaux.

Il faut toujours clore la déclaration de l'entité par un **end**. Sa syntaxe est :

```
entity NOM_DE_L_ENTITE is
port ( Description des signaux d'entrées /sorties ...);
end NOM_DE_L_ENTITE;
```

Exemple de déclaration d'entité :

```
entity SEQUENCEMENT is
port (
CLOCK : in std_logic;
RESET : in std_logic;
Q : out std_logic_vector(1 downto 0)
);
end SEQUENCEMENT;
```

4.4. Déclaration de l'architecture – description du fonctionnement

L'architecture décrit le fonctionnement souhaité pour un circuit ou une partie du circuit. En effet le fonctionnement d'un circuit est généralement décrit par plusieurs modules VHDL. Il faut comprendre par module le couple Entité/Architecture. L'architecture établit à travers les instructions les relations entre les entrées et les sorties. On peut avoir un fonctionnement purement combinatoire, séquentiel ou même les deux séquentiel et combinatoire.

Exemple de description VHDL des opérateurs de base :

```
entity PORTES is
port (A,B :in std_logic;
Y1,Y2,Y3,Y4,Y5,Y6,Y7:out std_logic);
end PORTES;
architecture DESCRIPTION of PORTES is
begin
Y1 <= A and B;
Y2 <= A or B;
Y3 <= A xor B;
Y4 <= not A;
Y5 <= A nand B;
Y6 <= A nor B;
Y7 <= not(A xor B);
end DESCRIPTION;
```

4.5. Les instructions de base de la logique combinatoire

Pour une description VHDL toutes les instructions sont évaluées et affectent les signaux de sortie en même temps. L'ordre dans lequel elles sont écrites n'a aucune importance. En effet la description génère des structures électroniques, c'est la grande différence entre une description VHDL et un langage informatique classique. Dans un système à microprocesseur, les instructions sont exécutées les unes à la suite des autres. Avec VHDL il faut essayer de penser à la structure qui va être générée par le synthétiseur pour écrire une bonne description, cela n'est pas toujours évident.

L'affectation simple (\leftarrow) est l'opérateur le plus utilisé, dans une description VHDL. Il permet de modifier l'état d'un signal en fonction d'autres signaux et/ou d'autres opérateurs.

Exemple: $S1 \leftarrow E2 \text{ and } E1$;

L'opérateur de concaténation ($\&$) permet de joindre des signaux entre eux.

Exemple : Soit A et B de type 3 bits et S1 de type 8 bits ; $A = "001"$ et $B = "110"$

$S1 \leftarrow A \& B \& "01"$; après cette affectation $S1 = "001110 01"$.

Les opérateurs arithmétiques :

Opérateur	VHDL
ADDITION	+
SOUSTRACTION	-
MULTIPLICATION	*
DIVISION	/

Les opérateurs logiques :

Opérateur	VHDL
ET	and
NON ET	nand
OU	or
NON OU	nor
OU EXCLUSIF	xor
NON OU EXCLUSIF	xnor
NON	not
DECALAGE A GAUCHE	sll
DECALAGE A DROITE	srl
ROTATION A GAUCHE	rol
ROTATION A DROITE	ror

Les Opérateurs relationnels permettent de modifier l'état d'un signal ou de signaux suivant le résultat d'un test ou d'une condition. En logique combinatoire ils sont souvent utilisés avec les instructions :- when ... else ...- with select

Opérateur	VHDL
Egal	=
Non égal	/=
Inférieur	<
Inférieur ou égal	<=
Supérieur	>
Supérieur ou égal	>=

L'Affectation conditionnelle modifie l'état d'un signal suivant le résultat d'une condition logique entre un ou des signaux, valeurs, constantes.

```
SIGNAL <= expression when condition
[else expression when condition]
[else expression];
```

Exemple de la structure d'un multiplexeur 4 vers 1

```
S2 <= E1 when (SEL="00" ) else
E2 when (SEL="01" ) else
E3 when (SEL="10" ) else
E4 when (SEL="11" )
else '0';
```

L'affectation sélective permet d'affecter différentes valeurs à un signal, selon les valeurs prises par un signal dit de sélection.

```
with SIGNAL_DE_SELECTION select
SIGNAL <= expression when valeur_de_selection,
[expression when valeur_de_selection,]
[expression when others];
```

Exemple Multiplexeur 4 vers 1

```
with SEL select
S2 <= E1 when "00",
E2 when "01",
E3 when "10",
E4 when "11",
'0' when others;
```

4.6. Les instructions du mode séquentiel

➤ Les processus (process) :

Un processus joue un rôle central dans la sémantique du programme puisque il présente une partie de la description d'un circuit dans laquelle les instructions sont exécutées séquentiellement.

Il permet d'effectuer des opérations sur les signaux en utilisant les instructions standard de la programmation structurée comme dans les systèmes à microprocesseurs.

L'exécution d'un processus est déclenchée par un ou des changements d'états de signaux logiques. Le nom de ces signaux est défini dans la liste de sensibilité lors de la déclaration du processus. La syntaxe générale d'un processus est donnée par :

```
[Nom_du_process :] process(Liste_de_sensibilité_nom_des_signaux)
Begin
-- instructions du process
end process [Nom_du_process] ;
```

Les changements d'état des signaux par les instructions du processus sont pris en compte à la fin du processus. Il existe deux structures utilisées très souvent dans les processus :

La structure conditionnelle dont la syntaxe est donnée comme suit :

```

if condition then
instructions
[elsif condition then instructions]
[else instructions]
end if ;

```

La structure sélective dont la syntaxe est donnée par :

```

case signal_de_sélection is
when valeur_de_sélection => instructions
[when others => instructions]
end case;

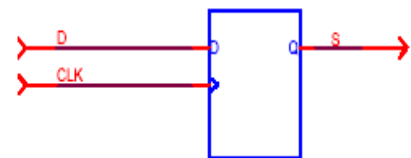
```

Exemple d'un processus de déclaration d'une bascule D :

```

Library ieee;
Use ieee.std_logic_1164.all;
Use ieee.numeric_std.all;
Use ieee.std_logic_unsigned.all;
entity BASCULED is
port (
D,CLK : in std_logic;
S : out std_logic);
end BASCULED;
architecture DESCRIPTION of BASCULED is
begin
PRO_BASCULED : process (CLK)
begin
if (CLK'event and CLK ='1') then
S <= D;
end if;
end process PRO_BASCULED;
end DESCRIPTION;

```



4.7. Les types

Le VHDL étant un langage typé tous les paramètres utilisés doivent avoir un type défini. Le type définit le format des données et l'ensemble des opérations applicables à ces données ; on distingue quatre catégories de type :

- Le type scalaire : définit les données qui n'ont pas de structure interne ; ce type est défini par un domaine de définition ou par une liste de valeurs, permettant de

limiter la taille des données lui appartenants. Ce type regroupe les entiers et les flottants.

Exemple de domaine de définition:

```
Type octet is range 0 to 255 ;
```

- Le type structuré : permet de structurer les données au moyen de tableaux. Un tableau est un ensemble d'éléments de même type.

Exemple de déclaration d'un vecteur :

```
Subtype natural is integer range 0 to integer'high;
Type bit_vector is array (natural range<>) of bit;
```

- Le type pointeur: ne correspond à aucun objet synthétisable dans le circuit. La syntaxe qui permet la déclaration de ce type est donnée comme suit :

```
Type nom_type is access type_pointe ;
```

- Le type fichier : Le fichier représente une séquence d'éléments de même type. Pour déclarer un fichier on utilise la syntaxe :

```
Type nom_de_type is file of type_des _éléments;
```

- Les constantes : permettent de paramétrer sous forme littérale les variables non modifiables au cours de l'exécution du programme.

Exemple : constant age : integer :=25;

- Les signaux : correspondent aux canaux de communication du circuit, donc ils ont un rôle central dans le langage.

Exemple : Signal temps : std_logic_vector (dimension -1 downto 0);

- Les variables : servent au stockage des valeurs nécessaires au bon déroulement d'un algorithme.

Exemple : Variable tmp : integer range 0 to 18 ;

```
tmp<=a+b ;
  if tmp>9 then
    tmp<=tmp-10;
  end if;
```


Nous avons donné dans cette partie un aperçu sur le langage VHDL et ses principes de base à savoir la procédure de déclaration ; et les syntaxes des instructions les plus utilisées.

Conclusion

Le VHDL est un langage de spécification, de simulation et également de conception. La normalisation a d'abord eu lieu pour la spécification et la simulation (1987) et ensuite pour la synthèse (1993). Etabli en premier lieu pour la spécification, c'est dans ce domaine que la norme est actuellement la mieux établie. Actuellement, il y a une nette tendance des fabricants à uniformiser le VHDL autour du VHDL de Synopsis. Il est donc probable que l'on s'approche d'un vrai VHDL standard. Le VHDL est également un langage de simulation. Pour ce faire, la notion de temps, sous différentes formes, y a été introduite. Des modules, dessinés uniquement à la simulation, peuvent ainsi être créés et utilisés pour valider un fonctionnement logique ou temporel du code VHDL.

Chapitre V

La Carte Virtex-II de Xilinx

5.1. La carte système Virtex-II

La carte système Virtex-II contient un FPGA, le support des circuits et le port d'extension P160 pour les applications spécifiques en y insérant des cartes. La figure 1 montre la carte et ses composants.

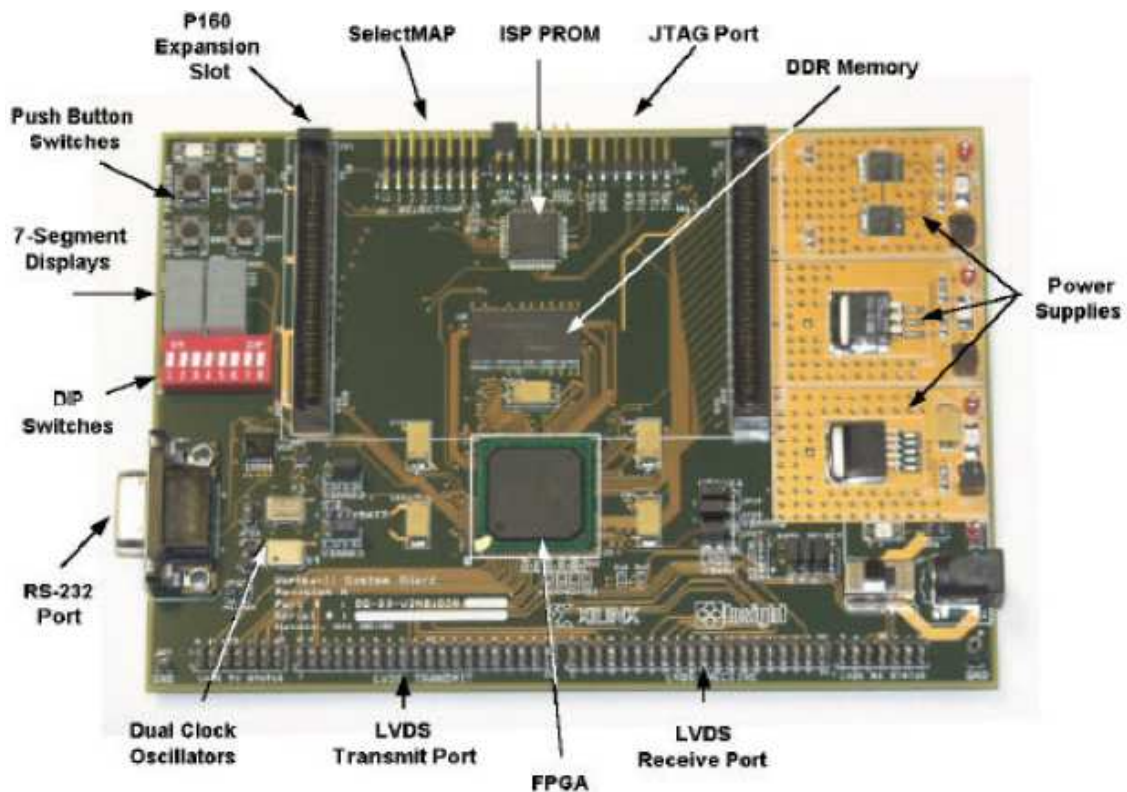


Figure 1. La carte système Virtex-II

5.1.1. Description de la carte système Virtex-II

Le schéma bloc de la carte est montré sur la figure 2 suivi d'une bref description de chaque sous bloc.

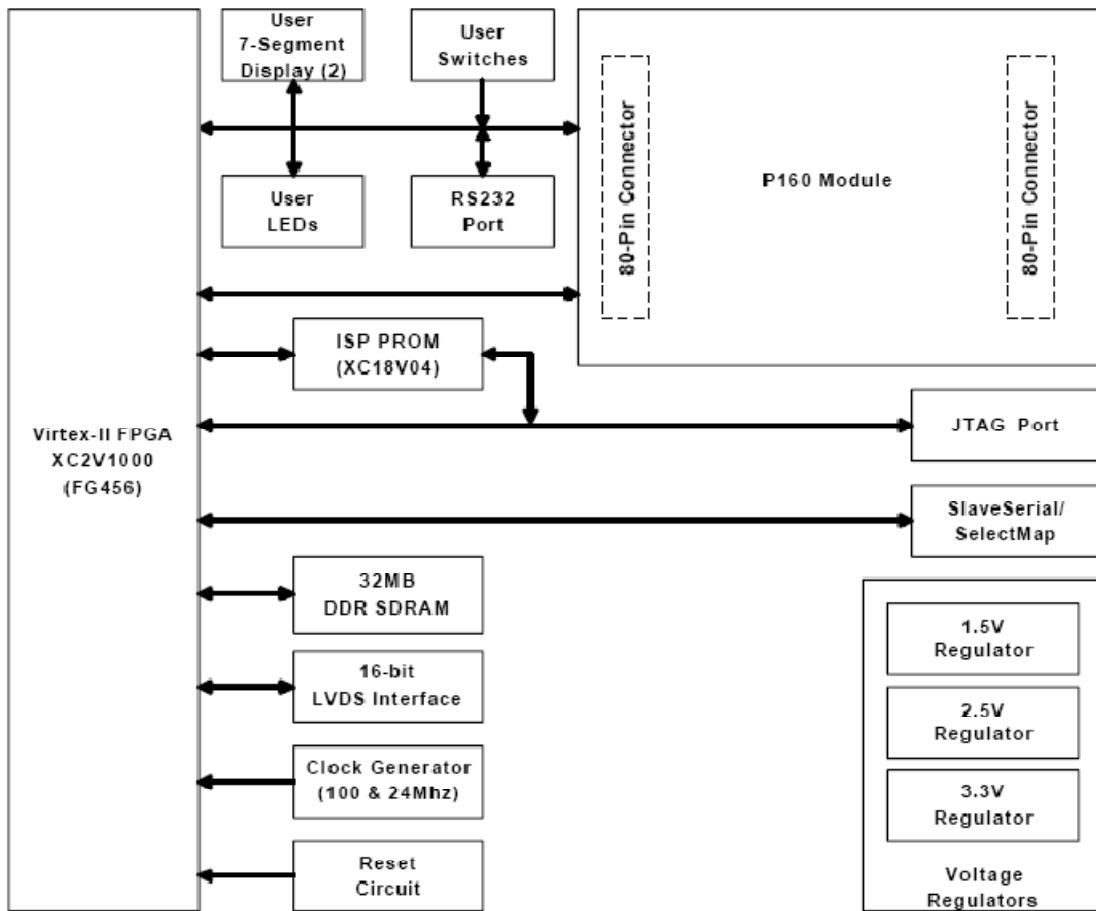


Figure 2. Le schéma bloc de la carte Virtex-II

5.1.2. La mémoire DDR

La carte de développement Virtex-II contient 32MBde mémoire DDR dans la carte système. Cette mémoire est implémentée en utilisant le Micron MT46V16M16TG-75 16Mx16 DDR. Le schéma bloc de l'interface DDR est montré en dessous de la figure 3 par une table décrivant les signaux de l'interface de la mémoire DDR.

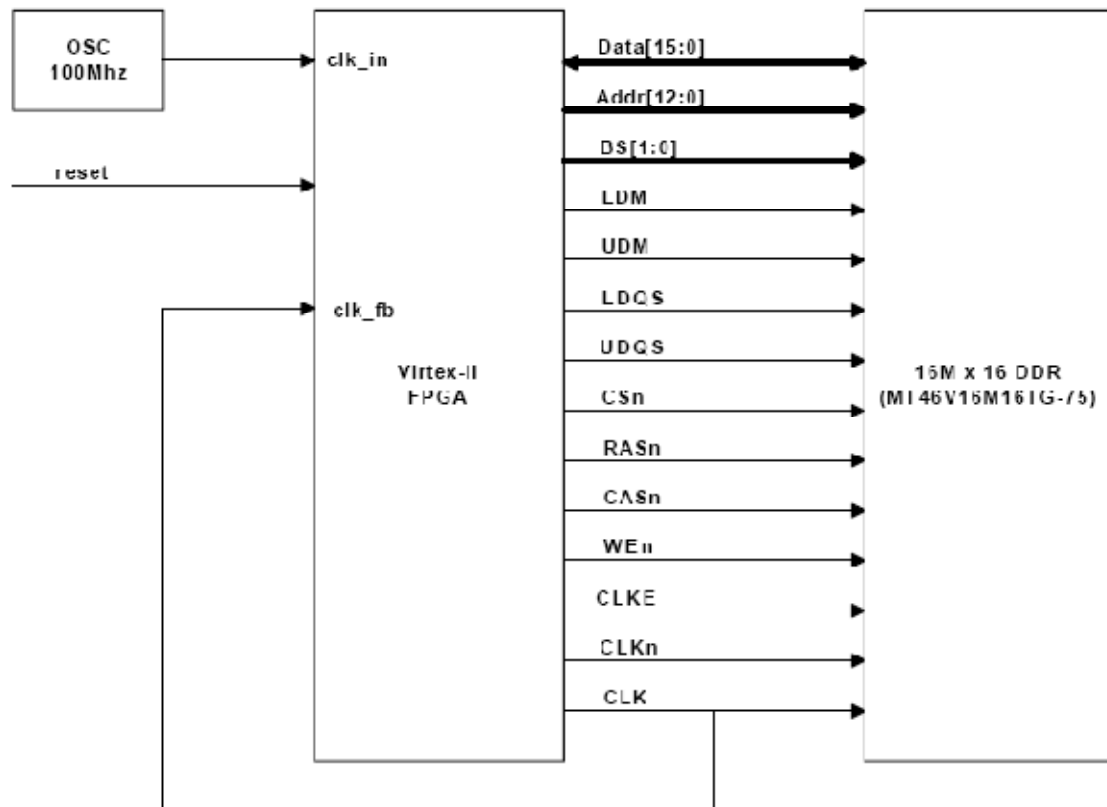


Figure 3. L'interface de la mémoire DDR

5.1.3. Génération de l'horloge

La carte système Virtex-II contient deux oscillateurs oscillant à 100 MHz (CLK CAN2) et 24 MHz (CLK CAN). L'oscillateur à 100 MHz est activé quand le JP24 est ouvert et désactivé quand le JP24 est fermé. Le JP23 contrôle l'oscillateur à 24 MHz, il l'active quand il est ouvert et le désactive quand il est fermé. La troisième prise d'horloge est ajoutée pour que l'utilisateur puisse introduire une horloge spécifiée. La table suivante donne une brève description des signaux de cette horloge.

5.1.4. Le circuit Reset

La carte Virtex-II utilise le dispositif le superviseur de voltage TI TPS3125 pour contrôler le voltage du noyau du FPGA Virtex-II (1.5 V). Ce circuit impose le signal Reset (FPGA_RESETh) au Virtex-II quand le voltage du noyau chute en dessous de sa spécification minimum (1.425 V). Le signal Reset du FPGA fixe à 100 ms la plus basse impulsion active. En plus du contrôle du voltage du noyau, ce circuit peut être utilisé pour générer un Reset en activant le signal Master Reset (MRn) au TPS3125 via le

bouton-poussoir (SW3). La figure suivante montre le circuit Reset dans la carte de développement Virtex-II.

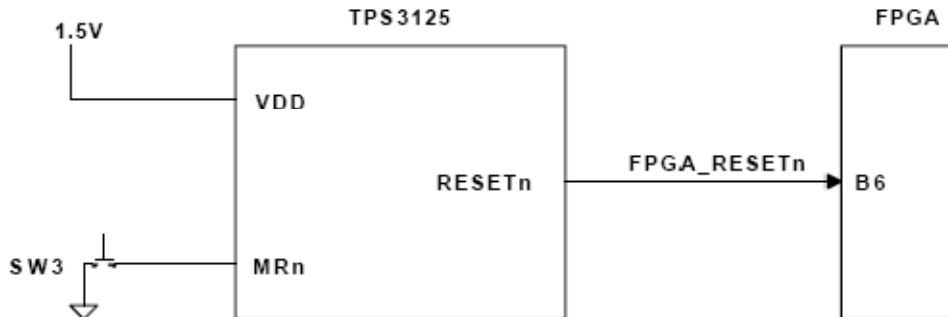


Figure 4. Le circuit Reset

5.1.5. Le port RS 232

Le Virtex-II donne un port RS232 qui peut être commandé par le FPGA Virtex-II. Le *subset* des signaux du RS232 est utilisé par la carte Virtex-II pour implémenter cette interface simple (signaux RD et TD).

Le Virtex-II donne la connexion DB-9 pour un simple port RS232. La carte utilise le driver TI MAX 2321 RS232 pour contrôler les signaux RD et TD. L'utilisateur donne le code RS232 UART qui se trouve dans le FPGA Virtex-II.

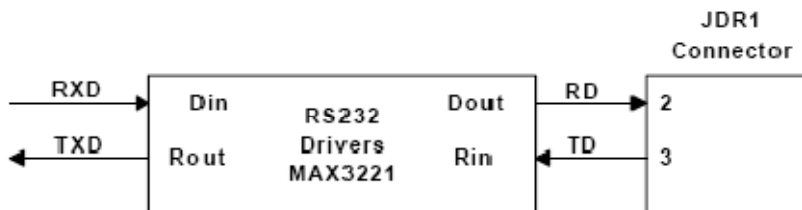


Figure 7. L'interface du RS232

Le tableau suivant donne les signaux du RS232.

Signal Name	Virtex-II Pin #	Description
RXD	A7	Received Data, RD to DB9
TXD	B7	Transmit Data, TD from DB9

Tableau 6. Les signaux du RS232

5.1.6. Le port JTAG

Le Virtex-II donne un connecteur JTAG qui peut être utilisé pour programmer le ISP PROM et configurer le FPGA Virtex-II. Deux options de connecteurs sont données, J2 est un *header* 1 x 7 utilisé pour connecter le câble JTAG standard et JP29 qui est utilisé pour la connexion du câble Xilinx Parallel IV JTAG.

La figure suivante montre le connecteur du câble Parallel IV JTAG. Ce dernier peut aussi être utilisé pour configurer le FPGA via le mode de configuration Slave Serial.

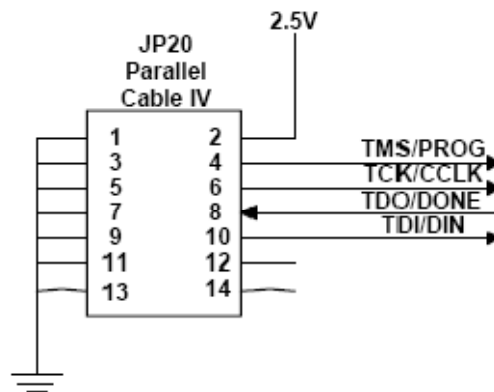


Figure 9. JP29 Parallel IV Port

La figure suivante montre la chaîne JTAG dans la carte Virtex-II. Jumper JP22 donne la capacité d'enlever le ISP PROM de la chaîne JTAG pour une connexion directe dans le FPGA.

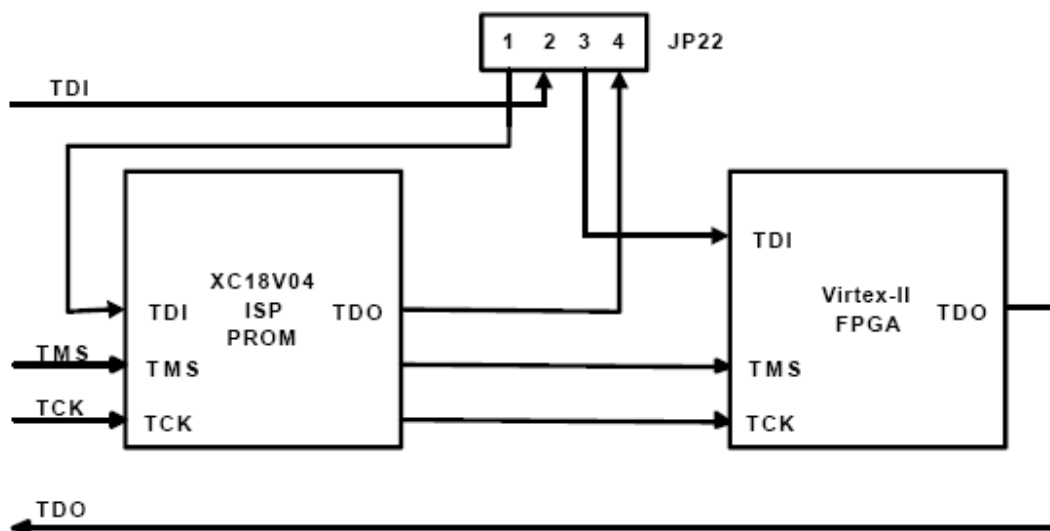


Figure 10. La chaîne JTAG du Virtex-II

Le tableau suivant montre la chaîne JTAG jumper montée sur la carte Virtex-II.

Jumper	Setting	Description
JP28	1-2 Closed	Disable PROM
	2-3 Closed	Enable PROM (normal setting)
JP22	1-2, 3-4	PROM in chain (normal setting)
	2-3	Remove PROM from chain (FPGA only)

Tableau 7. La chaîne JTAG jumper

5.1.7. Le voltage du banc d'entrée/sortie

La carte Virtex-II permet aux broches des I/O du Virtex-II d'être configurées pour une opération de 2.5 V ou 3.3 V. Toutes les broches d'entrée/sortie sont regroupées dans 8 bancs différents. Chaque broche de I/O sur la carte peut être configurée à opérer dans le mode 2.5 V ou 3.3 V.

5.1.8. ISP PROM

La carte Virtex-II utilise le Xilinx XC18V04 ISP PROM, permettant aux concepteurs des FPGA de charger rapidement les révisions qu'ils ont apportées à leurs conceptions et vérifier les changements afin de voir les besoins du système final. Le XC18V04 ISP PROM utilise deux interfaces pour accomplir la configuration du FPGA Virtex-II.

Le port JTAG dans le XC18V04 est utilisé pour programmer le PROM avec le fichier bit (*bit file*) du modèle conçu. Une fois le XC18V04 programmé, l'utilisateur peut configurer le Virtex-II dans le Master Serial ou dans le mode Master SelectMap. La configuration du Virtex-II est initiée en imposant le signal PROGn. Durant l'activation du signal PROGn (en pressant le switch SW2), le XC18V04 utilisera son port de configuration du FPGA afin de configurer le FPGA Virtex-II.

Le mode de configuration du Virtex-II est mis au Master Serial, les signaux PROM D0, CE, CCLK, RESET/OE et CF sont utilisés pour configurer le FPGA Virtex-II. Dans le mode Master SelectMap, en plus des signaux précédents, le FPGA Virtex-II utilisera le PROM D1-7 pour obtenir l'octet de la configuration de données durant chaque cycle d'horloge CCLK. La figure suivante montre l'interface du ISP PROM aux port JTAG et au port de configuration du FPGA Virtex-II.

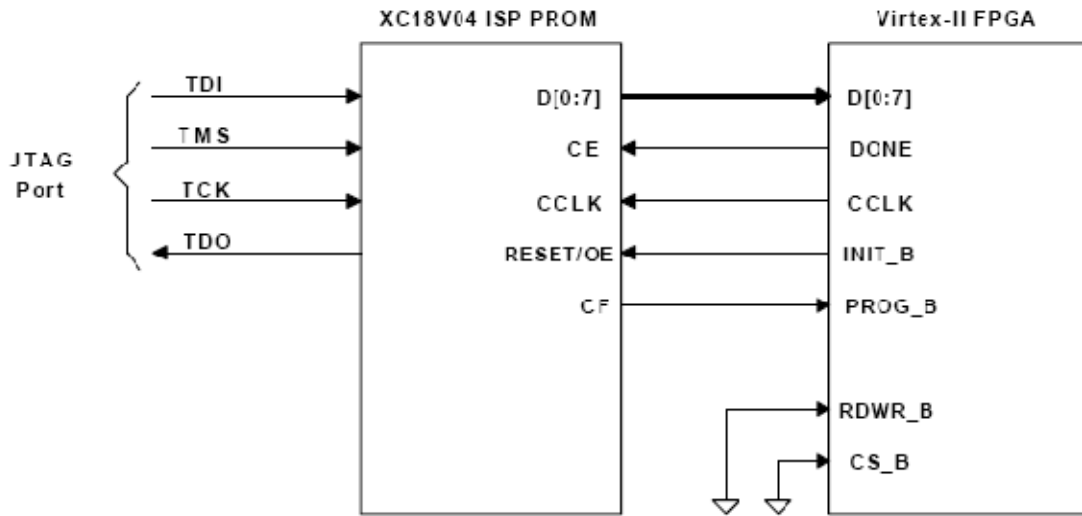


Figure 16. L'interface de l'ISP PROM

5.2. Chargement des conceptions

La carte de développement Virtex-II supporte plusieurs méthodes de configuration du FPGA du Virtex-II. Le port JTAG de la carte peut être utilisé pour configurer le FPGA du Virtex-II directement, ou pour programmer le XC18V04 ISP PROM. Une fois le ISP PROM programmé, il peut être utilisé pour configurer le FPGA du Virtex-II. Le port SelectMap/Slave Serial sur cette carte peut aussi être utilisé pour configurer le FPGA du Virtex-II. La figure suivante montre l'organisation de tous les modes de configuration du FPGA Virtex-II que la carte de développement Virtex-II supporte.

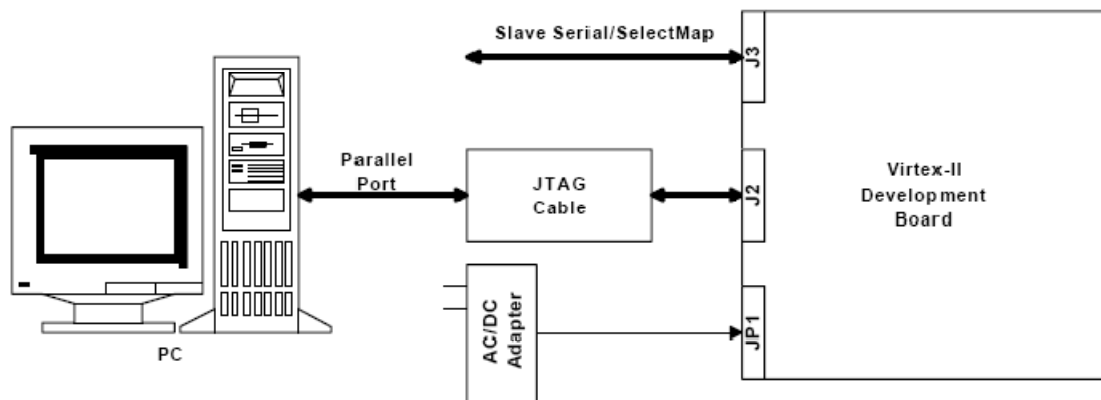


Figure 22. L'organisation du chargement

Chapitre VI

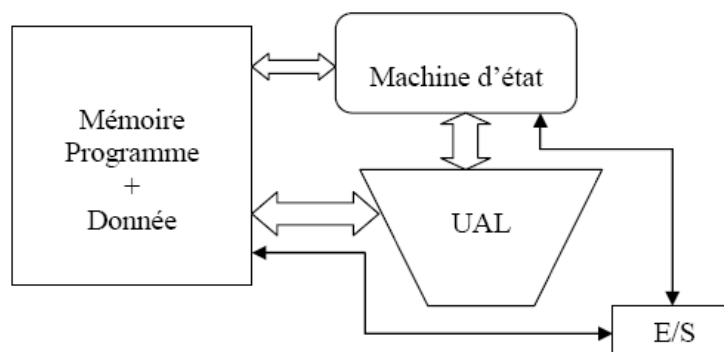
Programmation et implémentation

Introduction

Dans ce chapitre nous allons présenter les différentes les différentes méthodes que nous avons proposées pour l'implémentation de notre algorithme et les avantages et les inconvénients de celles-ci. Nous allons aussi expliquer le fonctionnement des blocs de base que nous avons utilisés avec les simulations de ceux-ci pour mieux illustrer la fonction de chacun d'eux.

6.1. Les différentes architectures proposées

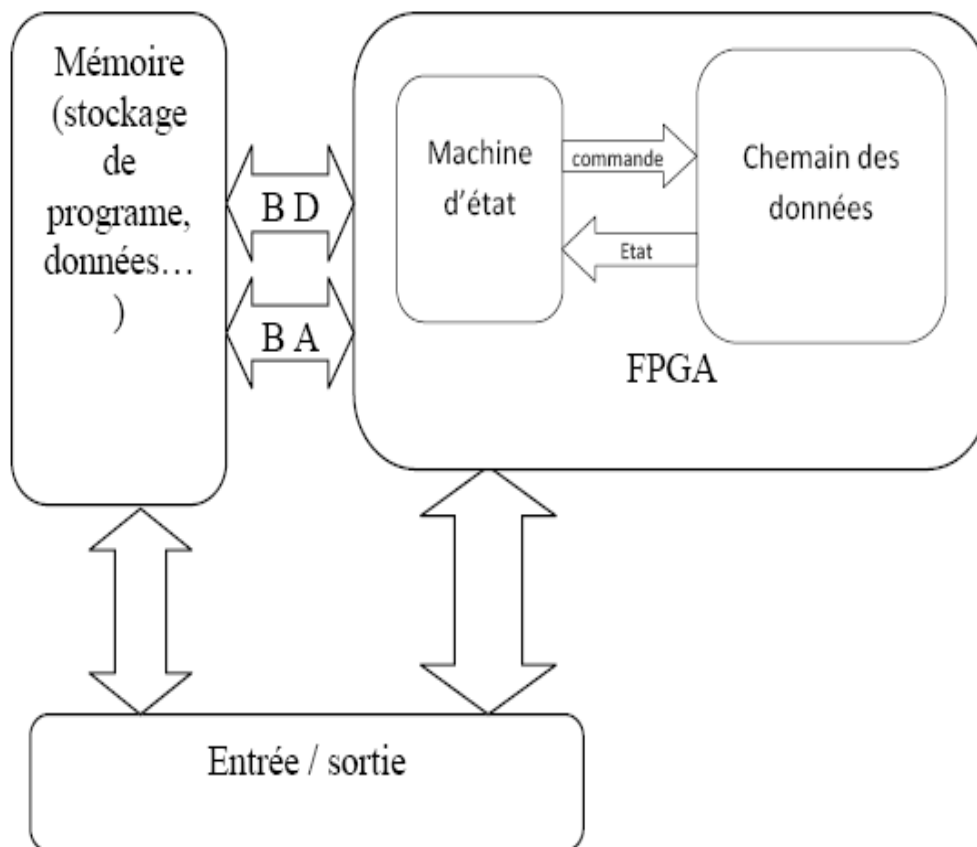
1^{ère} proposition :



On a d'abord proposé le schéma synoptique représenté ci-dessus qui est constitué d'une machine d'état pour gérer la mémoire, une unité de calcul et des entrées sorties. Ce modèle est conforme avec l'architecture d'un DSP ou de n'importe quel autre processeur classique. L'implémentation d'un algorithme selon ce modèle d'architecture donne une précision suffisante pour différentes applications. Or, les DSP ont une double précision et coûtent beaucoup moins cher que les FPGA pour un cas pareil. Donc il n'est pas intéressant d'utiliser les FPGA avec cette architecture, il faut donc penser à une autre architecture plus intéressante.

2^{ème} proposition :

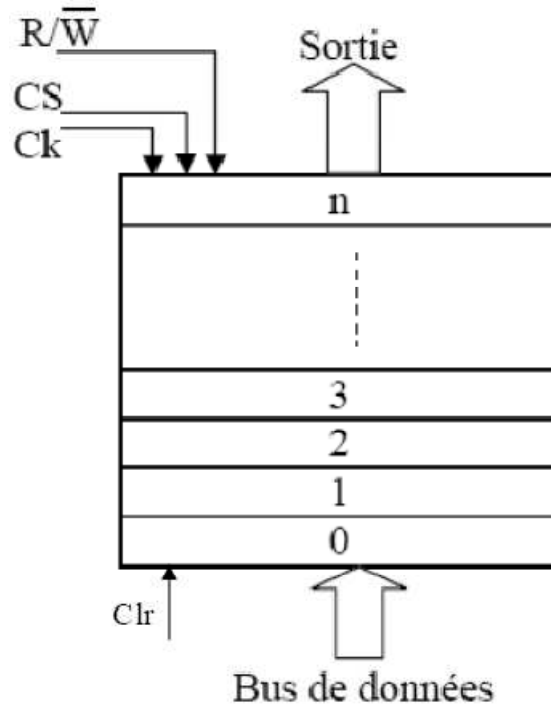
Comme nous avons déjà vu dans l'architecture précédente, on a besoin d'une autre architecture. Pour cela on a proposé d'utiliser le parallélisme pour notre application pour améliorer la rapidité et élargir les domaines d'utilisation. La figure ci-dessous représente le schéma synoptique proposé.



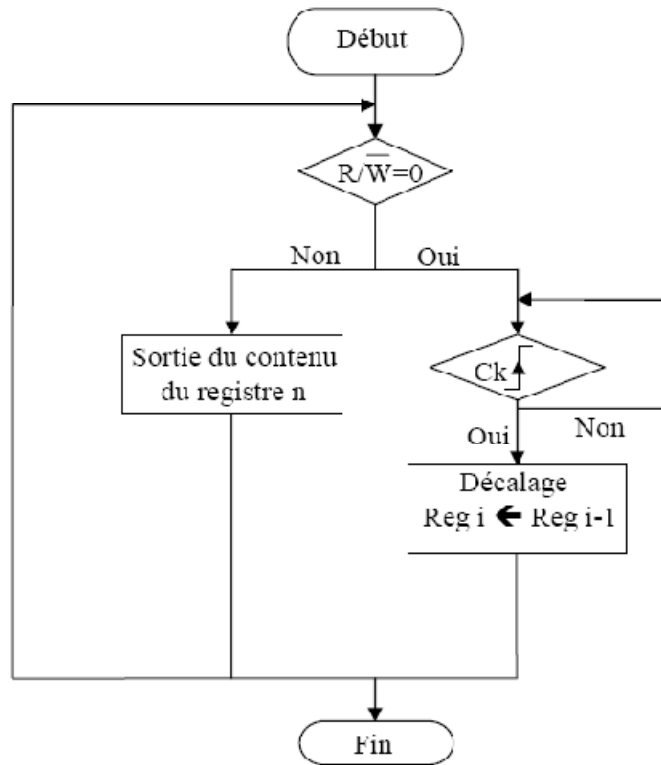
Le programme et les données sont stockés dans la mémoire RAM. Cette dernière communique avec le FPGA à travers les deux bus d'adresses et de données. Dans le FPGA, on implémente les différents blocs de calcul gérés par des machines d'état qui elles-mêmes sont gérées par une machine d'état principale. Notons que le rôle d'une machine d'état est la gestion de plusieurs blocs ; c'est-à-dire, elle donne l'ordre à un bloc X de travailler quand par exemple un bloc Y a fini sa tâche. La finalité étant de réaliser une fonction donnée.

6.2. Les blocs élémentaires utilisés pour l'implémentation

6.2.1. Le registre FIFO

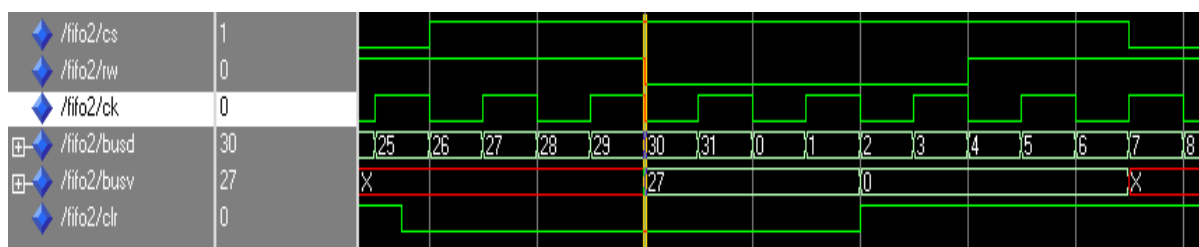


Le registre à décalage FIFO est un empilement de n registres de 8 bits chacun. Le CS indique si le registre est actif 1 ou inactif 0. Le R/\overline{W} indique si le registre est en écriture 0 ou en lecture 1. Le Ck est l'horloge. Le registre fonctionne selon l'organigramme suivant :

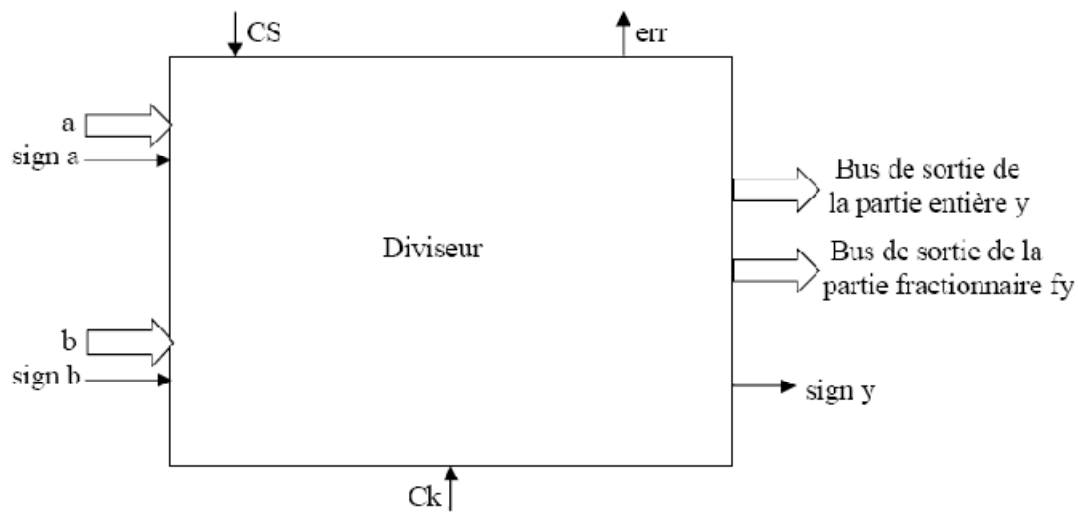


On considère que le registre est actif ($CS = 1$). Quand le $R/\overline{W} = 0$, on est en écriture ; et si en plus on a un front montant sur l'horloge, le registre FIFO reçoit une donnée sur 8 bits du bus de données qu'il stocke dans le premier empilement et de ce fait la donnée précédente va monter à l'empilement en dessus. Il y aura donc un décalage. Quand le $R/\overline{W} = 1$, on est en lecture et le contenu du dernier empilement n va aller en sortie. Notons que le registre est aussi contrôlé par un signal Clr qui lui parvient de la machine d'état et qui ordonne sa réinitialisation.

La figure suivante nous montre un exemple de simulation du registre FIFO.

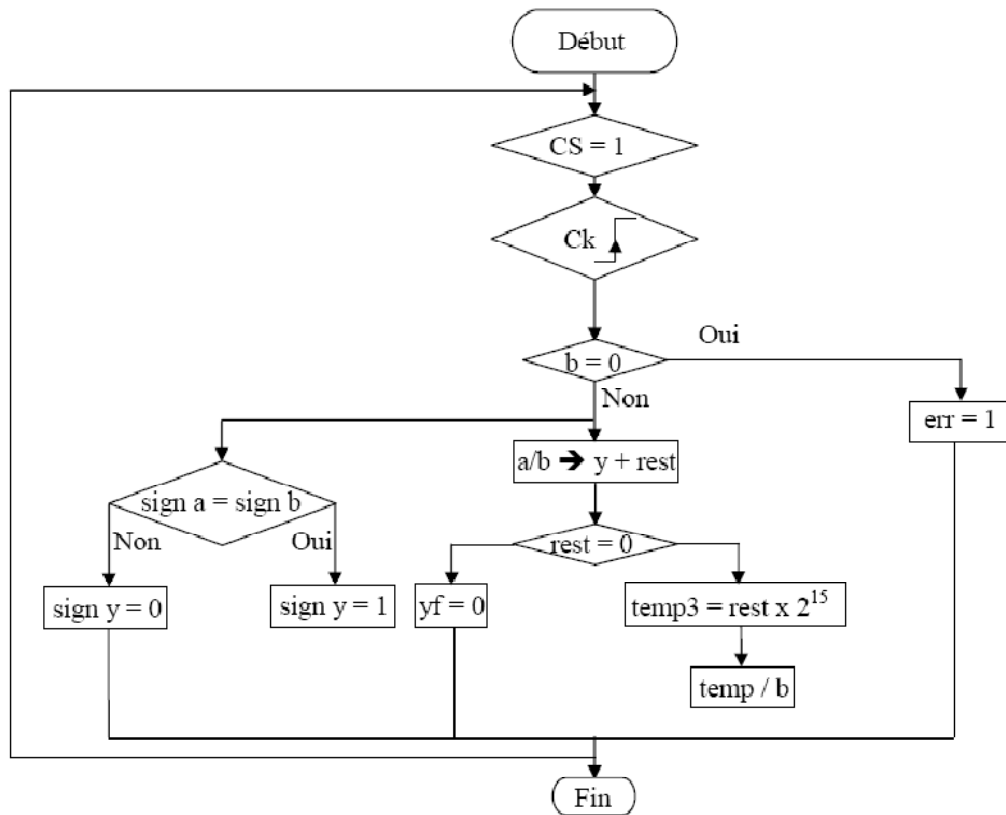


6.2.2. Le diviseur



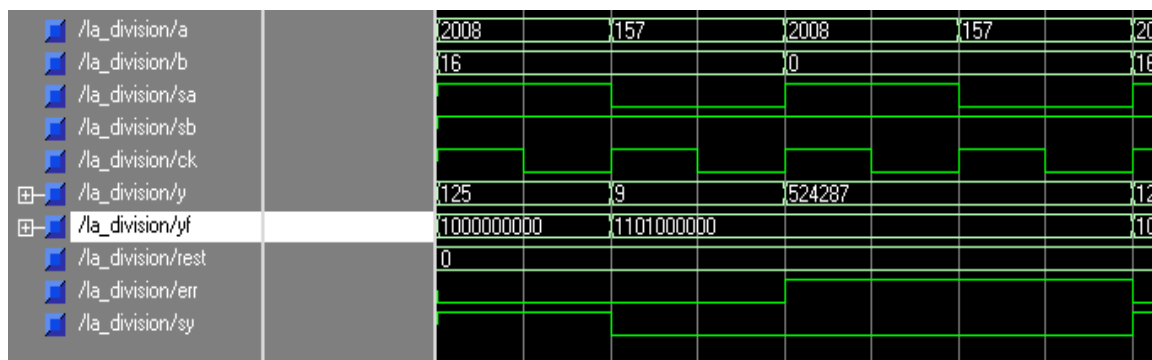
Le CS indique si le diviseur est actif 1 ou inactif 0. Le Ck est l'horloge. Les entrées a et b sont deux entiers signés de signes sign a et sign b respectivement. Quand le nombre est positif sign=0 et quand il est négatif sign=1. Au front montant de l'horloge, le diviseur fait l'opération a/b et nous donne le résultat en deux parties : entière y et fractionnaire fy et le signe du résultat de la division sign y.

L'organigramme suivant nous explique le fonctionnement du diviseur.

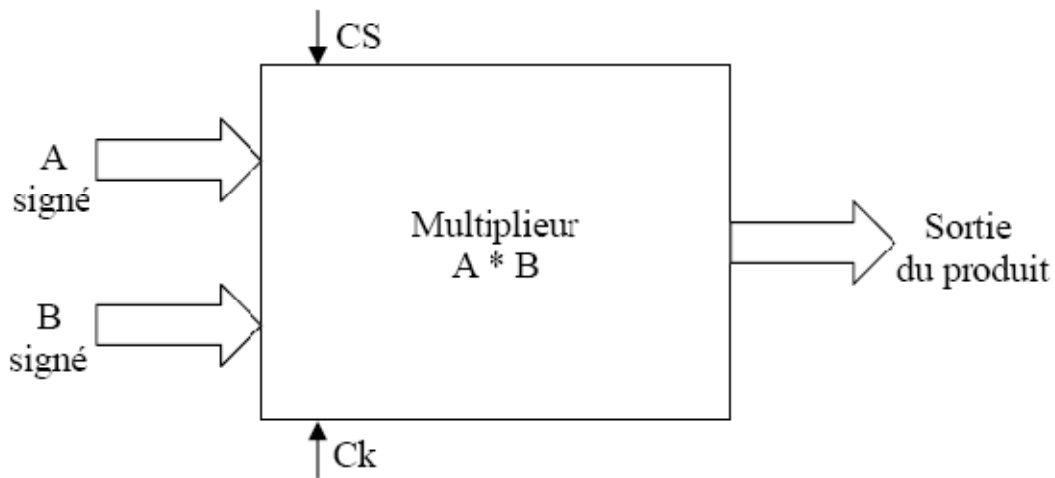


Si le CS=1 et de plus on a un front montant dans l'horloge, le diviseur calcule a/b. A la sortie, il nous donne la partie entière y et un reste. Ce dernier est multiplié par 2ⁿ avec n le nombre bits qu'on veut obtenir pour la partie fractionnaire. Le produit est, alors, divisé par b. Le résultat nous donne la partie fractionnaire fy.

La figure suivante nous montre un exemple de simulation du diviseur.

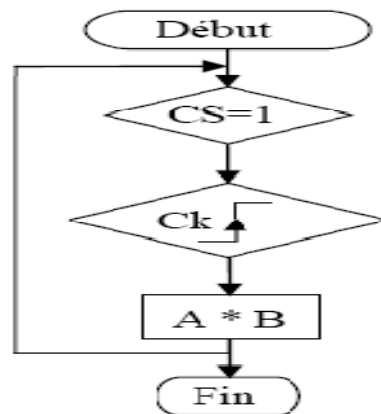


6.2.3. Le multiplieur



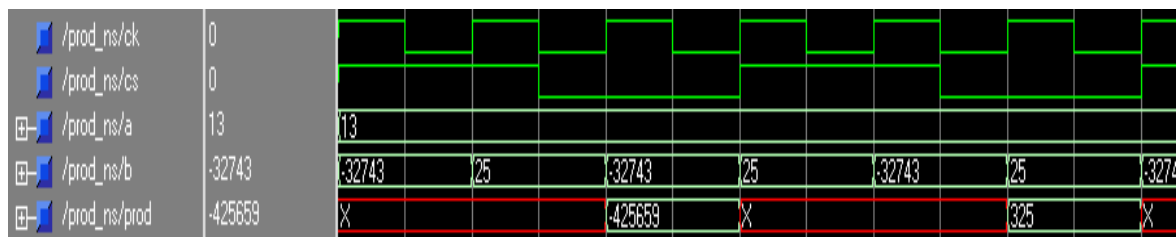
Le multiplieur fait le produit de nombres signés A et B. Le CS sert à indiquer l'état du multiplieur, actif quand CS=1 et inactif quand CS=0. L'opération de la multiplication se fait quand on a un front montant dans l'horloge Ck.

Le multiplieur fonctionne selon l'organigramme suivant.

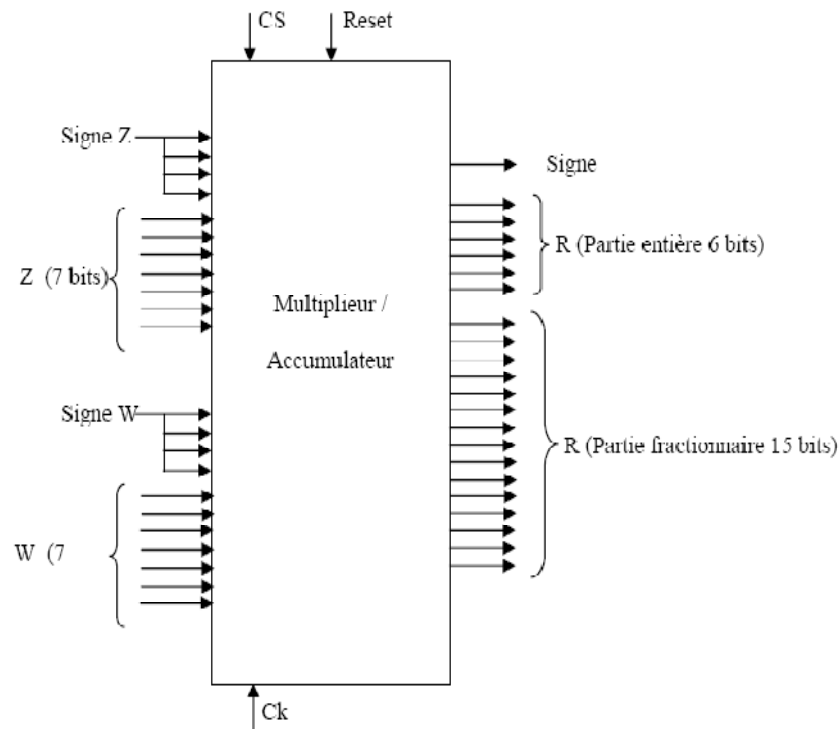


Si le CS=1 et de plus on a un front montant dans l'horloge, le multiplieur fait l'opération A*B. Le signe de la sortie dépend des signes des deux nombres, 0 s'ils ont un même signe et 1 s'ils sont de signes différents.

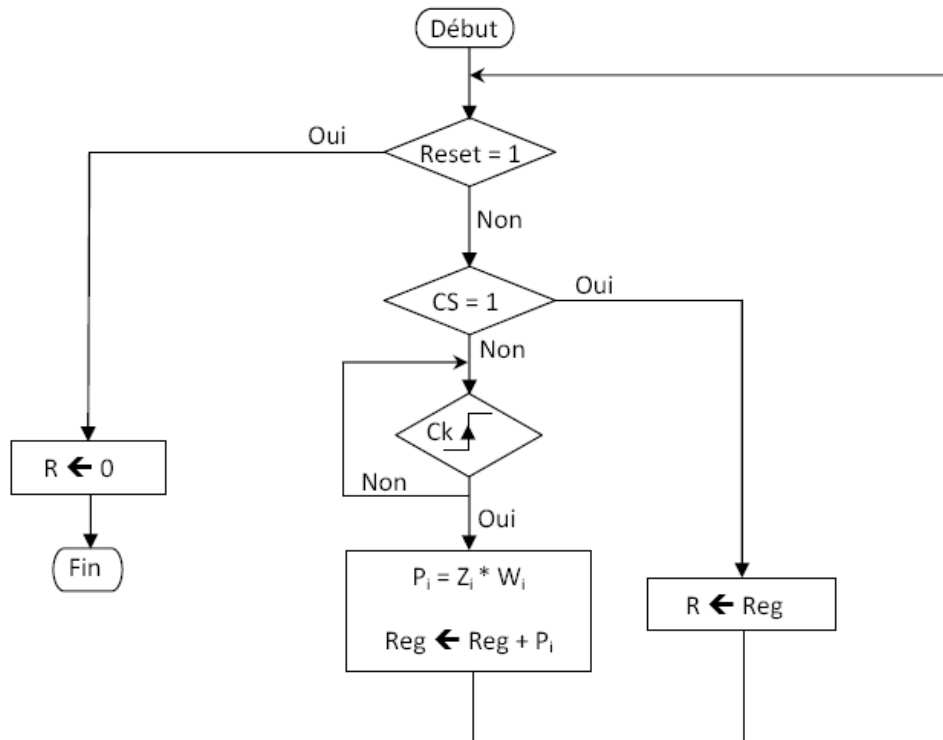
La figure suivante nous donne un exemple de simulation du multiplieur.



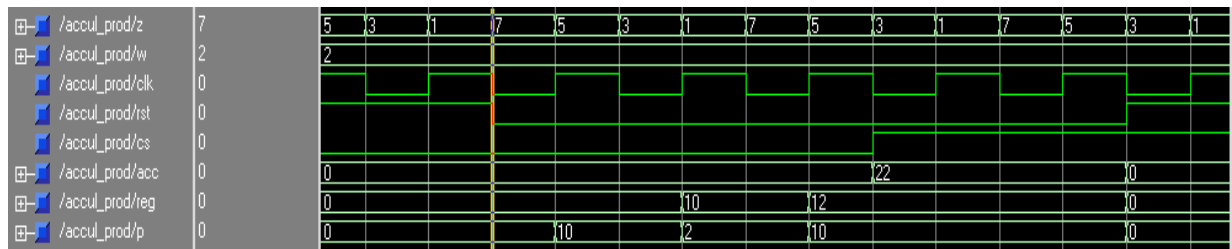
6.2.4. Le multiplieur/accumulateur



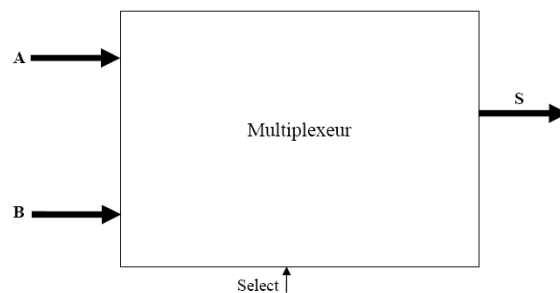
A l'entrée, on a deux nombres signés Z et W sur 7 bits + 4 bits pour le signe. Les 4 bits du signe sont reliés entre eux : positif "0000" et négatif "1111". On a, exprès, affecté 4 bits pour le signe pour avoir chaque nombre sur 11 bits histoire d'uniformiser les calculs à l'intérieur du multiplieur/accumulateur. Notons que les deux entrées Z et W sont uniformisés par rapport au maximum ; c'est-à-dire qu'elles sont inférieures ou égales à 1. Le multiplieur/accumulateur a, comme son nom l'indique, deux fonctions : il fait les multiplications des $Z_i * W_i = P_i$ et il accumule, au fur et à mesure, les P_i . A la sortie, on aura un nombre R sur 22 bits : 1 bit pour le signe + 6 bits pour la partie entière + 15 bits pour la partie fractionnaire. Sachant que le nombre d'échantillons est 80 et que les P_i sont normalisés ; on ne pourra, donc, pas avoir à la sortie un nombre supérieur à 80. C'est pour cela qu'on a affecté à la partie de la sortie entière 6 bits au maximum. L'organigramme suivant illustre le principe de fonctionnement du multiplieur/accumulateur.



La figure suivante nous montre un exemple de simulation du multiplieur/additionneur.

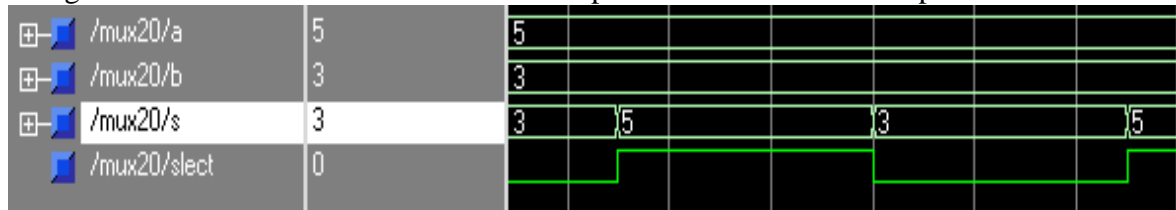


6.2.5. Le multiplexeur



Quand Select=1, la sortie S reçoit A ; et quand Select=0, elle reçoit B.

La figure suivante nous donne montre un exemple de simulation du multiplexeur.

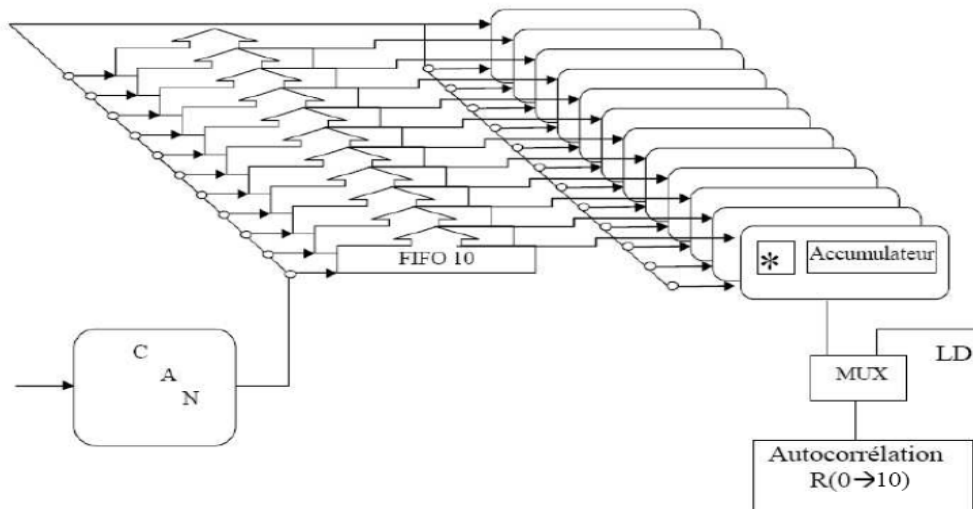


6.3. Les blocs fonctionnels

6.3.1. Le bloc d'autocorrélation

6.3.1.1. Le schéma synoptique

La figure suivante montre le principe de fonctionnement de ce bloc. Le but de ce dernier est le calcul des éléments de la matrice d'autocorrélation $R(i)$, avec $i = 0, \dots, 10$.



Le signal analogique rentre dans le convertisseur analogique/numérique pour être numérisé. Chaque échantillon étant codé sur 8 bits. A la sortie du CAN, l'échantillon $x(n)$ entre simultanément dans tous les registres à décalage FIFO1 jusqu'à FIFO10. Les registres FIFO contiennent respectivement 1 jusqu'à 10 empilements de 8 bits. En d'autres termes FIFO1 ne peut contenir qu'un échantillon à la fois alors que FIFO10 peut contenir jusqu'à 10 échantillons.

Dans les blocs de multiplication/accumulation ça se passe comme suit :

Le premier multiplieur reçoit les $x(i)$ de FIFO1 et il les multiplie par eux-mêmes, avec $i = 0, \dots, 79$.

Le deuxième multiplieur reçoit les $x(i)$ et $x(i+1)$ de FIFO2 et il les multiplie entre eux, avec $i=0, \dots, 78$.

Le troisième multiplieur reçoit les $x(i)$ et $x(i+2)$ de FIFO3 et il les multiplie entre eux, avec $i=0, \dots, 77$.

Et ainsi de suite jusqu'au dernier multiplieur qui fait le produit de $x(i)$ et $x(i+10)$, avec $i=0, \dots, 69$.

A chaque multiplieur est associé un accumulateur qui a pour rôle l'addition de tous les éléments calculés par le multiplieur.

Ainsi, par exemple, le premier accumulateur nous donnera

$$x^2(0) + x^2(1) + \dots + x^2(79) = R(0).$$

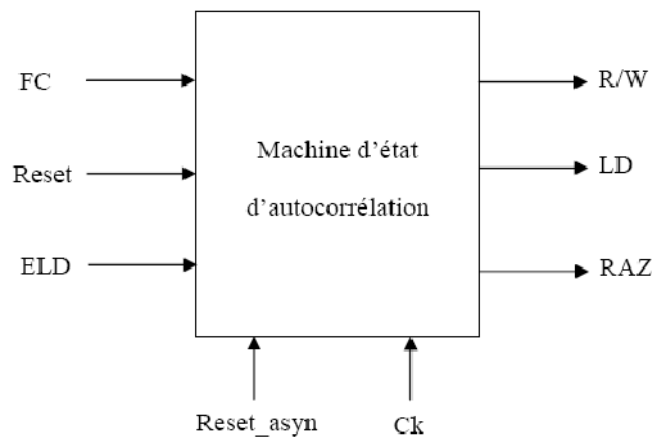
Et ainsi de suite jusqu'au dernier accumulateur qui nous donnera

$$R(10) = x(0).x(10) + x(1).x(11) + \dots + x(69).x(79).$$

A partir de là, on aura calculé tous les éléments de la matrice d'autocorrélation. Ces éléments passeront à l'étape suivante où il sera question de l'application de l'algorithme de Levinson-Durbin.

6.3.1.2. La machine d'état

Le schéma montre les différents signaux de la machine d'état du bloc d'autocorrélation.



FC : provient du convertisseur analogique/numérique, il indique que la conversion du signal analogique est terminée.

Reset : indique que 80 échantillons ont été prélevés, donc la période de stationnarité est terminée.

ELD : provient du bloc de Levinson_Durbin, il indique que ce dernier a terminé son travail.

R/W : pour la lecture et l'écriture des échantillons dans les registres FIFO.

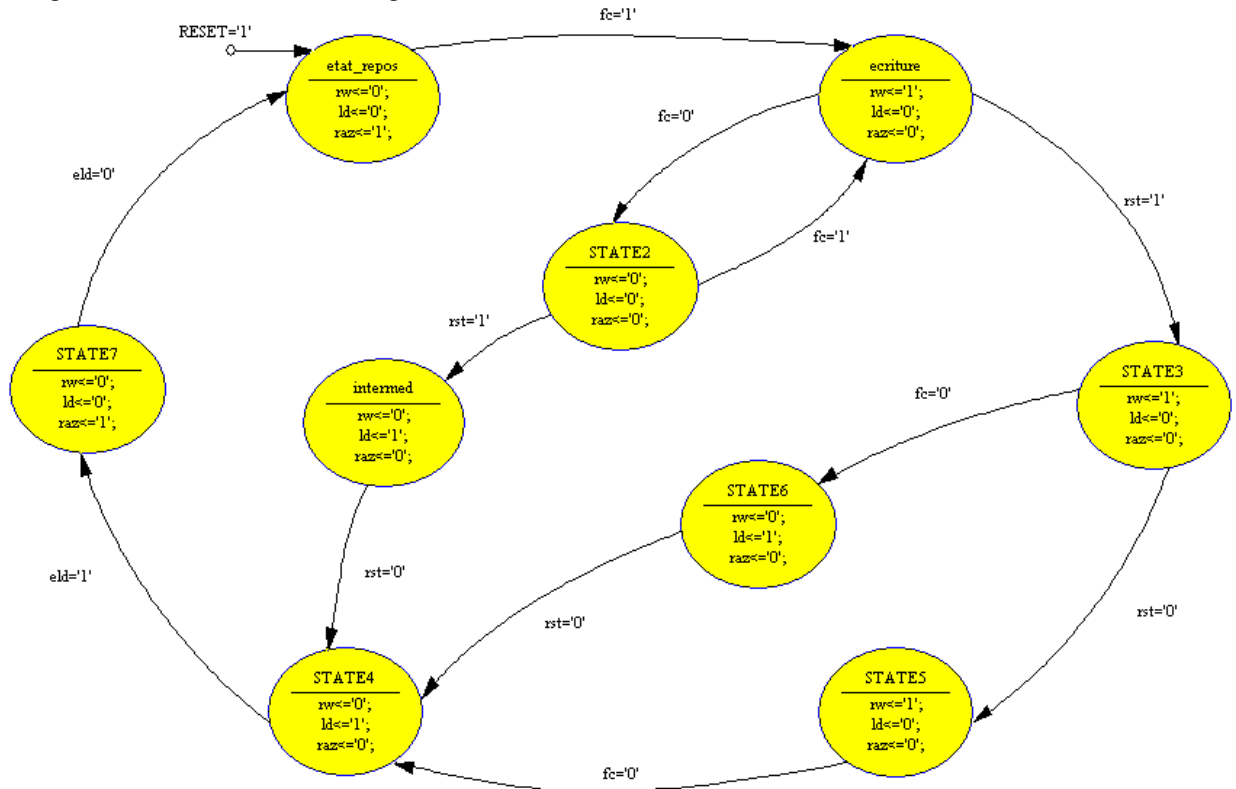
LD : pour envoyer les coefficients calculés au bloc de Levinson-Durbin.

RAZ : réinitialisation du bloc d'autocorrélation.

Reset_asyn : réinitialisation asynchrone.

Ck : horologe.

La figure suivante montre le diagramme d'état de notre machine.



6.3.2. Le bloc de Levinson-Durbin

Le but de cet algorithme est de calculer les coefficients de prédiction a_i en résolvant le système suivant :

$$\begin{bmatrix} R_0 & R_1 & R_2 & \cdots & R_9 \\ R_1 & R_0 & R_1 & & R_8 \\ R_2 & R_1 & R_0 & & R_7 \\ \vdots & & & \ddots & \vdots \\ R_9 & R_8 & R_7 & \cdots & R_0 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ \vdots \\ a_{10} \end{bmatrix} = \begin{bmatrix} R_1 \\ R_2 \\ R_3 \\ \vdots \\ R_{10} \end{bmatrix}$$

Avec les R_i ; $i=0, \dots, 10$; calculés dans le bloc précédent.

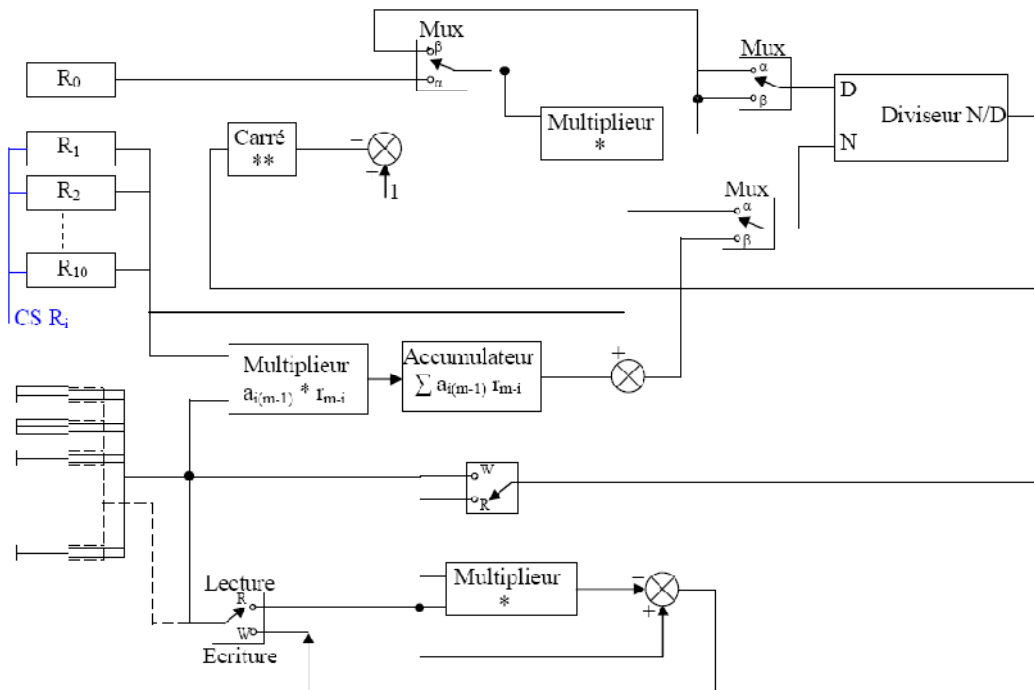
6.3.2.1. Le bloc de Levinson-Durbin par la méthode descriptive

a) Le schéma synoptique

La résolution de ce système est totalement séquentielle. L'utilisation du parallélisme ne sert à rien dans cet algorithme. Ça ne prend que de la place dans la FPGA. Alors, pour une meilleure vitesse, il faut jouer sur le chemin des données et la méthode de calcul. A cet effet, on a fait la description d'un circuit de calcul binaire. Ce dernier est, principalement, constitué de :

- ❖ Des ensembles de registres. Les éléments (registres) de chaque ensemble sont reliés entre eux avec le même bus de données mais un seul registre est actif à la fois.
- ❖ Deux types de multiplexeurs de 20 bits sont utilisés : ($\alpha\beta$) pour choisir les entrées du diviseur et (rw) écrire et lire les registres.
- ❖ Des unités arithmétiques et logiques (diviseurs, multipliers, accumulateur et additionneurs).

Le schéma suivant illustre le fonctionnement du bloc de Levinson-Durbin.



Les coefficients de prédiction se fait en trois étapes :

i. Calcul des valeurs initiales :

Au départ, les valeurs initiales a_{11} et E_1 sont calculées. Les multiplexeurs sont sur la position α et CS R_i active le registre R_1 et met les autres sont en haute impédance. Le diviseur reçoit, alors, R_0 au dénominateur D et R_1 au numérateur N ; et il fait la division $R_1 / R_0 = a_{11}$. Le a_{11} est injecté par la suite dans le bloc du carré. La sortie du carré est injectée dans l'additionneur lequel nous donnera à sa sortie la valeur $(1 - a_{11}^2)$. Ce dernier terme passe dans le multiplieur où il est multiplié par R_0 qui est égal aussi à E_0 ; on aura donc, à la fin, $E_1 = E_0 (1 - a_{11}^2)$. On aura ainsi calculé les valeurs initiales a_{11} et E_1 qui nous permettront de calculer les autres coefficients. Et à la fin de cette étape après la mémorisation de la machine d'état donne le signal pour l'étape suivante.

ii. Calcul des a_{mm} et E_m :

Une fois qu'on a calculé les valeurs initiales, on les injectes dans une autre unité de calcul qui nous permettra de calculer et stocker un coefficient a_{mm} donné à partir des coefficients $a_{i(m-1)}$ et E_{m-1} avec ($i \leq m-1$ et $m > 1$). Quand les multiplexeurs sont sur la position β , le diviseur reçoit au dénominateur E_{m-1} et au numérateur la somme $(R(m) - \sum a_{i(m-1)} R(m-i))$. Cette dernière est calculée à l'aide : d'un multiplieur qui fait les multiplications $(a_{i(m-1)} * R(m-i))$ avec $i=1 \dots m-1$ et d'un accumulateur qui additionne ces produits. Ensuite, à l'aide d'un soustracteur on calcule $(R(m) - \sum a_{i(m-1)} R(m-i))$. Enfin, en utilisant un diviseur on obtient

$$a_{mm} = (R(m) - \sum a_{i(m-1)} R(m-i)) / E_{m-1} .$$

Notons que les unités de calcul sont commandées par une machine d'état qui organise tout ce travail.

Le bus de a_{mm} relié au multiplexeur nous permet de mémoriser et de transmettre a_{mm} au sous bloc de calcul de E_m en un même cycle d'horloge.

E_m se calcule à partir du carré de a_{mm} auquel on retranche 1 et on multiplie par E_{m-1} . On aura, finalement

$$E_m = E_{m-1} (1 - a_{mm}^2).$$

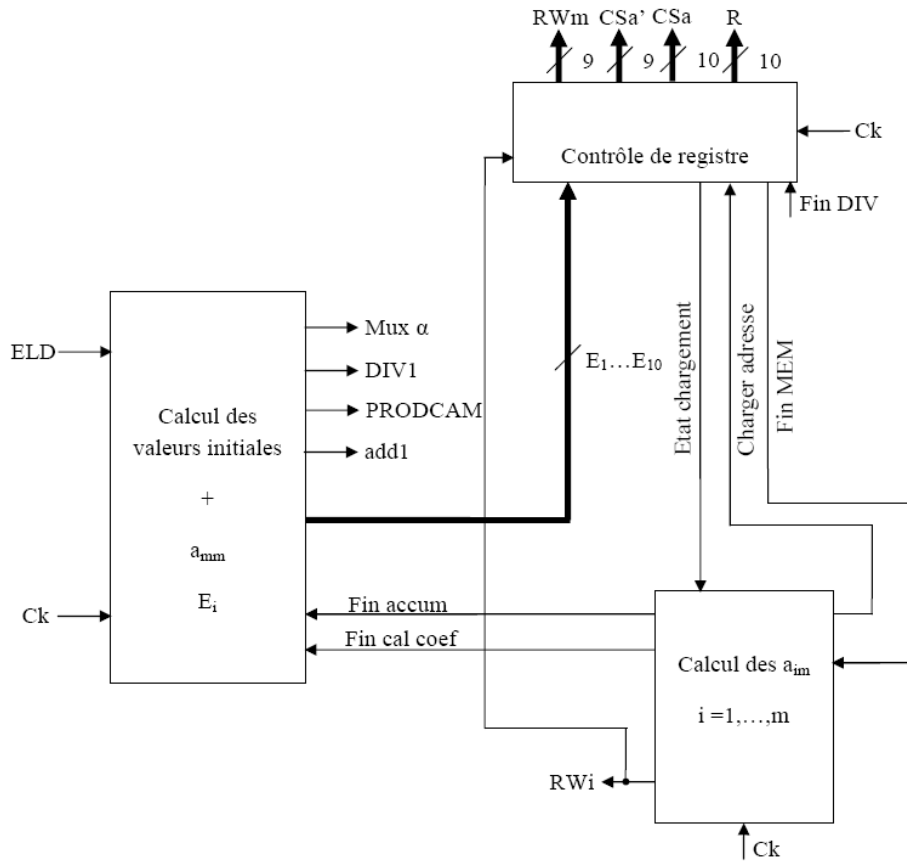
iii. Calcul des a_{im} :

Le calcul des a_{im} se fait à partir des a_{mm} et $a_{i(m-1)}$ qui son stockés dans des registres commandés par la machine d'état. Le bus de données est relié à tous les registres. D'autre part, un seul registre est actif à la fois à l'aide de signaux de commande (rw, CS). On a deux bus de données qui sont connectés au multiplieur : un pour les $a_{i(m-1)}$ et l'autre pour la sortie du multiplexeur k_m en l'occurrence. Le résultat $a_{(m-i)(m-1)}$ est retranché du coefficient $a_{i(m-1)}$; on aura à la sortie

$$a_{im} = a_{i(m-1)} - k_m a_{(m-i)(m-1)} \quad \text{avec } i \leq m-1.$$

b) La machine d'état

Le schéma suivant montre la machine d'état qu'on a conçu au départ pour gérée le bloc de Levinson-Durbin.



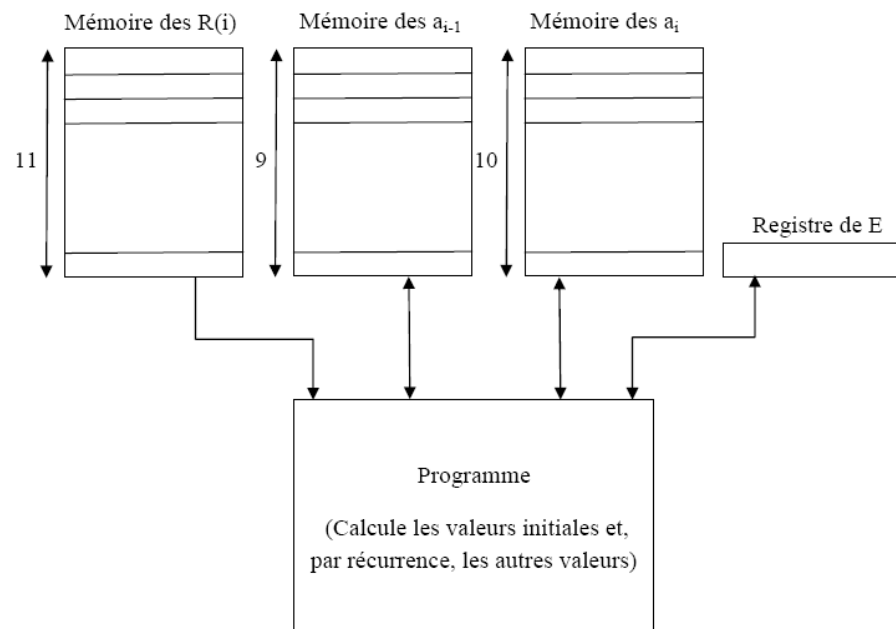
Remarque :

En pratique, il est difficile de réaliser la machine d'état de Levinson-Durbin précédente à cause de sa complexité puisqu'il y a en tout plus d'une centaine d'états ce qui est énorme. En plus, elle n'est pas très fiable.

Pour cela, on a pensé à une méthode plus simple. La méthode de la programmation en l'occurrence.

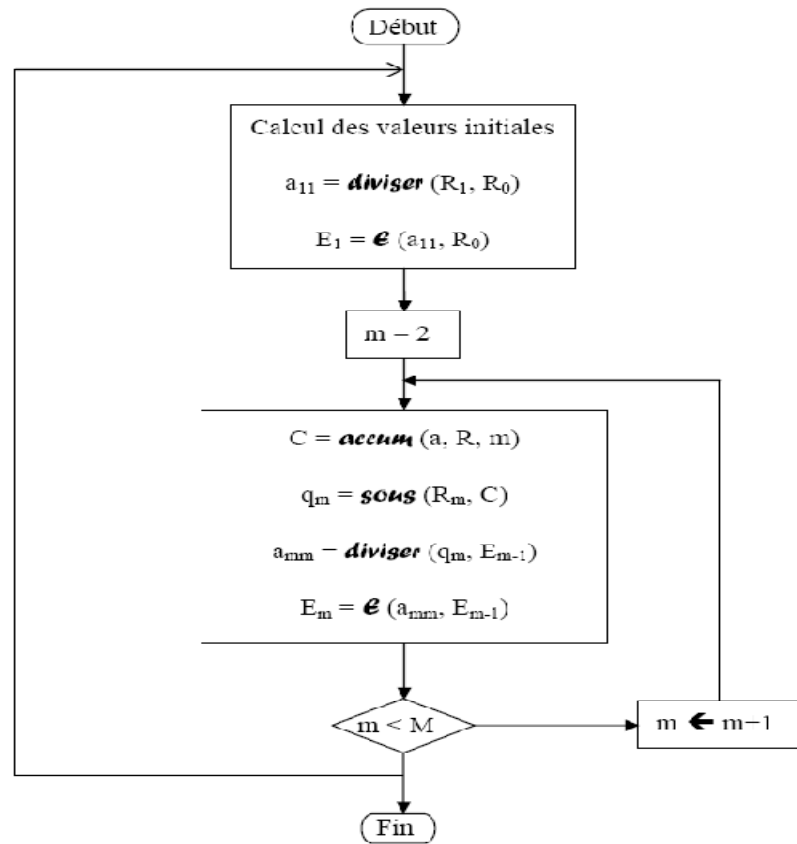
6.3.2.2. Le bloc de Levinson-Durbin par la programmation directe

Cette méthode consiste à construire des mémoires où on stocke les $R(i)$, a_i et a_{i-1} et de faire des fonctions (fonctions) qui feront les différentes opérations (multiplication, division ...).



L'organigramme suivant explique d'une façon plus explicite le fonctionnement du schéma précédent.

Diviser, e, accum, sous sont des fonctions à définir ultérieurement.



Diviser : on suppose qu'on va calculer $y = a/b$ où a , b et y ont le même nombre de bits $(n+1)$. Pour, par exemple, $a = "1011"$ (décimal 11) et $b = "0011"$ (décimal 3), duquel on doit avoir $y = "0011"$ (décimal 3) et le reste $a = "0010"$ (décimal 2). On va d'abord créer une version déplaçable de b de longueur $(2n+1)$. $b_inp(i)$ est tout simplement déplacé à droite de i positions comme le montre le tableau suivant. Le calcul se termine après $(n+1)$ itérations.

Indice (i)	a-déplaçable (a_inp)	comparaison	b-déplaçable (b_inp)	y (quotient)	Opération sur la 1 ^{ère} colonne
3	1011	<	0011000	0	Non
2	1011	<	0001100	0	Non
1	1011	>	0000110	1	$a_inp(i)-b_inp(i)$
0	0101	>	0000011	1	$a_inp(i)-b_inp(i)$
	0010 (reste)				

$e(a, b) = a(1-b^2)$: on suppose a et b deux nombres binaires de nombre de bits $2n$ (n bits pour la partie entière et n bits pour la partie fractionnaire). Quand le b est élevé au carré,

le nombre de bits de b^2 sera $4n$ ($2n$ bits pour la partie entière et $2n$ bits pour la partie fractionnaire). En multipliant a par b^2 le nombre de bits devient $6n$ ($3n$ bits pour la partie entière et $3n$ bits pour la partie fractionnaire). En négligeant les $2n$ bits les plus à droite de la partie fractionnaire et en sachant que la partie significative de la partie entière ne dépasse jamais les n bits ; on aura alors le produit ab^2 sur $2n$ bits (n bits pour la partie entière et n bits pour la partie fractionnaire). On calcule, enfin, $a - ab^2$ qui est égal à $a(1 - b^2)$. Le résultat est, évidemment, sur $2n$ bits.

Accum (a, b, m) : il fait les produits $a(i, m-1) * b(m-i)$ avec $i = 1, \dots, m-i$; et il accumule les résultats. Dans notre cas $m=10$ et les nombres a et b sont sur 21 bits (10 bits pour la partie entière et 11 bits pour la partie fractionnaire) chacun. Après l'accumulation des produits on pourra dépasser les 42 bits ; mais, on ne retiendra que 21 bits en négligeant les bits les plus à droite de la partie fractionnaire (on laisse seulement 11 bits) et en supprimant les bits les plus à gauche de la partie entière (on laisse seulement 10 bits) sachant bien sûr que ces bits ne sont pas significatifs (bits nuls).

Sous (a, b) = $a - b = a + C_2(b)$, avec a et b deux nombres binaires et $C_2(b)$ le complément à 2 du nombre b .

CONCLUSION GENERALE

Dans notre travail, nous avons étudié le codage par la prédiction linéaire. Ensuite, nous avons implémenté son algorithme sur un FPGA de la société Xilinx développé sur Virtex-II. On a conclu que l'utilisation des circuits ASIC's dans le domaine du codage de la parole offre de grandes opportunités pour les concepteurs, de même elle offre une autre dimension pour l'espace de jeu sur les paramètres de l'optimisation.

Le développement avec des langages de description VHDL l'un des principaux avantages de l'utilisation des FPGA, ce dernier nous permet de concevoir le circuit selon nos besoins

Le parallélisme de calcul dans les circuits configurables de type FPGA nous a permis d'obtenir de très bonnes performances en termes de temps de calcul.

Parmi les difficultés de l'utilisation de FPGA la représentation des nombres sur une taille limitée et prédéfinie ce qui rend le calcul moins précis. Pour cela nous proposons d'étudier la représentation des nombres avec virgule flottante sur FPGA pour plus de précision.

BIBLIOGRAPHIE

- [1] B. Atal. "A model of LPC excitation in terms of eigenvectors of the autocorrelation of the impulse response of the LPC filter". ICASSP, Vol. 1. Page(s) 45-48, 1989.
- [2] X. Huang, A. Acero, H. Hon, "Spoken language processing a guide to theory, algorithm and system design", Prentice Hall 2001.
- [3] S.Furui " Digital Speech processing, Synthesis, Recognition", Second edition, Marcel Dekker 2001.
- [4] W.C. Chu, Speech Coding Algorithms, Foundation and Evolution of Standardized Coders, John Wiley & Sons, Hoboken, 2003.
- [5] J. Makhoul, "Linear prediction: A Tutorial Review", Proceedings of the IEEE, vol. 634 pp 561-578.
- [6] K.K. Paliwal ,W.B. Kleijn, "Quantization of LPC parameters" in Speech Coding and Synthesis, W.B. Kleijn and K.K. Paliwal, Ed. Amsterdam: Elsevier, 1995, pp.443-466.
- [7] Coding of speech at 8 kbit/s using conjugate-structure algebraic-code-excited linear prediction (CS-ACELP). International Telecommunication Union (ITU). Recommendation UIT-T G729, March 1996.
- [8] B.Chaoub, L.Chontam "Compensation des systèmes standards de compression /décompression d'images".81p. Mémoire de maîtrise ès sciences appliquées. QUBEC : Université de Sherbrooke, Septembre 1995.

- [9] R. M. Gray, "Vector quantization," IEEE ASSP Mag., vol. 1, pp. 4-29, Apr. 1984.
- [10] F. Merazka , D. Berkani, " Vector Quantization Of LPC Parameters By Split", pp IEEE.1998. 434-437.
- [11] Baudoin et al. "Codage de la parole à bas et très bas débit". Annales des Télécommunications, 2000 :p 1-19.
- [12] A.S. Spanias, "Speech Coding : A Tutorial Review", Proc.IEEE, 82 (1994),pp. 1541–1582.
- [13] J.C. Nash "Compact numerical methods for computers: linear algebra and function minimization ", Second Edition Adam Hilger 1990.
- [14] P Corney, JS Mason. "Singular value decomposition and its modelling of speech excitation" IEE Conference, Loughborough, 1991. Page: 305-308
- [15] D.Berkani."Application de la transformée orthogonale SVD en compression de la parole", Séminaire National sur l'Automatique et les signaux SNAS, Annaba 1999.

