UNIVERSITÉ DU QUÉBEC

MÉMOIRE PRÉSENTÉ À
L'UNIVERSITÉ DU QUÉBEC À TROIS-RIVIÈRES

COMME EXIGENCE PARTIELLE
DE LA MAÎTRISE EN GÉNIE ÉLECTRIQUE

PAR :
EL ABIDI FAHIM

ÉTUDE D'UN ALGORITHME BASÉ SUR LE FILTRAGE H∞ POUR LA RECONSTITUTION DE MESURANDES DYNAMIQUES ET PROPOSITION D'UNE ARCHITECTURE DÉDIÉE

SEPTEMBRE 2004

Dédicace

À mes parents,

À ma femme et ma fille,

À toute ma famille,

À mes professeurs.

Remerciements

Je voudrais ici exprimer ma gratitude à tous ceux qui ont contribué à la réalisation de ce travail.

- Mes remerciements vont à mon codirecteur de recherche Mohammed Ben Slima qui m'aura encadré et soutenu tout au long de ce projet. Sa contribution technique a indéniablement constitué un très grand apport dans les résultats de cette recherche.

- J'adresse également mes sincères remerciements à mon directeur de recherche Ahmed Cheriti qui nous aura apporté une très grande contribution technique.

- Je tiens à remercier également tous les professeurs de l'UQTR du département de Génie Électrique, et spécialement Messieurs Kodjo Agbossou, Adam Skorek, Pierre Sicard.

- Je suis également reconnaissant envers le personnel du département génie électrique qui m'a toujours apporter l'aide dont j'avais besoin au où il le fallait.

- Que tous ceux qui n'ont pas été cités ne sentent nullement oublier, ils auront tous contribué à leur manière à la réalisation de ce travail.

LISTE DES SYMBOLES

$\mathbf{L_2}$	: Espace Hilbert des fonctions réelles du temps de carré intégrable
$\mathbf{X}$	: Espace des nombres réelles pour le mesurande x
$\mathbf{V}$	: Espace des nombres réelles pour les grandeurs d'influence v
$\mathbf{D_{ref}}$	: Ensemble des données de référence
$\mathbf{M}$	: Opérateur du modèle direct de mesure
$\mathbf{R}$	: Opérateur de reconstitution de mesurande
$\mathbf{F}$	: Opérateur du transformée de Fourrier
$\mathbf{L}$	: Opérateur du transformée de Laplace
$*$	: Opérateur de convolution
SNR	: rapport signal-bruit
$\circ$	: valeur exacte
$\hat{\cdot}$	: valeur estimée
$\tilde{\cdot}$	: valeur bruitée
$\mathbf{x}$	: grandeur à reconstituer – mesurande
$\mathbf{y}$	: mesure brute
$\mathbf{g}$	: réponse impulsionnelle du système de mesure
$\mathbf{v}$	: vecteur des grandeurs d'influence
$\mathbf{s}$	: vecteur des grandeurs de stimulation
η	: bruit de mesure
ξ	: processus générateur
t	: grandeur scalaire modélisant le temps
λ	: grandeur scalaire modélisant la longueur d'onde
ω	: pulsation
$\mathbf{X}(j\omega)$	: $\mathbf{F}\{x(t)\}$ transformée de Fourier de $\mathbf{x}(t)$
$\mathbf{X}(s)$	: $\mathbf{L}\{x(t)\}$ transformée de Laplace de $\mathbf{x}(t)$

N	: nombre d'échantillons de mesure brute
Mg	:dimension du vecteur d'états
T	: intervalle de temps ou support du mesurande
Δ_Y	: incertitude sur les mesures brutes
Δ_X	: limite supérieure de l'erreur de reconstitution
Δx	: variation du grandeur x
Δ_x^{ref}	: limite de l'erreur absolue du vecteur x^{ref}
x_{max}; x_{min}	: valeur maximale et minimale de x
$\Delta\lambda$	: pas de discrétisation de l'axe λ
$x_n = x(\lambda_n)$	: échantillon de $x(\lambda)$ prélevé en $\lambda = n\Delta\lambda$
$\{x_n\}$	: séquence d'échantillons de $x(\lambda)$
σ_η^2	: variance du bruit de mesure
σ_ξ^2	: variance du processus générateur
ε	: erreur quadratique moyenne relative (EQMR)
$\mathbf{A}$	: matrice d'état continu
$\mathbf{b}$	: vecteur d'entrée du système
$\mathbf{C}$	: vecteur de sortie du système
$\mathbf{z}$	: vecteur d'état discret
$\mathbf{0}_{n;m}$	: matrice zéro de dimension $n \times m$
$\mathbf{I}_n$	: matrice identité d'ordre n
$[\,]^T$	: vecteur ou matrice transposé
$E[\,]$	: espérance mathématique
$J(\,)$	: critère d'optimisation
$\delta(\,)$	: impulsion de Dirac
dim(:)	: dimension d'une matrice ou d'un vecteur
trace{ }	: trace d'une matrice
arg	: argument
sup	: supremum
$\|\mathbf{X}\|$	: norme euclidienne du vecteur $\mathbf{X}$

$\|\mathbf{G}\|_\infty$	: norme H∞ de la matrice **G**, maximum de la valeur singulière de **G**
β_{opt}	: valeur optimale de β
K	: vecteur gain (Kalam et H∞)
DSP	: processeur numérique à usage générale
ITGE	:intégration à très grand échelle (*en anglais* : VLSI)
ASIC	:*en anglais* : Application Specific Integrated circuit
Si	: Silicium
W_{obs}	: matrice d'observabilité d'un système
UAL	: unité arithmétique et logique
UGA	: unité d'aiguillage d'adresse
Mux	: multiplexeur
LIFO	: dernier entrant premier sortant
RAZ	: remise à zéro
PE	: processeur élémentaire
VHDL	:*en anglais* : Very High Speed Integrated Circuit Hardware Description Language
E/S	: entrée/ sortie
RISC	jeu d'instructions réduit (*en anglais* : Reduced Instruction Set Computers)

TABLE DES MATIÈRES

LISTE DES FIGURES

LISTE DES TABLEAUX

INTRODUCTION

La reconstitution de mesurandes représente un problème fondamental dans la métrologie. Elle consiste à estimer un signal x(t) qui ne peut pas être mesurer directement, à partir des résultats de mesure d'un autre signal $\tilde{y}(t)$ qui est lié, dans la plupart des cas, avec le premier de façon causale. Le traitement numérique des signaux joue un rôle de plus en plus important dans la reconstitution de mesurandes qui peut être une opération relativement complexe. Elle est utilisée pour l'interprétation de données séismiques, l'amélioration de la résolution de l'analyse spectrométrique et chromatograph, la correction dynamique de capteurs, la mesure de la thermocinétique de réactions chimiques, l'accélération des mesures quasi-statiques, l'égalisation de canaux de télécommunications, le diagnostique médical basé sur les techniques d'échographie ultra-sonore et dans le domaine de la robotique comme moyen de l'étalonnage cinétique et de l'estimation de paramètres dynamiques de manipulateurs, etc.

Les résultats de mesure obtenus sont toujours bruités, car lorsqu'une mesure est obtenue au travers d'une chaîne de conversion de mesure, les perturbations indésirables, telles que la température, les bruits de fond, les bruits d'amplification, les vibrations etc, entraînent des imperfections au niveau du résultat obtenu. La présence inévitable de ces erreurs de conversion affecte non seulement l'acquisition des résultats de mesure d'observations, mais aussi ne facilite pas leur interprétation et ne permet pas toujours d'avoir la valeur exacte recherchée. À cause de ces limitations et des imperfections des

éléments du système de mesure, le mesurande subit, au cours de son traitement, des dégradations. Dans ce cas, pour extraire la valeur originale, nous sommes obligés de se baser sur des observations, et des informations que nous connaissons *à priori* sur le système de conversion.

Nous savons qu'en absence de ces bruits, il existe diverses solutions formelles utilisant des transformations linéaires intégrales, mais en pratique, on ne peut envisager de solutions à ces problèmes qu'en ajoutant des contraintes physiques externes. Ce sont des problèmes mal posés, et leur résolution n'est pas triviale. Il faut donc traiter ces signaux à l'aide d'algorithmes de reconstitution de mesurandes dynamiques.

Problématique

De nombreuses études et domaines de recherche traitent le problème de reconstitution des signaux dans les systèmes de mesure [1][6][7][8][9]. La plupart des ces études ont été basées sur la minimisation de la variance de l'erreur d'estimation du signal estimé, i.e. l'approche de filtrage célèbre tel que Kalman. Ce type d'estimation suppose que le processus de mesure a une dynamique connue, et que les sources de bruit ont des propriétés statistiques connues.

Cependant, ces suppositions peuvent limiter l'application des estimateurs puisque dans beaucoup de situations seulement un modèle approximatif est disponible et / ou les statistiques des sources de bruit sont entièrement non connues ou indisponibles. En outre,

les estimateurs peuvent ne pas être robustes contre l'incertitude du modèle de système de mesure

Notre but est de trouver un filtre dont l'apport dans le domaine de traitement de signaux et spécialement en spectrométrie se résume en deux points : la connaissance statistiques des perturbations n'a aucun effet sur sa conception, en plus, ce filtre doit être robuste en horizon infini contrairement à d'autres filtres tels que Wiener et Kalman. D'autre part, les algorithmes itératifs tel que Van Cittert et Jasson nécessitent beaucoup de calcul pour obtenir une bonne précision contrairement à l'algorithme cherché qui doit nécessité moins d'opérations arithmétiques. Récemment, une nouvelle classe de filtre optimal a été développée en utilisant le critère d'estimation H∞ minimum du spectre d'erreur [3] [4] [5], ces filtre peuvent résoudre les problèmes en question en les modifiant suivant l'application, ces filtre doivent nous garantir une bonne estimation.

Les techniques de traitement numérique ont d'abord été utilisées sous forme d'algorithmes, généralement exprimés en langages évolués, puis dans des cartes de processeurs numériques de signaux ou DSP (Digital Signal Processor), pour répondre à des exigences de vitesse. Dans ce cheminement, l'étape suivante est la mise au point de circuits intégrés dédiés à des algorithmes précis. Cette étape, une fois réalisée, donne au système de mesure de nombreux avantages:

- rapidité d'exécution (temps réel);
- précision adaptée;
- taille et transportabilité (encombrement moindre).

Les progrès dans l'intégration sur Silicium de systèmes de mesures est rendu possible, grâce aux progrès récents dans les domaines connexes tel que la microélectronique et la micro usinage. Donc, une amélioration de la qualité des mesures peut être obtenue grâce à la conception des ASIC en utilisation des algorithmes de traitement des signaux (pour la reconstitution) et des capteurs micromécaniques (pour la conversion), les deux implantés en technologie ITGE et micro usinage. Comme ce filtre sera implanté en technologie ITGE, le mesurande (quelque soit sa nature) sera convertit en signal électrique analogique qui est à son tour convertit à travers un convertisseur analogique / numérique

Les exigences concernant la miniaturisation du système, sa fiabilité, sa consommation d'énergie et la facilité du design et/ou de la maintenance, sont les incitations les plus évidentes à l'intégration. De plus, si la reconstitution exige l'utilisation de modèles mathématiques nécessitant une capacité de calculs considérable s'ajoute l'intérêt envers un processeur spécialisé pour la reconstitution de mesurandes. Dépendamment des exigences concernant la précision, la vitesse de traitement et la fiabilité des mesures, on peut opter pour un processeur de traitement de signaux d'application générale (microcontrôleurs, DSP, etc.) ou bien, on peut avoir recours aux structures spécialisées, réalisées en technologie ITGE (FPGA, ASIC, etc.).

Généralement, l'application d'un processeur spécialisé à la place d'un DSP d'application générale est principalement justifiée par les exigences de vitesse. Utilisant le parallélisme inhérent de l'algorithme de traitement, le processeur spécialisé assure une vitesse beaucoup plus élevée.

Ce mémoire porte sur la mise en commun de trois champs d'études:

- le traitement des données numériques;
- utilisation des processeurs numériques de traitement de signaux à usage générale;
- la microélectronique (Very Large Scale Integration: VLSI/ITGE).

OBJECTIFS

Ce projet a pour objet l'application du filtrage H∞ à la reconstitution des signaux de mesures. Ils se résument ainsi:

l'étude théorique du filtrage H∞, puis, la comparaison de ce filtre avec d'autre méthode de reconstitution de mesurande notamment l'algorithme de Gold, Jasson et le filtre de Kalman. De plus, l'amélioration de ce filtre afin d'obtenir des résultats plus performants. Enfin, l'élaboration d'un programme de reconstitution des signaux pour le processeur numérique de traitement de signaux à usage générale 563xx de Motorola et la proposition et la conception de l'architecture d'un processeur spécialisé pour un algorithme de reconstitution des signaux basé sur le filtrage H∞.

MÉTHODOLOGIE

Les éléments principaux de la méthodologie que nous adopterons seront les suivants:

- une recherche bibliographique sur la norme H∞ et sur le filtrage H∞ en particulier.
- une élaboration d'un algorithme de filtrage H∞, des simulations de l'algorithme en se basant sur des données synthétiques utilisées dans la littérature et une comparaison de l'algorithme avec les algorithmes de référence, puis des validations de l'algorithme en

utilisant des données réelles spectrométriques ainsi que des données réelles de télécommunication et enfin des améliorations de l'algorithme;

- le choix du DSP convenable pour la reconstitution de signaux basé sur notre filtre, l'effet de quantification et rééchelonnement des données et la conception d'un algorithme optimisé pour le DSP;
- une recherche bibliographique sur les architectures VLSI pour le filtrage H∞, puis une étude de la première version du processeur spécialisé pour la reconstitution de signaux basé sur le filtrage H∞, ensuite une étude de l'algorithme à implanter. La recherche du parallélisme dans l'algorithme sera l'étape qui suit, puis le choix d'une architecture pour l'algorithme de reconstitution, enfin la conception d'une première version du processeur spécialisé et la rédaction d'un mémoire.

L'objet de notre recherche est le filtrage H∞ et son application à la déconvolution dans les système de mesure et la conception d'une architecture dédiée.

Ainsi, le 1er chapitre étudie la problématique du traitement du signal dans l'analyse du problème de reconstitution de mesurande dynamique. Le chapitre 2 expose les méthodes et algorithmes de reconstitution de mesurande dynamique de référence. Le chapitre 3 présente l'ensemble de l'analyse visant à résoudre le problème de reconstitution de mesurande dynamique par filtrage H∞; ce chapitre traite les points suivants :

- le choix du filtre H∞;
- le choix du modèle d'état ;
- des simulations de l'algorithme en se basant sur des données synthétiques;
- une comparaison des résultats obtenus avec ceux des autres méthodes de référence;

- des validations de l'algorithmes par des données réelles spectrométriques et de télécommunication;
- une proposition d'autres méthodes conjointes avec le filtrage H∞ pour augmenter le perfectionnent de reconstitution;

Le chapitre 4 étudie la reconstitution des signaux figée dans le DSP de 563xx de Motorola basée sur le filtrage H∞, et l'implantation de l'algorithme sur silicium. Ainsi, il traite les points suivants :

- le choix de l'algorithme;
- une discussion sur les choix préalables à faire, en ce qui concerne la numérisation des données et l'effet de quantification;
- l'étude de ce qui a déjà été réalisé dans ce domaine;
- une discussion architecturale pour dégager les traits généraux d'une architecture dédiée au filtrage H∞;
- la proposition d'une nouvelle architecture.

Finalement, le chapitre 5 présente le résumé de l'ensemble des résultats obtenus.

Chapitre 1

DÉFINITION ET ANALYSE DU PROBLÈME DE RECONSTITUTION DE MESURANDES DYNAMIQUES

1.1 Modèle de mesure

La modélisation des systèmes de mesure a pour but principal d'établir le lien entre la grandeur réelle et celle mesurée pour que la représentation des résultats fournis par l'instrument traduise la réalité. Cette modélisation est relativement complexe car elle contient des éléments matériels et logiciels, électriques et non électriques, analogiques et numériques.

Le modèle général d'un système de mesure est présenté à la figure 1.1. [17]. Dans ce modèle: $\mathbf{x}(t)$ est le mesurande, $\{\mathbf{y_n}\}$ est le vecteur des résultats bruts de mesure, $\mathbf{v}$ représente le vecteur des grandeurs d'influence et $\{\mathbf{u_n}\}$ est le signal stimulant l'objet de mesure. Dans tout ce qui suit le symbole avec "$\overset{\circ}{\cdot}$" indique une valeur exacte, le symbole avec "$\hat{\cdot}$" indique une valeur estimée et le symbole avec "$\tilde{\cdot}$" désigne une valeur bruitée. En particulier

- $\overset{\circ}{\mathbf{x}}(t)$, $\overset{\circ}{\mathbf{v}}$ et $\{\overset{\circ}{\mathbf{u}}_n\}$ sont les valeurs exactes de $\mathbf{x}(t)$, v et $\{\mathbf{u}_n\}$ respectivement;
- $\hat{\mathbf{x}}(t)$ et $\hat{\mathbf{v}}$ sont les valeurs estimées de $\mathbf{x}(t)$ et v respectivement;
- $\{\tilde{\mathbf{y}}_n\}$ et $\{\tilde{\mathbf{u}}_n\}$ sont les séquences des valeurs bruitées des vecteurs $\mathbf{y}_n$ et $\mathbf{u}_n$ respectivement.

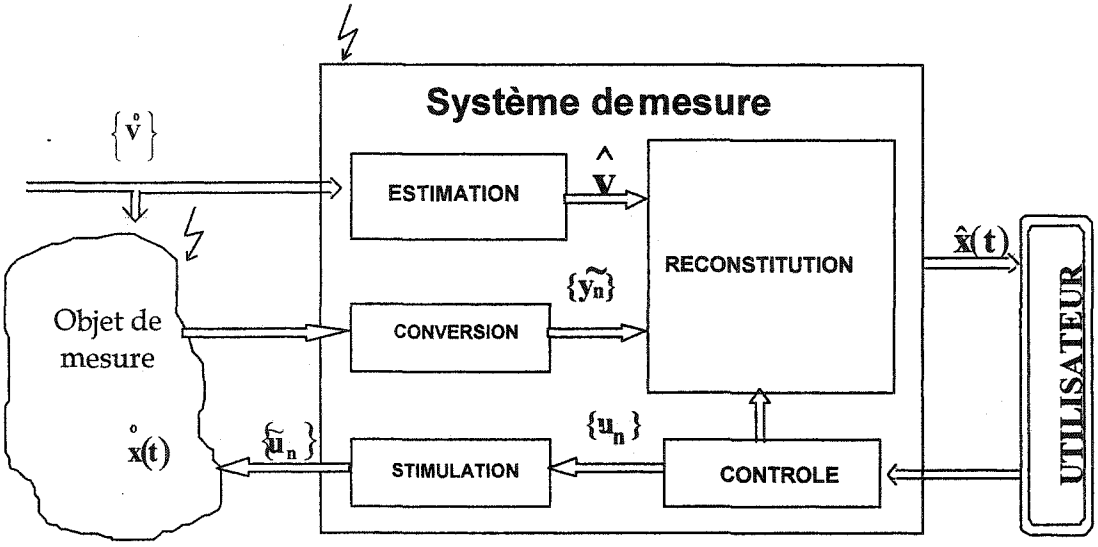


Figure 1.1: Modèle général d'un système de mesure

La conversion dans le système de mesure électrique consiste en une série de transformations de signaux $\overset{\circ}{\mathbf{x}}(t)$ reçus de l'objet de mesure (stimulé par $\{\tilde{\mathbf{u}}_n\}$) en des signaux électriques, standards, préférablement numériques qui représentent alors les résultats bruts de mesure $\{\tilde{\mathbf{y}}_n\}$. La reconstitution consiste donc à traiter ces résultats bruts de mesure, en utilisant les valeurs estimées $\hat{\mathbf{v}}$ de $\overset{\circ}{\mathbf{v}}$ et les valeurs exactes de $\{\overset{\circ}{\mathbf{u}}_n\}$, afin

d'obtenir des estimés $\hat{\mathbf{x}}(\mathbf{t})$ du mesurande; résultat final de mesure. En d'autres termes, la reconstitution d'un mesurande consiste en l'estimation d'un signal $\overset{\circ}{\mathbf{x}}(\mathbf{t})$, qui n'est pas mesurable directement, à partir des résultats partiels de mesure $\{\tilde{\mathbf{y}}_n\}$ liés avec $\overset{\circ}{\mathbf{x}}(\mathbf{t})$ de façon causale. Pour effectuer la reconstitution, il est nécessaire de connaître les relations entre ses grandeurs, alors d'identifier le modèle mathématique du système de mesure, ou plus précisément du bloc de conversion :

$$\left\{\overset{\circ}{\mathbf{y}}_n\right\} = \mathbf{M}\left[\overset{\circ}{\mathbf{x}}(\mathbf{t}); \overset{\circ}{\mathbf{v}}; \left\{\overset{\circ}{\mathbf{u}}_n\right\}\right] \tag{1.1}$$

Où $\{\overset{\circ}{\mathbf{y}}_n\}$ sont les valeurs exactes des résultats de mesure en l'absence des grandeurs parasites, **M** est un opérateur dont son inversion ou pseudo inversion est la clé à la reconstitution de mesurande. Ce modèle est généralement décrit par un système d'équations algébriques, différentielles ou intégrales.

Le bloc de la reconstitution est caractérisé par un opérateur de reconstitution de mesurande :

$$\hat{\mathbf{x}}(\mathbf{t}) = \mathbf{R}\left[\{\tilde{\mathbf{y}}_n\}; \hat{\mathbf{v}}; \left\{\overset{\circ}{\mathbf{u}}_n\right\}\right] \tag{1.2}$$

Généralement choisi comme l'inverse ou pseudo inverse du modèle de conversion (Eq. (1.1)). Cet opérateur peut être aussi déterminé directement lors de l'étalonnage du système de mesure.

1.2 Étalonnage d'un système de mesure et reconstitution d'un mesurande

L'étalonnage d'un système de mesure comprend l'ensemble des opérations qui permettent d'expliciter l'opérateur **M** défini par la relation (1.1) ou l'opérateur **R** défini par la relation (1.2). L'opération d'étalonnage peut être utilisée pour :

- appréhender le fonctionnement du système;
- élaborer une stratégie de commande ou de contrôle;
- valider des données;
- diagnostiquer l'état de fonctionnement d'un système;
- prédire ou maîtriser le comportement d'un système;
- minimiser ou prendre en compte les effets de perturbations.

Selon la nature du système de mesure (statique ou dynamique) nous distinguons deux catégories d'étalonnage / reconstitution:

- l'étalonnage / reconstitution statique où une valeur de la mesure brute $\mathbf{y}_n$ dépend uniquement d'une valeur du mesurande $\mathbf{x}(t_n)$;
- l'étalonnage / reconstitution dynamique où la mesure brute $\mathbf{y}_n$ dépend d'un ensemble de valeurs du mesurande

$$\left\{\mathbf{x}(t) \middle| t \in \left[t_n - T_1 \quad , \quad t_n + T_2\right]\right\}$$

Pour les deux cas, le but de l'étalonnage est d'estimer les paramètres ou les fonctions qui décrivent l'opérateur **M** ou **R** en se basant sur les données de référence $\mathbf{D}^{ref}$:

$$\mathbf{D}^{ref} = \left\{ \tilde{\mathbf{x}}_n^{ref}, \tilde{\mathbf{v}}_n^{ref}, \tilde{\mathbf{y}}_n^{ref} \middle| n = 1,2,....,N \right\} \tag{1.3}$$

sujet à des erreurs de mesure:

$$\left| \Delta\left[\tilde{\mathbf{x}}_n^{ref} \right] \right| \leq \bar{\Delta}\mathbf{x}_n^{ref}; \quad \left| \Delta\left[\tilde{\mathbf{v}}_n^{ref} \right] \right| \leq \bar{\Delta}\mathbf{v}_n^{ref}; \quad \left| \Delta\left[\tilde{\mathbf{y}}_n^{ref} \right] \right| \leq \bar{\Delta}\mathbf{y}_n^{ref} \tag{1.4}$$

La qualité de l'étalonnage dépend essentiellement de la qualité des données utilisées pour l'étalonnage ainsi que de l'approximation ou paramétrisation de l'opérateur **M** ou **R**.

1.3 Classification des problèmes de reconstitution de mesurande

Les problèmes de reconstitution de mesurande sont des cas spéciaux des problèmes de modélisation inverse. La résolution de ces problèmes n'est absolument pas triviale. La raison en est qu'il s'agit des problèmes mal posés dans la plupart des cas. Les différents critères qui sont utilisés pour la classification des problèmes de reconstitution de mesurande sont:

- le modèle mathématique du mesurande;
- le modèle mathématique de l'opérateur **M** ou **R** ;
- la nature physique du mesurande;
- les domaines d'applications.

Les deux premiers critères sont d'une grande importance pour l'utilisateur lorsqu'il vient d'élaborer une méthode de reconstitution. En utilisant le premier critère, nous pouvons classifier encore les problèmes de reconstitution selon la structure des variables **x** et **t** du mesurande à savoir :

- **x** est un scalaire ou un vecteur;
- **x** est aléatoire ou déterministe;
- **t** est vide, scalaire ou vectoriel;
- **t** contient une variable modélisant le temps ou autres quantités physiques;

Pour le deuxième critère, le modèle mathématique de l'opérateur **M** ou **R** est généralement représenté par un système d'équations logiques, algébriques, différentielles ou intégrales. Comme l'opérateur **M** modélise généralement une relation empirique causale entre le mesurande et la mesure brute, on peut ne pas avoir une solution unique au problème inverse. Cette solution peut être instable vis-à-vis de faibles variations des résultats crus de mesure. Ainsi, les problèmes de reconstitution de mesurandes peuvent être classifiés selon le degré de cette sensibilité aux variations des résultats crus de mesure.

1.4 Caractérisation générale du problème de reconstitution de mesurandes

Nous pouvons distinguer deux classes principales des opérateurs **M** suivant la complexité du problème de reconstitution de mesurande :

- dynamiques, linéaires et non linéaires;
- statiques, linéaires et non linéaires.

Pour le cas des opérateurs statiques linéaires, la résolution du problème de reconstitution de mesurandes est très simple, car la dépendance entre le mesurande et le résultat de mesure est linéaire.

Concernant les opérateurs dynamiques non linéaires, ils forment une vaste classe sans mode de présentation universelle et aucune théorie globale n'existe permettant, comme dans le cas linéaire, de déterminer simplement les relations liant la mesure brute au mesurande. Ils sont généralement représentés en série de fonctionnelles de Volterra, et dans ce cas la résolution du problème de reconstitution du mesurande s'avère une tâche très difficile. Dans de nombreux cas on essaye de linéariser le système ou parfois utiliser la partie linéaire du modèle et ainsi résoudre le problème de reconstitution.

Le problème de reconstitution dans le cas statique non linéaire, avec une grandeur scalaire d'influence, repose sur l'utilisation de l'opérateur **M** de la forme:

$$\tilde{\mathbf{y}} = \mathbf{G}(\mathbf{x}, \mathbf{v}, \mathbf{p}) \quad \text{avec} \quad \mathbf{x} \in \mathbf{X} \ , \ \mathbf{v} \in \mathbf{V} \tag{1.5}$$

Où **X**, **V** sont des sous-espaces des nombres réels, **G** est une fonction connue, réelle et monotone pour $(\mathbf{x}, \mathbf{v}) \in \mathbf{X} \times \mathbf{V}$, **p** est le vecteur des paramètres qui décrivent l'opérateur **G** . Ce dernier peut être estimé lors de l'étalonnage du système de mesure en se basant sur les données de référence D^{ref} . Le résultat de l'étalonnage peut être ensuite utilisé pour reconstituer le mesurande de la façon suivante :

$$\hat{\mathbf{x}} = \arg_{\mathbf{x}}\{\tilde{\mathbf{y}} = \mathbf{G}(\mathbf{x}, \hat{\mathbf{v}}, \hat{\mathbf{p}})\} \tag{1.6}$$

La méthodologie de la résolution du problème de reconstitution de mesurandes statiques repose sur trois éléments de base:

- la méthode de paramétrisation du modèle direct **M** ;
- la méthode d'estimation des paramètres **p**;
- la méthode de calcul de $\hat{\mathbf{x}}$ selon l'équation (1.6).

Le problème de reconstitution dans le cas dynamique linéaire, sans grandeurs d'influence, repose sur l'utilisation de l'opérateur M de la forme:

$$\{\tilde{\mathbf{y}}_n\} = \left\{\int_T \mathbf{g}(t_n, t, \mathbf{p}).\mathbf{x}(t)dt \middle| t_n \in \mathbf{T}\right\} \text{ pour } \mathbf{x}(t) \in \mathbf{X(T)} \tag{1.7}$$

où **T** est un sous-espace des nombres réels, t_n et $t \in \mathbf{T}$ sont des variables réelles modélisant souvent le temps, **X(T)** est un espace des fonctions réelles de t et $g(t_n, t; \mathbf{p})$ est une fonction connue. Le vecteur des paramètres **p** est estimé lors de l'étalonnage du système de mesure et le résultat est utilisé pour la reconstitution :

$$\hat{\mathbf{x}} = \mathbf{R}[\{\tilde{\mathbf{y}}_n\}, \hat{\mathbf{p}}] \tag{1.8}$$

Contrairement au cas statique non linéaire, le problème de reconstitution de mesurande pour le cas dynamique est souvent mal posé [58]. Les difficultés dans la résolution du problème proviennent du caractère régularisant de l'opérateur dynamique **M**

qui rend son inversion s'accompagne d'une amplification des effets des petites fluctuations irrégulières des résultats bruts de mesure $\{\tilde{\mathbf{y}}_n\}$. Cette instabilité fait que la solution obtenue s'avère être non unique et sujet à des erreurs très larges, alors elle est difficile à interpréter. En ce sens, la méthodologie de résolution de ce problème repose sur les prémisses suivantes [17]:

1 le résultat final de mesure $\hat{\mathbf{x}}(t)$ généré par **R** doit appartenir à l'espace **X(T)**;

2 son image $\mathbf{M}[\hat{\mathbf{x}}(t); \mathbf{p}]$ doit être proche des résultats de mesure $\{\tilde{\mathbf{y}}_n\}$ mais pas nécessairement identique;

3 optionnellement, $\hat{\mathbf{x}}(t)$ doit satisfaire certains critères additionnels qui reflètent les informations *a priori* disponibles sur la classe des mesurandes.

Les deux dernières prémisses (2) et (3) consistent à apporter une information supplémentaire sur la solution désirée afin de résoudre correctement le problème mal posé. En d'autres termes, elles introduisent des contraintes capables de restreindre l'ensemble des solutions acceptables et d'obtenir une solution unique et stable vis-à-vis des erreurs sur la mesure brute.

En résumé, la méthodologie de la reconstitution de mesurande dynamique dépend essentiellement de quatre éléments de base:

- la méthode de paramétrisation de l'opérateur **M** ou **R** ;
- la méthode d'estimation des paramètres **p**;
- la méthode de régularisation incorporée dans l'opérateur **R** ;
- la méthode de calcul de $\hat{\mathbf{x}}(t)$ selon l'équation (1.8).

Vu son comportement comme un problème inverse mal posé et les difficultés qui résultent de la résolution de ce genre de problème, nous nous intéresserons ici uniquement aux problèmes de la reconstitution de mesurandes dynamiques qui peuvent être résolus en utilisant des opérateurs **M** linéaires et invariants par translation.

1.5 Modélisation mathématique des données de mesure

La première étape de l'analyse du problème de reconstitution sera la modélisation mathématique des données de mesure $\{\tilde{y}_n\}$ qui sont généralement discrètes, en nombre fini et entachées d'erreurs de mesure ou bruits. Nous pouvons les représenter sous la forme d'un vecteur [1]:

$$\tilde{\mathbf{y}} = \left[\tilde{y}_0 \mid \tilde{y}_1 \mid \cdots \mid \tilde{y}_{N-1} \right]^T = \overset{o}{\mathbf{y}} + \eta \tag{1.9}$$

Où

$$\overset{o}{\mathbf{y}} = \left[\overset{o}{y}_0 \mid \overset{o}{y}_1 \mid \cdots \mid \overset{o}{y}_{N-1} \right]^T$$

et η est le vecteur qui représente les erreurs (ou bruits) additives:

$$\eta = \left[\eta_0 \mid \eta_1 \mid \cdots \mid \eta_{N-1} \right]^T$$

Nous savons qu'un instrument de mesure est toujours caractérisé par sa réponse impulsionnelle g(t), qui dans le cas où le système de mesure est linéaire et invariant, affecte le mesurande par convolution :

$$\tilde{y}_n = \int_\Gamma g(t_n - \tau).x(\tau)d\tau + \eta_n \qquad n=0, \ldots, N-1 \tag{1.10}$$

g(t), la réponse impulsionnelle du système de mesure, est censée être connue ou déterminée durant la procédure d'étalonnage. Si ce dernier peut être décrit par une combinaison linéaire de fonctions $\psi(t)e^{-\alpha t}$, où $\psi(t)$ est un polynôme algébrique, nous pouvons alors décrire l'équation de convolution par une équation différentielle ordinaire. De plus, si nous nous restreindrons au cas où **x**(t) et **y**(t) sont des fonctions tempérées ayant une transformée de Fourier, nous pouvons représenter l'équation de convolution (1.10) dans le domaine des transformées de Fourier ou dans le domaine des transformées de Laplace.

1.6 Mal conditionnement numérique du problème de reconstitution

En fait, un problème est dit mal posé au sens de Hadamard [59], s'il ne satisfait pas au moins à une parmi les trois exigences suivantes:

- la solution existe;
- la solution soit définie de façon unique;
- la solution est stable.

Le problème de reconstitution défini à temps continu ne respecte pas les trois exigences de Hadamard; alors c'est un problème mal posé.

En effet, soient x(t); y(t) et g(t) possédant des transformées de Fourier, nous pouvons reformuler notre problème (1.10) dans le domaine fréquentiel:

$$\tilde{\mathbf{Y}}(j\omega) = \mathbf{X}(j\omega).\mathbf{G}(j\omega) + \mathbf{N}(j\omega) \tag{1.11}$$

Ainsi, en procédant par filtrage inverse (utilisant le filtre qui a comme entrée $\tilde{\mathbf{Y}}$ et comme sortie **X**) suite a l'équation (1.11), on obtient :

$$\mathbf{X}(j\omega) = \frac{\tilde{\mathbf{Y}}(j\omega)}{\mathbf{G}(j\omega)} - \frac{\mathbf{N}(j\omega)}{\mathbf{G}(j\omega)} \tag{1.12}$$

Il faut que 1 / **G**(jω) existe et soit une fonction tempérée. De plus, **G**(jω) doit être une fonction qui ne s'annule pour aucune valeur de fréquence ω et qui ne tende pas vers zéro à l'infini plus vite qu'une puissance de 1/ω, pour que nous puissions définir une solution unique. Dans la pratique ces conditions ne sont jamais satisfaites; on peut rencontrer dans beaucoup de cas que **G** (jω) est une fonction qui, théoriquement, s'annule au delà d'une fréquence de coupure ω_c et qu'elle tende vers zéro quand ω s'accroît. Alors, **G** (jω) présentera des passages par zéro et la division 1/ **G** (jω) est pratiquement impossible.

D'autre part, le rapport

$$\frac{\mathbf{N}(j\omega)}{\mathbf{G}(j\omega)}$$

peut ne pas admettre de transformée de Fourier inverse à cause de l'influence des hautes fréquences ω de la réalisation du processus aléatoire N(jω). Même si elle admet une transformation inverse, l'écart de cette fonction du zéro peut être aussi grande que l'on veut, et nous tombons dans la sur sensibilité vis-à-vis de faibles variations des données de mesure.

1.7 Régularisation du problème de reconstitution

La régularisation consiste à remplacer un problème mal posé par un problème bien posé dont les solutions sont des approximations du problème mal posé. La régularisation signifie les points suivants [23] [24] [25] [57]:

- un changement de la notion même de la solution en introduisant les solutions approchées, les quasis solutions, etc. ;
- un changement d'espaces et de topologie dans la formulation du problème (le même problème mal posé peut s'avérer bien posé dans d'autres espaces métriques);
- une imposition directe des contraintes sur l'ensemble des solutions;
- une introduction d'opérateurs régularisants dans le critère d'optimisation de la solution;
- une extension stochastique d'un problème initialement déterministe.

Chaque méthode de régularisation peut être interprétée comme une méthode de construction de l'ensemble des solutions admissibles.

Chapitre 2

MÉTHODES ET ALGORITHMES DE RECONSTITUTION DE MESURANDES DYNAMIQUES

2.1 Classification et caractérisation générale

Dans ce paragraphe, on trouve une revue des différentes méthodes de reconstitution les plus répandues proposées dans la littérature, en utilisant le mécanisme de régularisation comme critère de leur classification.

- **Méthodes directes** : elles consistent en la solution directe des équations algébriques résultant de la discrétisation du modèle de données. La méthode de discrétisation du modèle de données dépend du type du modèle : équation intégrale ou équation différentielle. L'ensemble des solutions admissibles est contraint à ceux des signaux continus qui peuvent être représentés exactement par les échantillons discrets obtenus comme résultats de reconstitution [8] [21] [23] [57].

Notons ici l'importance du choix du pas d'échantillonnage. En effet, il joue le rôle de paramètre de régularisation; sa valeur peut être choisie de façon à minimiser l'erreur totale de reconstitution. Le choix de ce paramètre implique un compromis entre la

stabilité numérique et l'exactitude de la solution. En effet, lorsqu'on diminue le pas d'échantillonnage on diminue l'erreur de discrétisation, mais dans ce cas les valeurs propres les plus faibles de la matrice G tendent vers zéro entraînant par la suite la dégradation du conditionnement de la matrice. Nous comprenons donc, que plus l'erreur de discrétisation est faible, plus la solution directe est sujette à caution. [1]

- **Méthodes variationnelles :** elles consistent à contraindre l'ensemble des solutions admissibles à celles qui minimisent (ou maximisent) un critère, noté **J**[x], définissant la qualité de la reconstitution:

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} \{\mathbf{J}[\mathbf{x}]\} \tag{2.1}$$

L'énergie de l'erreur résiduelle $\tilde{\mathbf{y}}$ - **M**[x] est utilisée plus fréquemment comme critère **J**[x], et la minimisation se fait généralement dans des espaces métriques.

Nous utiliserons souvent ce critère d'erreur combiné avec d'autres contraintes supplémentaires afin d'obtenir une solution optimale unique et stable au problème. Parmi les contraintes supplémentaires, nous retrouvons la minimisation de l'énergie, la maximisation de la puissance et la maximisation de l'entropie du mesurande reconstitué. Nous retrouverons, dans cette classe, la méthode avec régularisation au sens de Tikhonov [18] [60] [61] (régularisation continue), la méthode avec régularisation au sens de Phillips et Twomey [62] [63] (régularisation discrète), la méthode des moindres carrés avec contraintes [64], la méthode de projection sur des ensembles convexes. [24]

Dans le cas ou les erreurs de données sont une réalisation d'un bruit blanc gaussien, la minimisation de ce critère d'erreur donne les mêmes résultats que la méthode à maximum de vraisemblance [1]. Mais dans plusieurs cas, ce critère de fidélité aux données ne permet pas a lui seul de produire une solution suffisamment stable à cause de la présence du bruit qui rend l'énergie du mesurande reconstitué très grande pour les hautes fréquences.

- **Méthodes probabilistes :** elles consistent à contraindre l'ensemble des solutions admissibles rendant certaines d'entre elles plus probables que d'autres. En effet, dans cette classe de méthodes, les mesurandes sont supposés être des réalisations de processus aléatoires et l'information *a priori* porte alors sur les lois de probabilités de ces processus. Le problème de reconstitution devient un problème classique d'estimation dans lequel cette information *a priori* est utilisée pour optimiser un estimateur optimal. Nous distinguons plusieurs méthodes qui utilisent ce principe combiné avec celui des méthodes variationnelles: méthode bayésienne [65], méthode à vraisemblance maximale [57], méthode à variance minimale [66] [67], filtre de Wiener [33], filtre de Kalman [26] [31].

- **Méthodes itératives :** compte tenu de leur simplicité, les algorithmes de reconstitution itératifs sont beaucoup utilisés par les chercheurs dans le domaine de la reconstitution des signaux. Ces algorithmes consistent à contraindre l'ensemble des solutions admissibles à celui des solutions qui peuvent être générées par une procédure itérative :

$$\mathbf{x}^{k+1} = \mathbf{J}(\mathbf{x}^k) \tag{2.2}$$

Où k est le nombre d'itération et **J** un opérateur qui peut être déterminé à partir du modèle de données. Mais l'on est confronté à des problèmes tels que la forte quantité de calcul dû au très grand nombre d'itérations demandés pour la convergence à la solution optimale. Or en utilisant une méthode itérative pour la résolution d'un problème, une règle d'arrêt doit être fixée pour terminer les itérations. En générale, cette règle d'arrêt est basée sur l'évolution de l'erreur d'estimation déterminée par la norme

$$\left\| \mathbf{x}^{k+1} - \mathbf{x}^{k} \right\| \leq \varepsilon \tag{2.3}$$

Bien sur l'évolution de l'erreur résiduelle qui est évaluée par la norme :

$$\left\| \mathbf{y} - \mathbf{M}\left[\mathbf{x}^{k}\right] \right\| \leq \gamma \tag{2.4}$$

où $\varepsilon > 0$ et $\gamma > 0$ sont des seuils d'arrêt choisis au début des itérations de façon empirique. Notons qu'on ne peut étudier d'une façon théorique que la dernière phase de la convergence lorsque $\hat{\mathbf{x}}^{k+1}$ est voisin de $\hat{\mathbf{x}}^{k}$. Ainsi, de nombreux algorithmes itératifs, tel que celui de Gold, Van Cittert, Jansson etc, ont été élaborés dans le domaine des traitements de signaux pour déterminer $\hat{\mathbf{x}}$. [9] [12] [33]

Ces méthodes sont couramment utilisées vue leur simplicité, la facilite d'incorporation des contraintes déterministes (telles que la positivité du signal et spectre borne). Cependant, en général elles nécessitent une forte quantité de calcul à cause du grand nombre d'itérations demandées pour l'approximation optimale de la solution.

- **Méthodes paramétriques:** elles consistent à contraindre l'ensemble des solutions admissibles à ceux des fonctions connues **x**(t; **p**), paramétrées en **p**. Le problème se réduit ainsi à déterminer le vecteur des paramètres **p**. [6] [28] [68]

La paramètrisation du signal reconstitue peut être linéaire ou non linéaire. La paramètrisation linéaire est plus désirée vu qu'elle conduit a des algorithmes de reconstitution simple. Cependant, la paramètrisation non linéaire, en général utilise moins de paramètres et offre plus de flexibilité en ce qui concerne la représentation du signal reconstitué. [1]

- **Méthodes de transformation:** elles consistent en la formulation des contraintes en utilisant différents domaines de transformation, en particulier le domaine spectral et le domaine cepstral.[24] [29]

Chaque classe de méthodes possède son champ d'applications spécial la ou d'autres échouent ou donnent des performances (exactitude, quantité de calcul) inférieures. Nous comprenons donc, que le choix d'une méthode de reconstitution dépend de l'application en question en premier lieu et des facilites offertes par les outils matériels et logiciels.

2.2 Algorithmes de référence utilisés

2.2.1 Méthode itérative de Van Cittert

L'algorithme de Van Cittert permet une bonne estimation du signal à reconstituer compte tenu de sa simplicité.

Ainsi, d'une manière itérative nous améliorons l'estimation à chaque itération k jusqu'à atteindre une estimation satisfaisante.

L'algorithme de Van Cittert est donc régit par les deux équations suivantes [41] :

$$\hat{\mathbf{x}}^0 = \tilde{\mathbf{y}} \tag{2.5}$$

$$\hat{\mathbf{x}}^{k+1} = \hat{\mathbf{x}}^k + [\tilde{\mathbf{y}} - \mathbf{g} * \hat{\mathbf{x}}^k] \tag{2.6}$$

Le signal observé $\tilde{\mathbf{y}}_n$ est pris comme la première approximation du signal estimé $\hat{\mathbf{x}}^0$. La correction sur l'estimation $\hat{\mathbf{x}}^k$ est obtenue à chaque itération k par l'erreur d'estimation sur le signal observé :

$$\tilde{\mathbf{y}} - \hat{\mathbf{y}} \quad \text{où} \quad \hat{\mathbf{y}} = \mathbf{g} * \hat{\mathbf{x}}^K \tag{2.7}$$

2.2.2 Méthode itérative de Jansson

À la différence de l'algorithme de Van Cittert, celui de Jansson utilise une fonction de relaxation **r** et se définie par les équations suivantes [9] :

$$\hat{\mathbf{x}}^0 = \tilde{\mathbf{y}} \tag{2.8}$$

$$\hat{\mathbf{x}}^{K+1} = \hat{\mathbf{x}}^K + \mathbf{r}\left(\hat{\mathbf{x}}^K\right).\left(\tilde{\mathbf{y}} - \mathbf{g} * \hat{\mathbf{x}}^K\right) \tag{2.9}$$

$$\mathbf{r}\left(\hat{\mathbf{x}}^K\right) = \mathrm{c}\left(1 - \left(\frac{2}{\mathrm{a}+\mathrm{b}}\left|\hat{\mathbf{x}}^K - \frac{\mathrm{a}+\mathrm{b}}{2}\right|\right)\right) \tag{2.10}$$

avec $\hat{\mathrm{x}}^k$ la valeur estimée de $\hat{\mathbf{x}}$ à l'itération k, c une constante empirique, a et b désignant respectivement la valeur minimale et maximale du signal recherché $\overset{o}{\mathbf{x}}$. Il faut noter que la fonction de relaxation **r** a pour but de contraindre l'estimation de $\overset{o}{\mathbf{x}}$ à rester

dans ses limites comprises entre a et b. Elle a aussi pour but de tenir compte des bruits qui entachent le signal afin d'obtenir une meilleure reconstitution de $\hat{\mathbf{x}}$.

2.2.3 Algorithme de reconstitution basé sur le filtre de Kalman

Le système de mesure est modélisé par l'équation d'état et l'équation d'observation suivantes [31] :

$$\mathbf{z}_{k+1} = \mathbf{\Phi}_n \mathbf{z}_k + \mathbf{b}_n \mathbf{w}_n \tag{2.11}$$

$$\mathbf{y}_{n+1} = \mathbf{h}_n^T \mathbf{z}_{n+1} + \mathbf{v}_n \tag{2.12}$$

Où $\mathbf{\Phi}_n$, $\mathbf{b}_n$ et $\mathbf{h}_n$ sont les matrices d'état et sont déterminées suivant le système étudié (voir chapitre 3), w_n et v_n est une réalisation d'une variable aléatoire $\underline{w}_n$ et $\underline{v}_n$ centrée (bruits blancs). Les variables aléatoires $\underline{w}_n$ et $\underline{v}_n$ sont supposées être non corrélées et des estimés *a priori* de leurs variances respectives σ_w^2 et σ_v^2 sont disponibles.

Le problème consiste maintenant à estimer le vecteur d'état z_n en se basant sur les observations $\{y_n\}$. Le filtre de Kalman est un estimateur linéaire qui permet d'obtenir l'estimé $\hat{z}_n$ du vecteur z_n à variance minimale, c'est à dire pour lequel:

$$\text{trace}\left\{(\hat{z}_n - z_n)(\hat{z}_n - z_n)^T\right\}$$

est minimale. L'estimé du vecteur d'état peut être obtenu par la récurrence suivante

$$\Sigma_{n+1}^{+} = \Phi_n \Sigma_n \Phi_n^T + \sigma_w^2 b_n b_n^T \tag{2.13}$$

$$\Sigma_{n+1} = \Sigma_{n+1}^{+} - \Sigma_{n+1}^{+} h_n h_n^T \Sigma_{n+1}^{+} \left[h_n \Sigma_{n+1}^{+} h_n^T + \sigma_v^2 \right]^{-1} \tag{2.14}$$

$$K_{n+1} = \Sigma_{n+1} h_n / \sigma_v^2 \tag{2.15}$$

$$\hat{z}_{n+1} = \Phi_n \hat{z}_n + K_{n+1} \left(\tilde{y}_n - h_n^T \Phi_n \hat{z}_n \right) \tag{2.16}$$

2.3 Méthodologie de l'évaluation des algorithmes de reconstitution de mesurandes dynamiques

2.3.1 Principe

Le choix d'une méthode de reconstitution dépend essentiellement des exigences concernant le système de mesure: son exactitude, sa vitesse et sa complexité. Ainsi, la qualité de la mise en œuvre d'un algorithme de reconstitution de mesurande se mesure à l'aide de ces trois exigences, à savoir:

1. l'exactitude de reconstitution;
2. la rapidité en réponse et la reconstitution en temps réel;
3. la complexité de calcul et espace de mémoire requise.

Ainsi, l'étude des algorithmes de reconstitution sera essentiellement portée à leur évaluation en utilisant ces critères et en utilisant aussi bien des données synthétiques que expérimentales.

Selon le domaine d'applications, les performances des algorithmes de reconstitution proposés seront comparées à celles des algorithmes de référence utilisés dans ce domaine.

2.3.2 Génération des données synthétiques

Pour l'application en spectrométrie, les données de mesure synthétiques exactes y(λ) sont celles utilisées par Crilly [69]. Elles sont simulées en utilisant la relation:

$$\mathbf{y}(\lambda) = \mathbf{g}(\lambda) * \mathbf{x}(\lambda) \tag{2.17}$$

où nous considérons que le mesurande exacte x(λ) et la réponse impulsionnelle du spectromètre g(λ) sont connus à l'avance. La séquence $\{\tilde{\mathbf{y}}_n\}$ est générée selon l'équation:

$$\tilde{\mathbf{y}}_n = \mathbf{y}(\lambda_n) + \eta_n \qquad n=0,1, \ldots, N \tag{2.18}$$

En utilisant une séquence de nombres pseudo-aléatoires $\{\eta_n\}$ normalement distribués, de moyenne nulle et de variance connue σ_n^2. Ainsi, la connaissance de $\{\tilde{\mathbf{y}}_n\}$ et de g(λ) nous permettra de reconstituer le mesurande.

2.3.3 Acquisition des données réelles

Les données spectrométriques réelles ont été acquises à l'aide d'un spectromètre avec les paramètres instrumentales suivantes:

Start WL : 350 nm

Stop WL : 550 nm

Largeur du fente spectrale : 2nm

Test points : 501 points.

D'autres données réelles de télécommunication optique prélevées à l'aide d'un spectromètre sont aussi utilisées pour tester l'algorithme.

les paramètres du spectromètre sont les suivants :

Start WL : 1553 nm

Stop WL : 1558 nm

Sensitivité : -70 dB

Test points : 1001 points

2.3.4 Procédure d'évaluation des algorithmes

Suivant l'application traitée, un algorithme de reconstitution de mesurande peut être évalué selon la procédure suivante [1]:

- Un problème test: il s'agit de prendre un exemple commun de données de mesure synthétiques traitées dans la littérature et qui représentent de façon raisonnable la situation réelle. Ces données sont générées en supposant que le modèle de données et le mesurande sont connus de façon exacte.

- Exactitude de reconstitution: l'erreur de reconstitution entre la valeur exacte du mesurande et celle estimée caractérise principalement l'exactitude de l'algorithme de reconstitution. Elle est évaluée selon l'erreur quadratique moyenne relatif (EQMR):

$$\varepsilon = \frac{\left\| \overset{o}{\mathbf{x}} - \hat{\mathbf{x}} \right\|_2}{\left\| \overset{o}{\mathbf{x}} \right\|_2} \tag{2.19}$$

• Vitesse et complexité de calcul: le nombre d'opérations arithmétiques (additions et multiplications) nécessité par l'algorithme de reconstitution.

• Espaces des paramètres de l'algorithme : il s'agit d'analyser l'influence de la variation des paramètres libres de l'algorithme de reconstitution sur l'erreur de reconstitution. Cette analyse nous informe sur le domaine où l'algorithme donne les meilleurs performances. Nous pouvons cité comme paramètres libres : le nombre de points du mesurande, la variance du bruit de mesure σ_n^2, paramètre de régularisation, etc.

Chapitre 3

ALGORITHME DE RECONSTITUTION PROPOSÉ

3.1 Modèle mathématique du système de mesure

3.1.1 Modèle du système de mesure [2]

Nous considérons que le système de mesure est examiné périodiquement et que l'état à l'instant t_k est linéairement lié à l'état à l'instant t_{k+1} par les matrices A(k) et B(k), pondérant respectivement l'état précédent et un bruit aléatoire borné w(k). Le modèle du système est donc représenté par l'équation d'état :

$$\mathbf{z}(k+1) = \mathbf{A}(k)\mathbf{z}(k) + \mathbf{B}(k)\mathbf{w}(k) \tag{3.1}$$

Avec **z** est le vecteur d'état choisi convenablement selon la réponse impulsionnelle du système de mesure. (cf. 3.1.3 et 3.1.4)

3.1.2 Modèle de mesure [2]

Supposons maintenant un certain nombre d'observations y_0, y_1, y_2, ... ,y_n. Ces meures sont linéairement liées au vecteur d'état décrivant le processus par la matrice de

mesure C(k). Les mesures sont aussi perturbées par un bruit dit de mesure v(k) qui sera supposé borné. Ceci nous donne le modèle de mesure suivant :

$$y(k) = C(k)z(k) + v(k) \tag{3.2}$$

On suppose que le système est stationnaire c'est-à-dire la dynamique du système reste la même d'une mesure à l'autre, ce qui entraîne que les matrices A(k), B(k) et C(k) sont constantes durant la mesure.

3.1.3 Réponse impulsionnelle non gaussoïdale

Soit la réponse impulsionnelle donnée par la fonction ci-dessous :

$$g(\lambda) = \begin{cases} 6(e^{-0.7\lambda} - e^{-0.9\lambda}) & \text{si} \quad \lambda \geq 0 \\ 0 & \text{si} \quad \lambda < 0 \end{cases} \tag{3.3}$$

Le temps d'échantillonnage est supposé $\Delta\lambda = 25/128$.

La transformé en z de **g** est la suivante :

$$G(z) = A(\alpha - \beta).\frac{z}{z^2 - (\alpha + \beta)z + \alpha\beta} = \frac{y(z)}{x(z)} \tag{3.4}$$

Avec A =6 , $\alpha = e^{-0.7\Delta\lambda}$, $\beta = e^{-0.9\Delta\lambda}$.

Ce qui nous conduit au modèle d'état (3.1)-(3.2) avec:

$$A(k)=\begin{pmatrix}\alpha+\beta & -\alpha\beta & A(\alpha-\beta)\\ 1 & 0 & 0\\ 0 & 0 & 1\end{pmatrix},\quad B(k)=\begin{pmatrix}0 & 0 & 1\end{pmatrix}^T$$
$$\text{et}\quad C(k)=\begin{pmatrix}\alpha+\beta & -\alpha\beta & A(\alpha-\beta)\end{pmatrix}. \tag{3.5}$$

Le vecteur d'état du système égale à $z_k=(y_{k-1} \quad y_{k-2} \quad x_{k-1})^T$. De plus, la qualité de correction est évaluée selon l'erreur relative des moindres carrés définie précédemment.

3.1.4 Réponse impulsionnelle gaussoïdale

La réponse impulsionnelle du système de mesure est dans ce cas une gaussoïde parfaite et centrée, elle n'admet pas de transformé de Laplace ni de Fourrier exacte, ceci conduit à introduire des méthodes d'approximations de celle-ci (RLS, FFT , etc.).

L'approche classique [5] engendre beaucoup de calcul à cause de la grande dimension des matrices d'état qui est égale au nombre d'échantillons. Pour surmonter ce problème, la réponse impulsionnelle du système peut être approximée sous la forme d'une convolution de deux fonctions notées g_+ et g_- tel que:

$$g(\lambda) \cong g_+(\lambda) * g_-(\lambda) \tag{3.6}$$

Avec

$$g_+(\lambda)=\begin{cases}\sum_{k=1}^{Mg} a_k e^{-b_k\lambda} & \lambda \geq 0\\ 0 & \lambda < 0\end{cases} \qquad \text{et} \qquad g_-(\lambda)=g_+(-\lambda) \tag{3.7}$$

En premier lieu, l'algorithme utilisant la fonction $g_-(\lambda)$ est appliqué sur la mesure brute et un résultat intermédiaire est obtenu, après avoir inverser l'ordre du résultat

obtenu, il est utilisé comme entrée de l'algorithme de reconstitution une autre fois avec la fonction $g_+(\lambda)$ pour obtenir le résultat final de reconstitution.

Soit la réponse impulsionnelle définit comme suit :

$$g(\lambda) = 0.0525 \exp\left(-\frac{\lambda^2}{2\sigma^2}\right) \quad \text{avec } \sigma = 8 \tag{3.8}$$

Prenons le cas où Mg=3, on aboutit aux coefficients a_k et b_k suivants :

$a_1 = 0.1926$; $a_2 = -0.1912$; $a_3 = 0.0206$;

$b_1 = 0.1327$; $b_2 = 0.4345$; $b_3 = 2.8191$.

Ce qui nous conduit au modèle d'état ARMA (3.1)-(3.2) avec:

$$\mathbf{A}(k) = \begin{pmatrix} -d_2 & -d_1 & -d_0 & n_2 & n_1 & n_0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \tag{3.9}$$

$$\mathbf{B}(k) = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}^T$$
$$\text{et } \mathbf{C}(k) = \begin{pmatrix} -d_2 & -d_1 & -d_0 & n_2 & n_1 & n_0 \end{pmatrix} \tag{3.10}$$

Avec,

$p_1 = e^{-b_1 \Delta\lambda}$; $p_2 = e^{-b_2 \Delta\lambda}$; $p_3 = e^{-b_3 \Delta\lambda}$;

$k_1 = \left(\frac{a_1}{b_1}\right)(1 - p_1)$; $k_2 = \left(\frac{a_2}{b_2}\right)(1 - p_2)$; $k_3 = \left(\frac{a_3}{b_3}\right)(1 - p_3)$;

$n_2 = k_1 + k_2 + k_3$;

$n_1 = -(k_1.p_2 + k_1.p_3 + k_2.p_1 + k_2.p_3 + k_3.p_1 + k_3.p_2)$;

$n_0 = k_1.p_2.p_3 + k_2.p_1.p_3 + k_3.p_1.p_2$;

$d_2 = -p_1 - p_2 - p_3$;

$d_1 = p_1.p_2 + p_1.p_3 + p_2.p_3$;

$d_0 = -p_1.p_2.p_3$;

La réponse impulsionnelle $\mathbf{g}(\lambda)$ et son approximation $\mathbf{g}_+(\lambda) * \mathbf{g}_-(\lambda)$ sont montrées sur la figure 3.1, l'erreur quadratique moyen relative (EQMR) égale à 0.1307. Notons que les simulations sont tournées sous le logiciel Matlab de MathWorks, Inc. Les figures sont aussi obtenues sous le même logiciel.

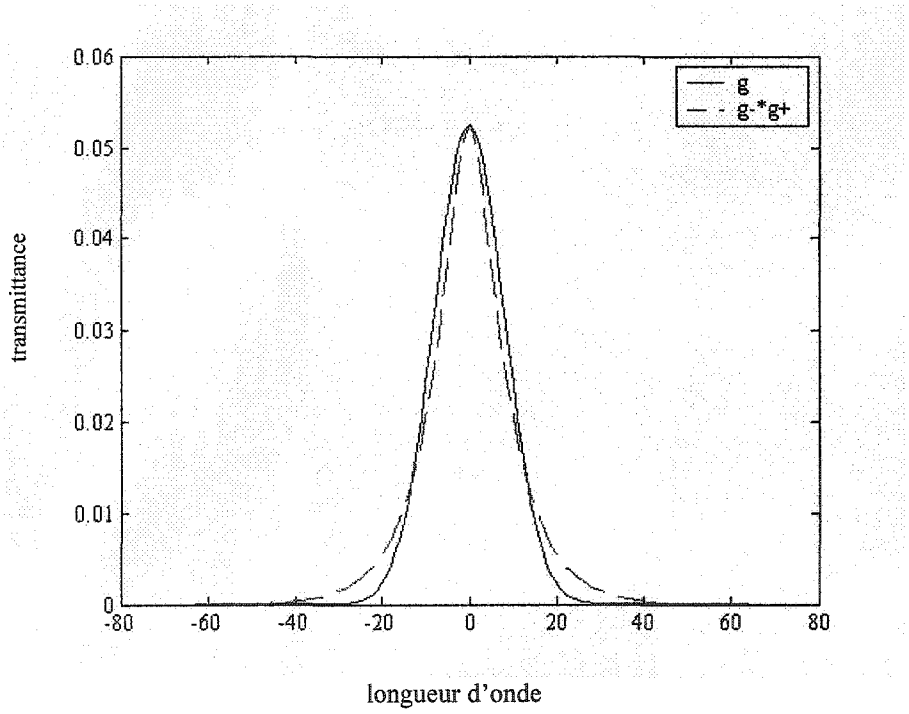


Figure 3.1 :Réponse impulsionnelle g et son approximation g-*g+

3.2 Algorithme de reconstitution basé sur le filtre H∞

Cet algorithme [2] est basé sur le filtrage H∞ avec une approche quadratique (LQ).

L'apport de ce filtre dans le domaine de traitement de signaux et spécialement en spectrométrie se résume en deux points : la connaissance statistiques des perturbations n'a aucun effet sur sa conception, la seule condition est qu'ils sont bornés, en plus, ce filtre est robuste en horizon infini contrairement à d'autres filtres tels que Wiener et Kalman [2]. D'autre part, les algorithmes itératifs tel que Van Cittert et Jasson nécessitent beaucoup de calcul pour obtenir une bonne précision contrairement a l'algorithme basé sur le filtrage H∞ qui nécessite beaucoup moins d'opérations arithmétiques.

Considérons un système de mesure de modèle d'état définie par (3.1) et (3.2), et supposons que $(\mathbf{A}(k),\mathbf{B}(k))$ est contrôlable et $(\mathbf{C}(k),\mathbf{A}(k))$ est observable. Soit l'ensemble des mesures précédentes définie par $\mathbf{Y}_k = (\mathbf{y}_k,\ 0 \le k \le N-1)$. L'estimé de l'état $\hat{\mathbf{z}}_k$ à l'instant k est calculé en se basant sur l'histoire de mesure $\mathbf{Y}_k$ jusqu'à N-1. Nous nous intéressons à estimé une combinaisons linéaire de l'état $\mathbf{z}_k$ [2].

$$\mathbf{x}_k = \mathbf{L}_k \mathbf{z}_k \tag{3.11}$$

Le filtre H∞ est conçu pour fournir une erreur d'estimation uniformément petite, $\mathbf{e}_k = \mathbf{x}_k - \hat{\mathbf{x}}_k$, pour n'importe quel bruit w_k, $v_k \in L_2$ et n'importe quelle état initiale $\mathbf{z}_0 \in \mathbf{R}^2$. La mesure de performance (fonction coût) est donnée par [2]:

$$J(\gamma) = \frac{\sum_{k=0}^{N-1} \left\| \mathbf{x}_k - \hat{\mathbf{x}}_k \right\|^2_{\mathbf{Q}(k)}}{\left\| \mathbf{z}_0 - \hat{\mathbf{z}}_0 \right\|^2_{\mathbf{p}^{-1}} + \sum_{k=0}^{N-1} \left(\left\| w \right\|^2_{\mathbf{w}_k^{-1}} + \left\| v_k \right\|^2_{\mathbf{v}_k^{-1}} \right)} \tag{3.12}$$

Où les matrices $\mathbf{Q}_k \ge 0$, $\mathbf{p}_0^{-1} > 0$, $\mathbf{W}_k^{-1} > 0$ et $\mathbf{V}_k^{-1} > 0$ sont des matrices de pondération, avec la notation: $\| \mathbf{r} \|^2_{\mathbf{M}} = \mathbf{r}^T \ \mathbf{M} \ \mathbf{r}$. L'estimateur sous optimal doit

satisfaire: $\sup J(\gamma) < 1/\gamma$, avec $\gamma>0$ un niveau prescrit d'atténuation de bruit. Ce filtre peut être interprété comme un problème minmax suivant :

$$\min_{\hat{\mathbf{x}}_k} \max_{(v_k, w_k, \mathbf{z}_0)} J(\lambda) = -\frac{1}{2\gamma}\left\|\mathbf{z}_0 - \hat{\mathbf{z}}_0\right\|^2_{\mathbf{p}^{-1}} + \frac{1}{2}\left[\sum_{k=0}^{N-1}\left\|\mathbf{x}_k - \hat{\mathbf{x}}_k\right\|^2_{\mathbf{Q}(k)} - \frac{1}{\gamma}\left(\sum_{k=0}^{N-1}\left(\|w\|^2_{\mathbf{w}_k^{-1}} + \|v_k\|^2_{\mathbf{v}_k^{-1}}\right)\right)\right] \tag{3.13}$$

Le modèle du filtre est donné par l'équation d'état (3.14):

$$\begin{aligned}\hat{\mathbf{z}}(k+1) &= \mathbf{A}(k)\hat{\mathbf{z}}(k) + \mathbf{K}(k)(y(k) - \mathbf{C}(k)\hat{\mathbf{z}}(k)) \\ \hat{\mathbf{x}}(k) &= \mathbf{L}(k)\hat{\mathbf{z}}(k)\end{aligned} \tag{3.14}$$

Le gain du filtre H∞ est donné par l'équation suivante (3.15) [2]:

$$\begin{aligned}\mathbf{K}(k+1) = \mathbf{A}(k)\mathbf{P}(k)(\mathrm{I} - \gamma\overline{\mathbf{Q}}(k)\mathbf{P}(k) + \mathbf{C}^{\mathrm{T}}(k)\mathbf{V}^{-1}(k)\mathbf{C}(k)\mathbf{P}(k))^{-1} \\ \mathbf{C}^{\mathrm{T}}(k)\mathbf{V}^{-1}(k)\end{aligned} \tag{3.15}$$

À condition de l'existence d'une matrice $\mathbf{P}(k)$, k=1, … , N, définie positive qui est la solution de l'équation de Riccati (3.16) [2].

$$\begin{aligned}\mathbf{P}(k+1) = \mathbf{A}(k)\mathbf{P}(k)\mathbf{A}^{\mathrm{T}}(k) + \mathbf{B}(k)\mathbf{W}(k)\mathbf{B}^{\mathrm{T}}(k) + \\ \mathbf{A}(k)\mathbf{P}(k)(\gamma\overline{\mathbf{Q}}(k) - \mathbf{C}^{\mathrm{T}}(k)\mathbf{V}^{-1}(k)\mathbf{C}(k)) \\ \left[\mathbf{I} - \mathbf{P}(k)(\gamma\overline{\mathbf{Q}}(k) - \mathbf{C}^{\mathrm{T}}(k)\mathbf{V}^{-1}(k)\mathbf{C}(k))\right]^{-1}\mathbf{P}(k)\mathbf{A}^{\mathrm{T}}(k)\end{aligned} \tag{3.16}$$

Avec comme condition initiale $\mathbf{P}(0)=\mathbf{p}_0$ et $\overline{\mathrm{Q}}(k) = \mathrm{L}(k)^{\mathrm{T}}.\mathrm{Q}(k).\mathrm{L}(k)$, le signal $\hat{\mathbf{x}}(k)$ représente l'estimé de $\mathbf{x}(k)$.

3.3 Résultats de l'évaluation de l'algorithme

3.3.1 Résultats avec des données synthétiques

3.3.1.1 Le mesurande contient seulement deux pics

Pour illustrer l'algorithme proposé, nous avons utilisé un exemple de signaux spectrométriques synthétiques traités souvent dans la littérature[1][6][7][31]. Le mesurande exacte est formé par deux pics comme montre la figure (3.2), il sont définis par :

$$\overset{\circ}{x}(\lambda) = 2\exp\left(-\frac{(\lambda-45)^2}{2\sigma_1^2}\right) + 6\exp\left(-\frac{(\lambda-65)^2}{2\sigma_2^2}\right) \tag{3.17}$$

avec $\sigma_1 = \sigma_2 = 2.25$

On a donc $\overset{\circ}{y}(\lambda) = g(\lambda) * \overset{\circ}{x}(\lambda)$ et $\tilde{y}(\lambda_k) = \overset{o}{y}(\lambda_k) + v_k$, v_k est supposé un bruit blanc pour rendre applicable les autres algorithmes de référence et surtout le filtre de Kalman. Le temps d'échantillonnage est pris égale à l'unité $\Delta\lambda = 1$ pour le cas d'une fonction gausoïdale, et pour l'autre cas on a pris $\Delta\lambda = 25/128$.

$\overset{\circ}{x}(\lambda)$ et $\overset{\circ}{y}(\lambda)$ sont montrés dans la figure (3.2a) et (3.2b), l'axe des x représente la longueur d'onde.

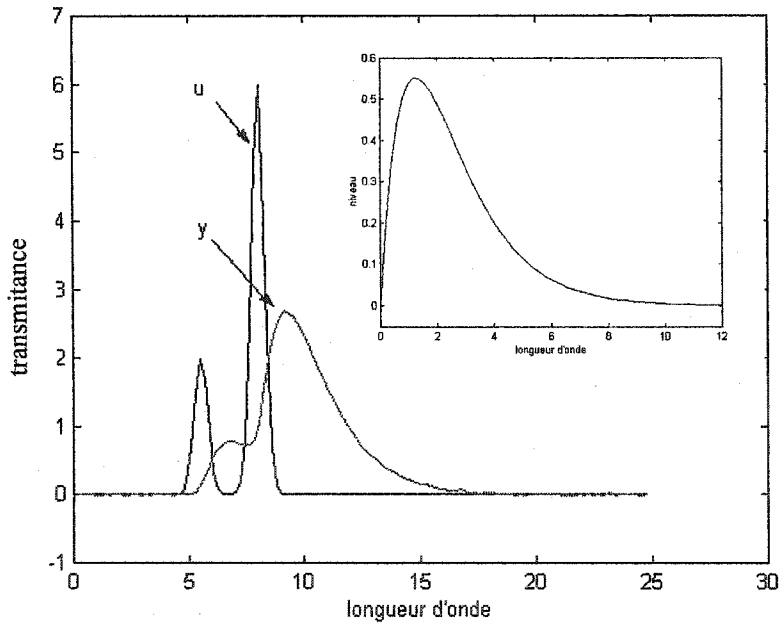
Le niveau de perturbation sur les données est caractérisé par le rapport signal sur bruit :

$$\mathrm{SNR} = 10\log\frac{\| \overset{\circ}{y} \|_2^2}{\| v \|_2^2} \tag{3.18}$$

la qualité de correction est évaluée selon l'erreur quadratique moyen relative des moindres carrés (EQMR):

$$\varepsilon = \frac{\| \hat{x}(\lambda) - \overset{\circ}{x}(\lambda) \|_2}{\| \overset{\circ}{x}(\lambda) \|_2} \tag{3.19}$$

(a)



(b)

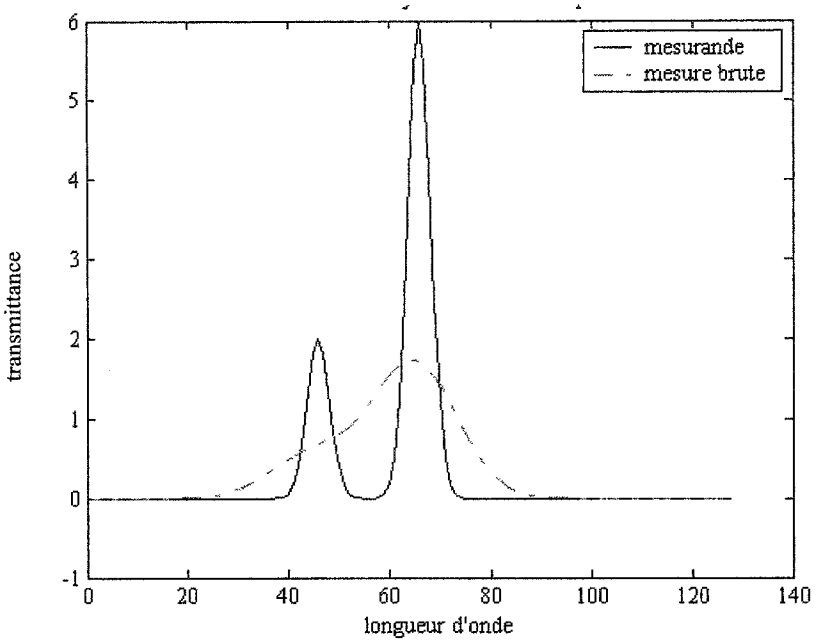


Figure 3.2 : Le mesurande et la mesure brute du système pour

(a) une fonction non gausoïdale et (b) une fonction gausoïdale

a) Fonction de transfert est non gausoïdale

Les résultats de l'algorithme proposé sont comparés avec les résultats du filtre de Kalman, selon des différents SNR. Dans le tableau 3.1, notre filtre, non optimalisé, donne des résultats très bons comparé avec les autres algorithmes, de plus la très petite valeur de la variance de l'erreur de l'estimation montre la robustesse du filtrage H∞ par rapport autres algorithmes.

La figure (3.3) montre les résultats de corrections obtenues par l'algorithme proposé pour SNR=38.6dB

Tableau 3.1: Moyenne de 50 réalisations de l'erreur quadratique moyen relative ($\bar{\varepsilon}$) obtenu par H∞, Kalman, Jasson et Van Cittert.

Algorithme		SNR (dB)		
		58.6	**38.6**	**18.6**
H∞	γ	1.1	1.1	1.1
	$\bar{\varepsilon}$	0.0482	0.1587	0.4393
	σ_ε	$1.57\ 10^{-5}$	0.0117	$7.97\ 10^{-4}$
Kalman	β_{opt}	10^6	10^5	10^4
	$\bar{\varepsilon}$	0.0341	0.1374	0.4345
	σ_ε	0.0019	0.0113	0.0412
Jasson	K_{opt}	202	199	76
	$\bar{\varepsilon}$	0.1581	0.1721	0.5536
	σ_ε	0.0012	0.0138	0.0872
Van Cittert	K_{opt}	103	88	16
	$\bar{\varepsilon}$	0.7962	0.8003	0.8834
	σ_ε	$7.27\ 10^{-4}$	0.0065	0.0216

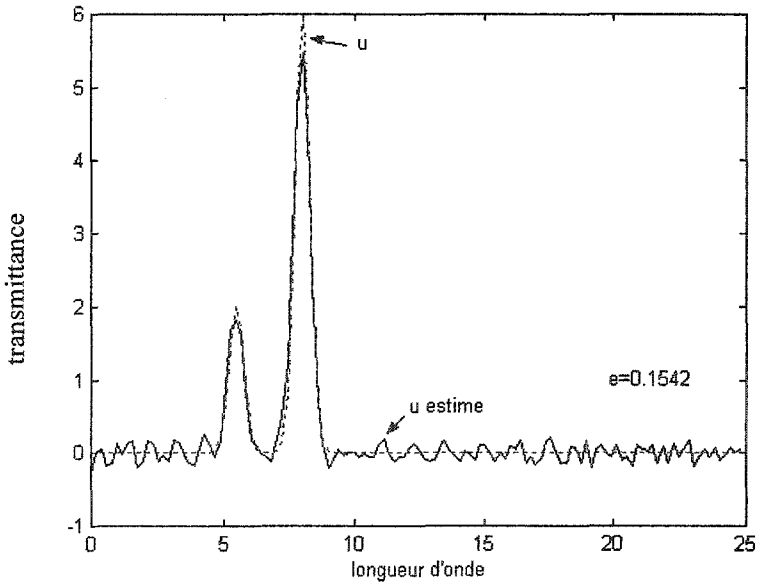


Figure 3.3: Résultat de correction obtenu par le filtre H∞, M=64, N=128, $\Delta\lambda$=25/128 et SNR=38.6dB

b) **La fonction de transfert est gausoïdale**

On applique le filtre au système de fonction de transfert définie à (3.8), la figure (3.4) montre le mesurande et son estimé pour un niveau de bruit SNR=64.6dB.

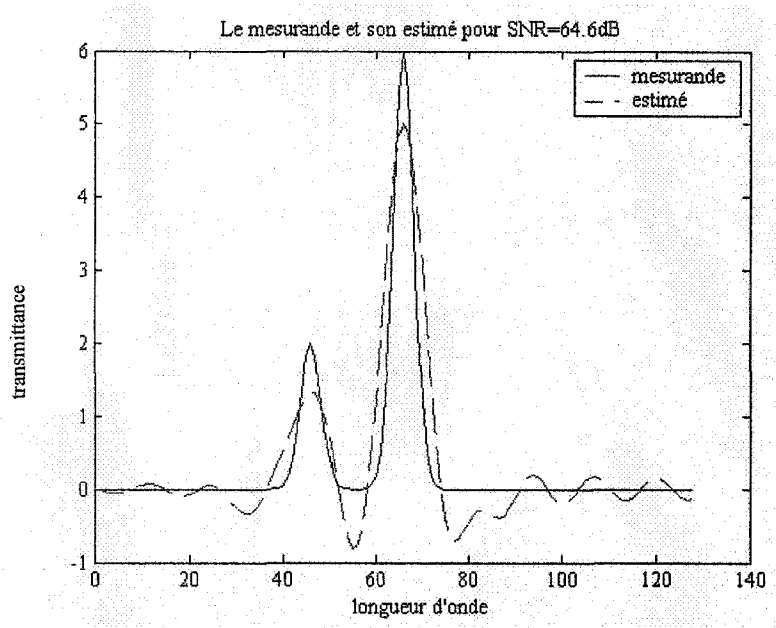


Figure 3.4 : Signal à mesurer et son estimé par le filtre H∞, M=N=128, Δλ=1 et SNR=64.6dB

Le tableau (3.2) contient la moyenne de 50 réalisations de l'erreur moyen relative des moindre carrés ε suivant 4 niveau de bruits, la méthode est comparée avec le filtre de Kalman, Gold (nombre d'itérations est 500) et Jasson_L. Ces deux dernières méthodes sont des nouvelles versions performantes des algorithmes origines et sont dernièrement publiées. Notons que le filtre de Kalman est basé sur le même modèle que le filtre H∞, et non sur le modèle classique du modèle d'état. ce tableau montre que notre algorithme est plus performant que le filtre de Kalman et donne des résultats très proche à l'algorithme

de Gold, l'algorithme de Jasson_L reste le plus puissant pour approcher l'estimé, mais le problème de ces deux derniers méthodes est le nombre important d'opérations arithmétiques qui est un point faible pour l'estimation en temps réelle. En conclusion, notre algorithme est puissant est donne de très bon résultats d'estimation et rapide en même temps, ce que lui permet de pouvoir prendre la place du filtre de Kalman pour l'estimation en temps réelle.

De plus, la figure 3.5 montre que l'erreur décroît exponentiellement quand le SNR croit.

Tableau 3.2 : Moyenne de 50 réalisations de l'erreur relative des moindres carrés obtenu par H∞, Kalman, Jasson_L et Gold.

Algorithme	**SNR (dB)**			
	64.6	**54.6**	**44.6**	**34.6**
H∞	0.4189	0.4405	0.4797	0.5206
Kalman	0.5268	0.5287	0.5318	0.5420
Jasson_L	0.0398	0.0400	0.04977	0.06925
Gold	0.2375	0.2383	0.2360	0.2363

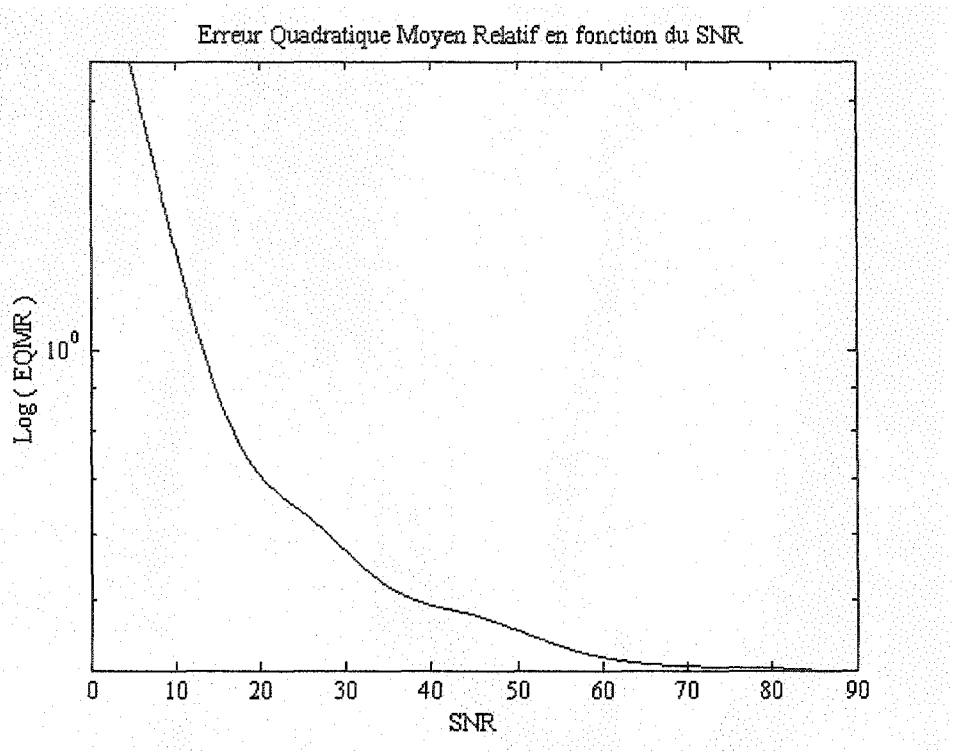


Figure 3.5 : Erreur quadratique moyen relatif suivant le niveau de bruit

c) Espace des paramètres

Pour analyser l'effet des paramètres sur l'optimalité du filtre, on varie ses paramètres libres et on note leurs influences sur l'erreur d'estimation, les figures 3.6 à 3.9 montrent cette influence pour SNR égale à 64.6dB, 54.6dB, 44.6dB et 34.6dB.

Ces paramètres sont :

$\mathbf{p}_0$, la valeur initiale de la matrice solution de l'équation de Riccati;

γ, définit le degré de la sous-optimalité du filtre.

W, la matrice de pondération du bruit agissant sur l'état du système (w).

V, la matrice de pondération du bruit additif à la sortie du mesure(v).

Q, matrice de pondération de l'erreur d'estimation :cette matrice n'a pas d'effet directe, car il est compensé pas γ.

D'après ces figures, on constate que l'influence du paramètre p0 est minime, tandis que les autres paramètres (γ, **W**, **V**) jouent un rôle important à minimiser l'erreur d'estimation (grande plage de variation de l'erreur quadratique moyen relative). Pour cela, il faut chercher pour chaque paramètres la valeur qui minimise le mieux l'erreur d'estimation, cette valeur optimum reste presque invariant quelque soit le niveau de bruit qui montre bien la robustesse de l'algorithme en présence du bruit de statistique inconnu. Une recherche des valeurs optimum suffit pour un système de mesure quelque soit le niveau de bruit contrairement au filtre de Kalman [35].

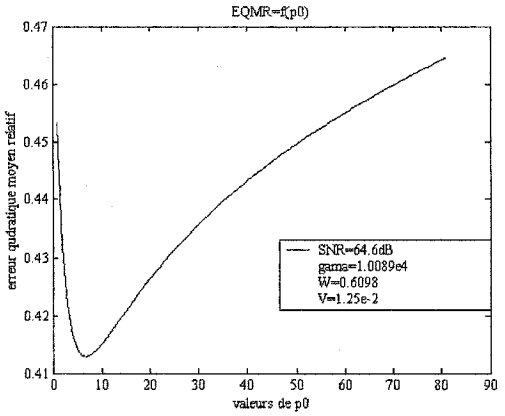


figure 3.6-a

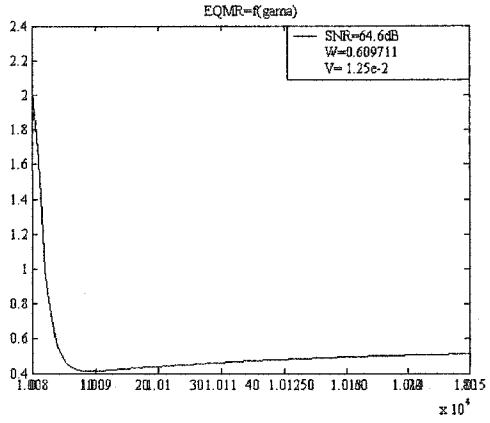


figure 3.6-b

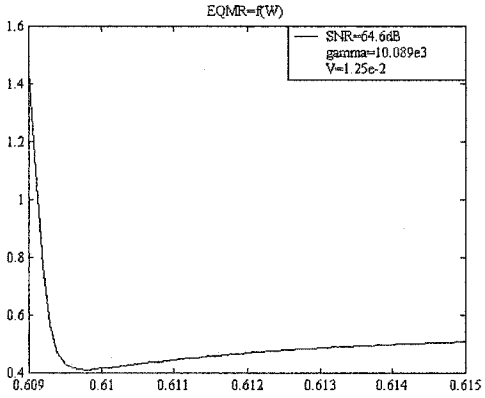


figure 3.6-c

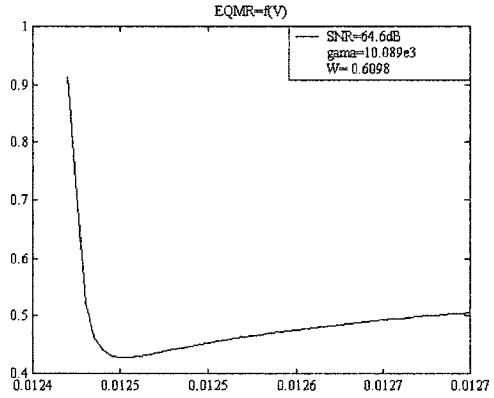


figure 3.6-d

Figure 3.6 : Effet de la variation des paramètres sur le résultat du filtre, SNR=64.6dB

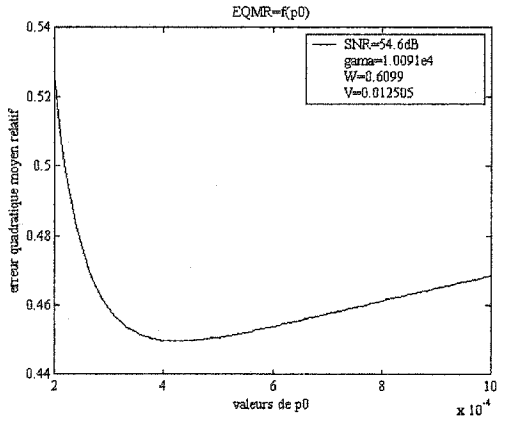


figure 3.7-a

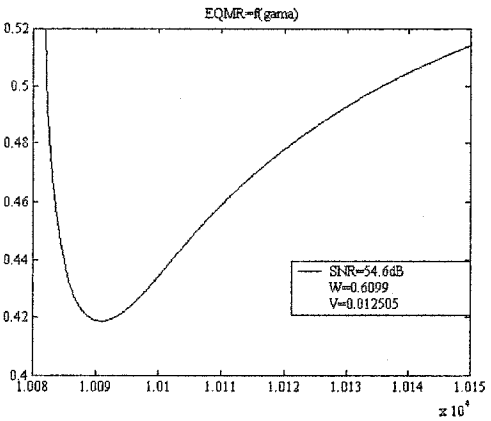


figure 3.7-b

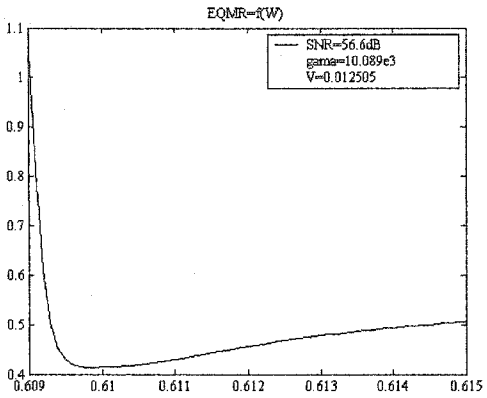


figure 3.7-c

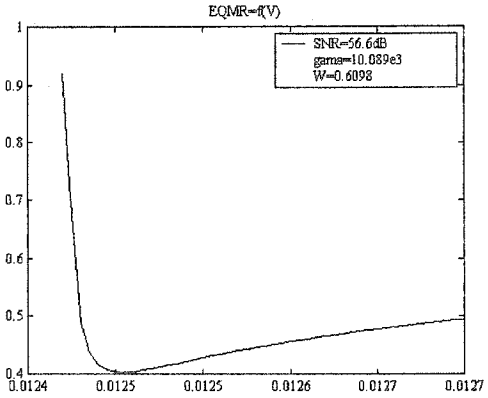


figure 3.7-d

Figure 3.7 : Effet de la variation des paramètres sur le résultat du filtre, SNR=54.6dB

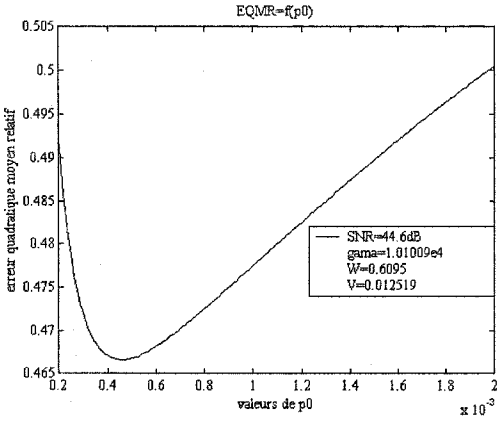


figure 3.8-a

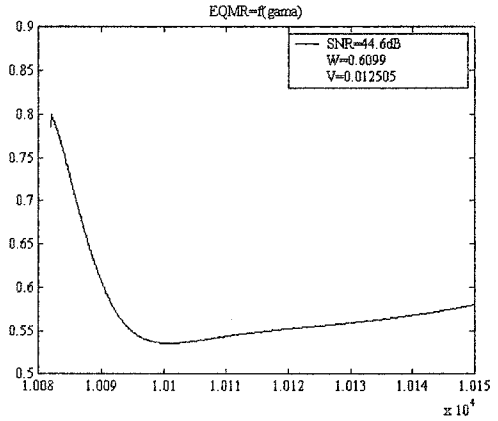


figure 3.8-b

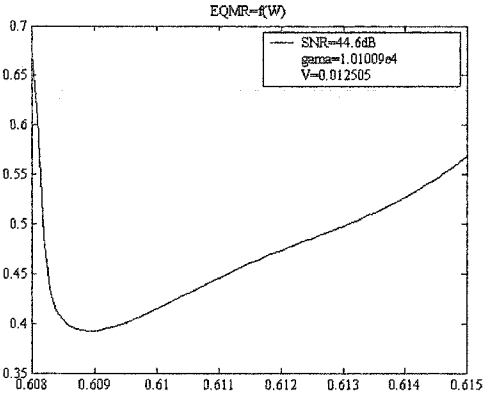


figure 3.8-c

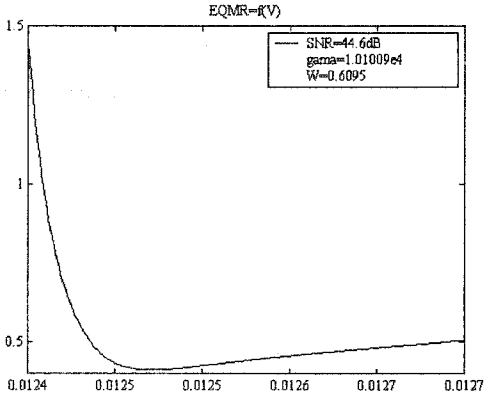


figure 3.8-d

Figure 3.8 : Effet de la variation des paramètres sur le résultat du filtre, SNR=44.6dB

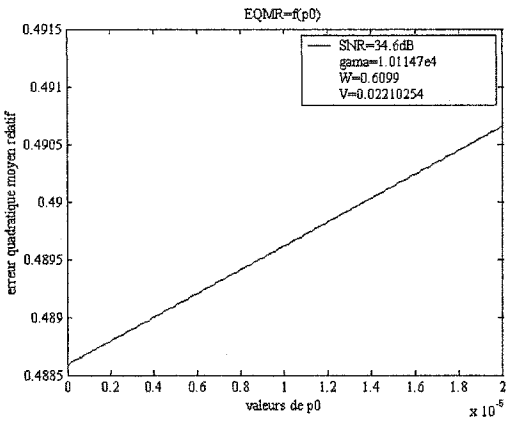


figure 3.9-a

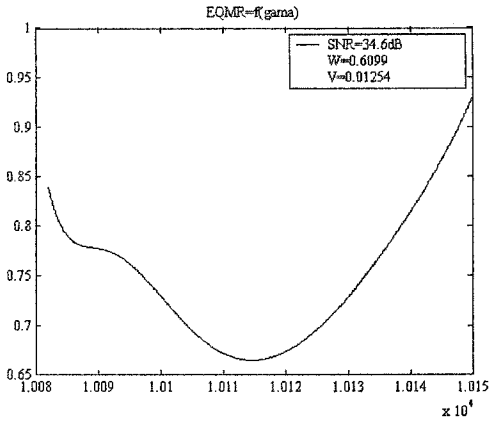


figure 3.9-b

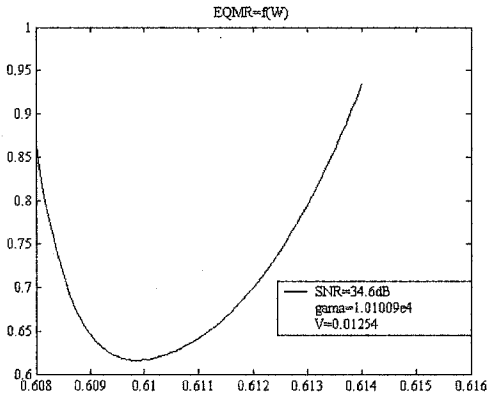


figure 3.9-c

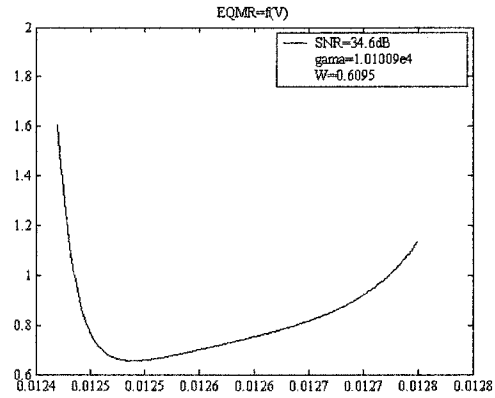


figure 3.9-d

Figure 3.9 : Effet de la variation des paramètres sur le résultat du filtre, SNR=34.6dB

3.3.1.2 Cas d'un mesurande contenant plusieurs pics

Soit un système de mesure, de réponse impulsionnelle de la forme $g(\lambda) = 0.07\exp\left(-\frac{\lambda^2}{2\sigma^2}\right)$ avec σ prend les valeurs 8, 4 et 2. (voir fig. 3.10).

Dans cette étude on analysera l'effet de la largeur de la bande passante du système de mesure sur le résultat de reconstitution

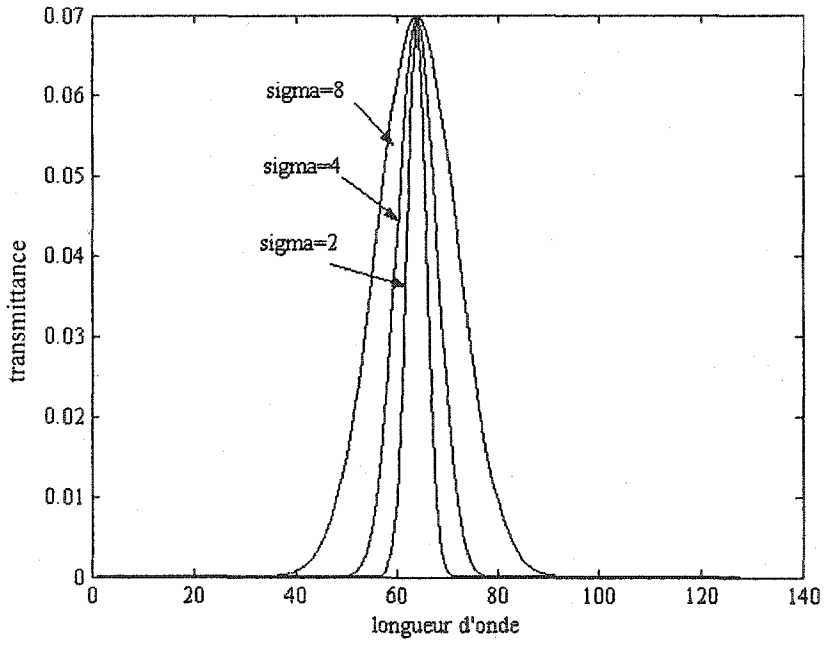


Figure 3.10 : Réponse impulsionnelle du système de mesure

Soit le signal a mesure à la forme :

$$\overset{\circ}{\mathbf{x}}(\lambda) = 2\exp\left(-\frac{(\lambda-25)^2}{2\sigma_1^2}\right) + 4\exp\left(-\frac{(\lambda-45)^2}{2\sigma_2^2}\right) + 6\exp\left(-\frac{(\lambda-65)^2}{2\sigma_3^2}\right) + 4\exp\left(-\frac{(\lambda-85)^2}{2\sigma_4^2}\right) + 7\exp\left(-\frac{(\lambda-95)^2}{2\sigma_5^2}\right) + 10\exp\left(-\frac{(\lambda-115)^2}{2\sigma_6^2}\right) \quad (3.20)$$

avec $\sigma_1 = \sigma_2 = \sigma_4 = \sigma_5 = 2.25$, $\sigma_3 = 1.5$ et $\sigma_6 = 1.25$

On a donc $\overset{\circ}{y}(\lambda) = g(\lambda) * \overset{\circ}{x}(\lambda)$ et $\tilde{y}(\lambda_k) = \overset{\circ}{y}(\lambda_k) + v_k$, v_k est supposé un bruit blanc. De plus, le temps d'échantillonnage est pris égale à l'unité $\Delta\lambda = 1$

$\overset{\circ}{x}(\lambda)$ et $\overset{\circ}{y}(\lambda)$ sont montrés dans la figure 3.11 (resp. figure 3.13), pour le niveau de bruit SNR=64.6dB et σ=8 (resp. SNR=34.6dB et σ=2)

La figure 3.12 (resp. Figure 3.14) montre le mesurande et son estimé pour le niveau de bruit SNR=64.6dB et σ=8 (resp. SNR=34.6dB et σ=2)

Le tableau (3.3) contient la moyenne de 50 réalisations de l'erreur relative des moindre carrés ε suivant 4 niveau de bruits et suivant la bande passante de la fonction de transfert du système, la méthode est comparée avec le filtre de Kalman, Gold(avec 1000 itérations) et Jasson_L.

Les figures montrent bien l'effet de la largeur passante du système de mesure sur le résultat de l'estimation. En effet, la précision de l'estimation est de plus en plus meilleur si σ diminue, ce qui veut dire que la réponse impulsionnelle tend vers l'impulsion de Dirac.

Le tableau (3.3) montre que notre algorithme est plus performant que le filtre de Kalman et donne des résultats très proche à l'algorithme de Gold quand σ croit, l'algorithme de Jasson_L reste le plus puissant pour approcher l'estimé, mais le problème de ces deux derniers méthodes est le nombre important d'opérations arithmétiques qui est un point faible pour l'estimation en temps réelle.

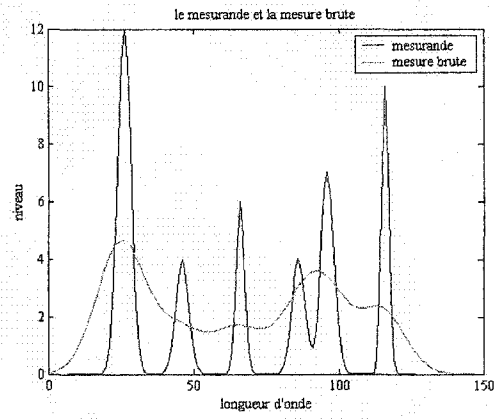


Figure 3.11 : Mesurande et la sortie du système de mesure pour σ=8 et SNR=64.6dB

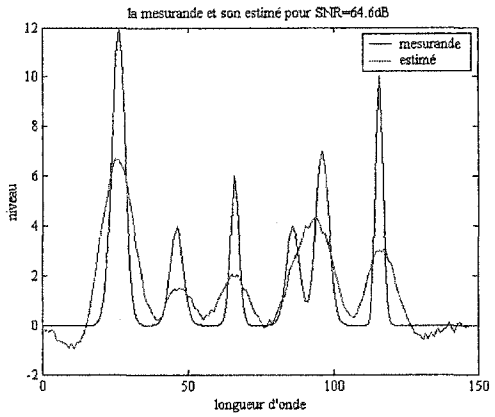


Figure 3.12 : Mesurande et son estimé pour σ=8 et SNR=64.6dB

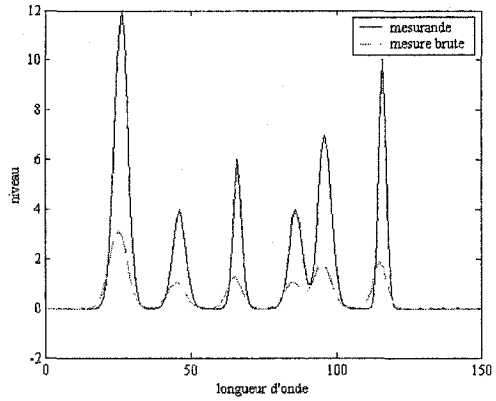


Figure 3.13 : Mesurande et la sortie du système de mesure pour σ=2 et SNR=34.6dB

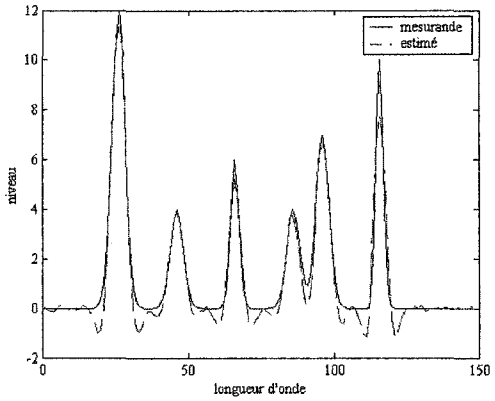


Figure 3.14 : Mesurande et son estimé pour σ=2 et SNR=34.6dB

Tableau 3.3: Moyenne de 50 réalisations de l'erreur relative des moindre carrés obtenu par H∞, Kalman, Gold et Jasson_L suivant le niveau de bruit et la largeur de la bande passante du système de mesure.

		Algorithme			
σ	**SNR (dB)**	**H∞**	**Kalman**	**Gold**	**Jasson_L**
8	**64.6**	0.5000	0.5779	0.3951	0.1017
	54.6	0.5010	0.5785	0.3956	0.1088
	44.6	0.5191	0.5806	0.3957	0.1320
	34.6	0.5436	0.5895	0.3985	0.1544
4	**64.6**	0.3181	0.3196	0.0634	0.0327
	54.6	0.3201	0.3219	0.0637	0.0390
	44.6	0.3280	0.3290	0.0637	0.0545
	34.6	0.3441	0.3439	0.0872	0.0585
2	**64.6**	0.1742	0.1751	0.0060	0.0104
	54.6	0.1743	0.1751	0.0065	0.0108
	44.6	0.1743	0.1753	0.0151	0.0193
	34.6	0.1754	0.1776	0.0670	0.0416

Suivant les résultats précédents, l'algorithme proposé donne de bon résultats et il a satisfait les conditions attendues de l'équipe de recherche au laboratoire de microsystèmes de mesure qui est : un bon résultat d'estimation des positions de pics, rapidité de calcul pour l'utiliser en temps réel (cf. 3.4), valeurs optimums reste presque invariant quelque soit le niveau de bruit. L'algorithme est un nouveau outil qui complémente les autres algorithmes de reconstitution des signaux spectrométriques au le laboratoire est pourra être implémenter sur Silicium pour des traitement en temps réel.

3.3.2 Résultats obtenus pour des données réelles :

On a testé l'algorithme proposé sur des données spectrométriques et de télécommunication réelles et ce pour vérifier la performance du filtre en temps réel traitant des données concrets tout à fait réelles.

La spectrométrie est, à l'heure actuelle, l'outil indispensable de recherche et d'analyse de plusieurs sciences et techniques. Des milliers de spectromètres sont journellement employés, non seulement dans les grands laboratoires chimiques, mais les industries les plus diverses en font usage pour contrôler leurs fabrications. Les exemples les plus classiques des milieux utilisant les spectromètres sont les milieux biologiques en général (médecine, pharmacologie, etc.), l'environnement (mesure et analyse des polluants atmosphériques et des eaux), les industries pétrolières, etc. Les spectres obtenus, grâce à ces spectromètres, sont des outils puissants pour l'identification et l'analyse des substances inconnues, la mesure quantitative des constituants de mélanges complexes et l'étude de leur nombreux paramètres physico-chimiques. Mais, la qualité de mesure de ces spectres dépend fortement du pouvoir de résolution de l'instrument utilisé. En fait, c'est ce paramètre métrologique qui caractérise essentiellement, en terme de coût, un appareil spectrométrique. [1]

L'instrumentation moderne est progressivement confronté à des analyses de plus en plus complexes, liées aux nombres important de constituants présents et aux quantités extrêmement faibles à détecter. La demande aux analyses spectrométriques est devenue

grandissante dans les laboratoires les plus diverses (environnementales, biochimiques, physiques, etc.) et le besoin pour diminuer les coûts des analyses à haute résolution est devenu très important. Également le développement de nouveaux outils d'analyse intégrés et miniaturisés, apte à travailler sur le terrain, est devenu indispensable pour le monitoring en temps réel.

a)- données spectrométriques

Les données spectrométriques sont prélevées à l'aide d'un spectromètre de résolution 4nm. La réponse impulsionnelle du spectromètre $\mathbf{g}(\lambda)$ (figure 3.15) a été identifiée à l'aide des données de références (figure 3.16). Les résultats de reconstitution obtenus avec l'algorithme proposé sont présentés à la figure 3.17 et ils sont comparés avec les mesures prises à partir d'un spectromètre de résolution 0.2nm.

D'autres résultats de reconstitution obtenus avec l'algorithme H∞ sont présentés à la figure 3.18. Ces données sont prélevées à l'aide d'un autre spectromètre de résolution 4nm.

Comme nous pouvons remarquer, le filtre est un bon estimateur pour un mesurande dynamique avec un erreur d'estimation minime. Cette reconstitution a été obtenue sans vraiment optimiser le filtre, en effet seules les paramètres **W** et **V** sont choisit adéquatement pour avoir le meilleur résultat possible à l'étape d'étalonnage.

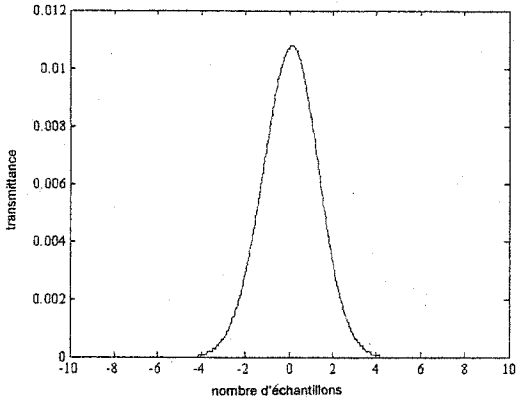


fig. 3.15 : Réponse impulsionnelle du spectromètre de résolution 4nm trouver à l'aide de Gold

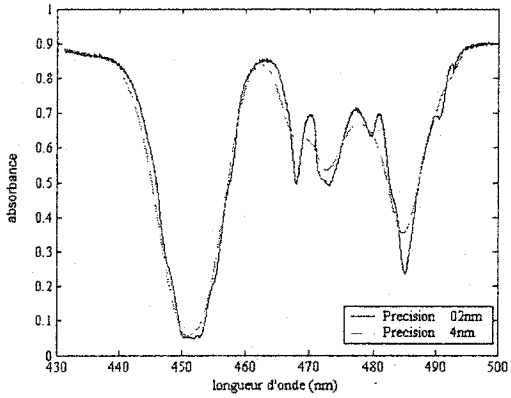


fig. 3.16a : Données d'étalonnage à l'aide de spectromètres de résolutions 0.2nm et 4nm

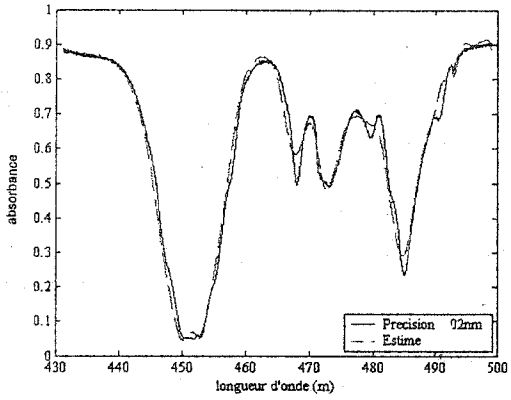


fig. 3.16b : Étalonnage du système : l'estimation et la sortie du spectromètre de résolution 0.2nm

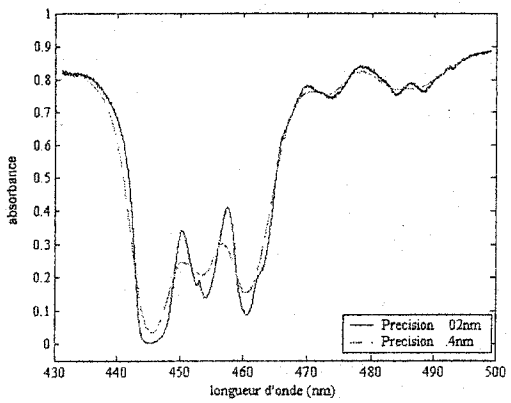


fig. 3.17a : Données de validation à l'aide de spectromètres de résolutions 0.2nm et 4nm

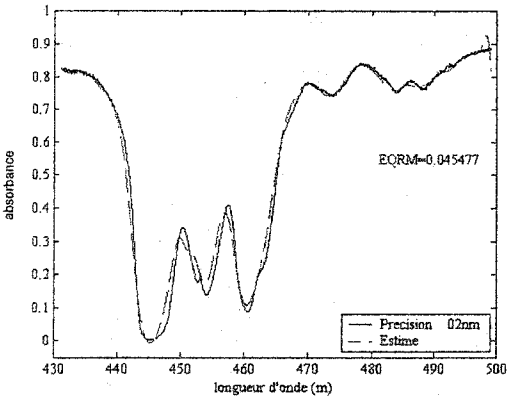


fig. 3.17b : Estimation comparer à la sortie du spectromètre de résolution 0.2nm

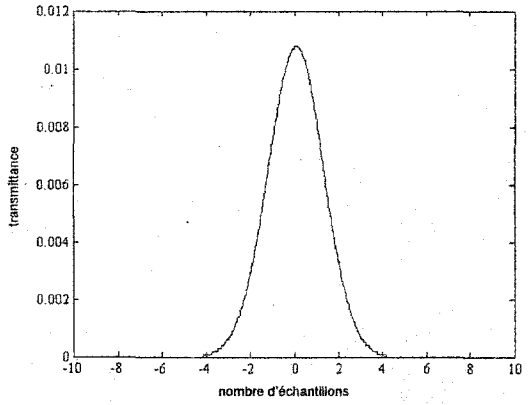


fig. 3.18a : Réponse impulsionnelle du spectromètre de résolution 4nm trouver à l'aide de Gold

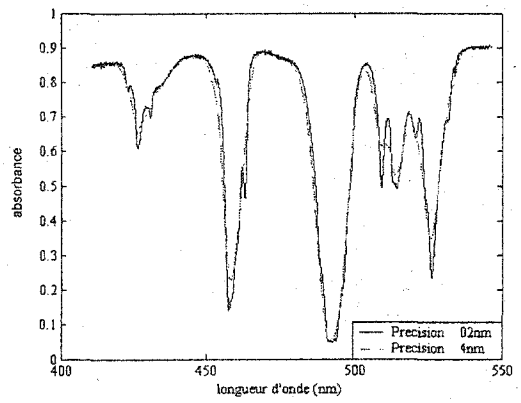


fig. 3.18b : Données d'étalonnage à l'aide de spectromètres de résolutions 0.2nm et 4nm

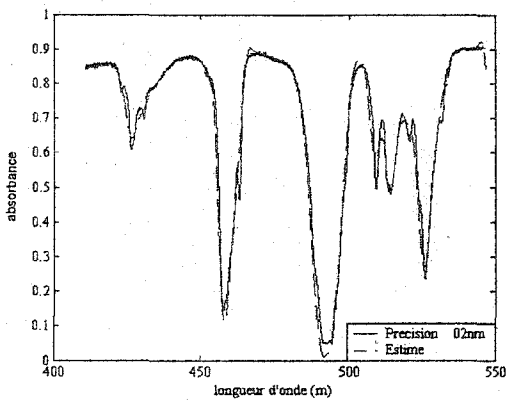


fig. 3.18c : Étalonnage du système : l'estimation et la sortie du spectromètre de résolution 0.2nm

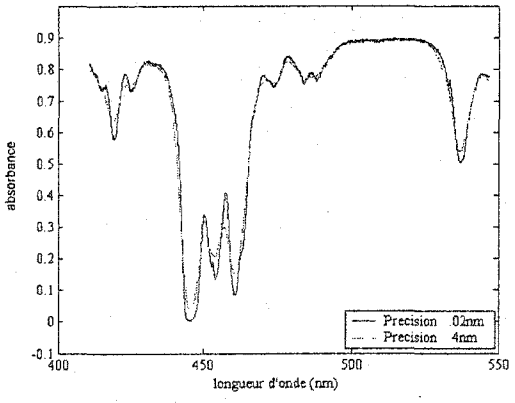


fig. 3.18d : Données de validation à l'aide de spectromètres de résolutions 0.2nm et 4nm

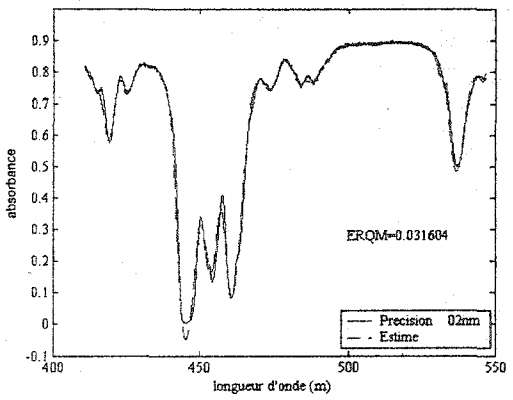


fig. 3.18e : Estimation comparer à la sortie du spectromètre de résolution 0.2nm

Figure 3.18 : Visualisation de la réponse impulsionnelle du système de mesure de résolution 4nm, des données d'étalonnage et les données de validation

b)- données de télécommunication

Les données de télécommunication sont prélevées à l'aide des deux spectromètre de résolution 0.2nm et 0.5nm respectivement. La réponse impulsionnelle du spectromètre $\mathbf{g}(\lambda)$ (figure 3.19 et 3.22) a été identifiée expérimentalement avec des données de références (figure 3.20 et 3.23).

L'étalonnage a été basé sur un jeu de trois prélèvements qui constituent les données de calibration du système. Un autre jeu de trois prélèvement constituent cette fois les données de validation du filtre.

Les résultats de reconstitution obtenus avec l'algorithme en question sont présentés à la figure 3.21 et 3.24 et ils sont comparés à des mesures pris à l'aide d'un spectromètre très précis de résolution 0.06nm.

Ces résultats montre que le filtre proposé apporte des résultats assez bonnes et peut être la première étape de la détermination les positions des pics du spectre et leurs amplitudes. Une partie non négligeable de l'erreur de reconstitution est dû à l'identification du système qui n'était pas très précise.

Notons que la réponse impulsionnelle ne peut être estimée par le produit de convolution de deux séries de fonctions exponentielles décroissants et croissants respectivement, la méthode classique qui mis en jeux cette fois, et on peut même utiliser la transformation en z du $\mathbf{g}(\lambda)$. De plus comme la taille de $\mathbf{g}(\lambda)$ est de 106 ce qui va alourdir le calculateur, alors on a sous-échantillonné ces données à $\frac{dt}{5}$ qui diminuera la taille de $\mathbf{g}(\lambda)$ à 21 éléments.

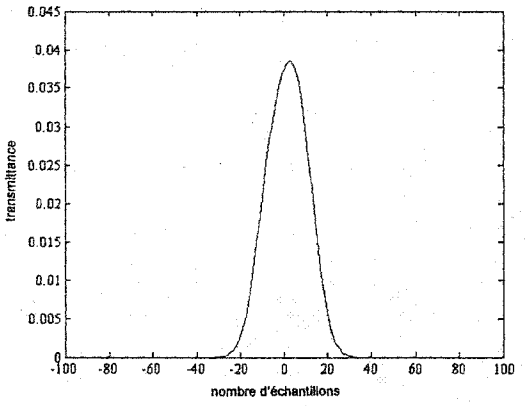


fig 3.19 : Réponse impulsionnelle du spectromètre de résolution 0.2nm trouver à l'aide de Gold

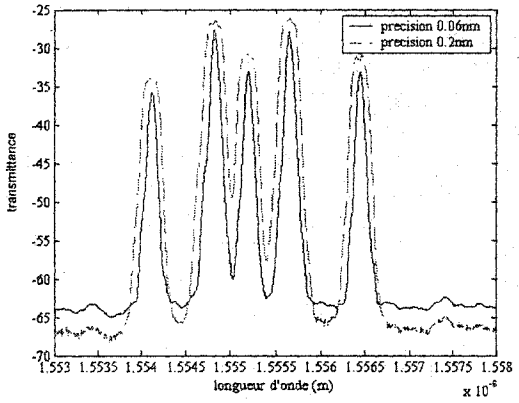


fig 3.20a : Données d'étalonnage à l'aide de spectromètres de résolutions 0.06nm et 0.2nm

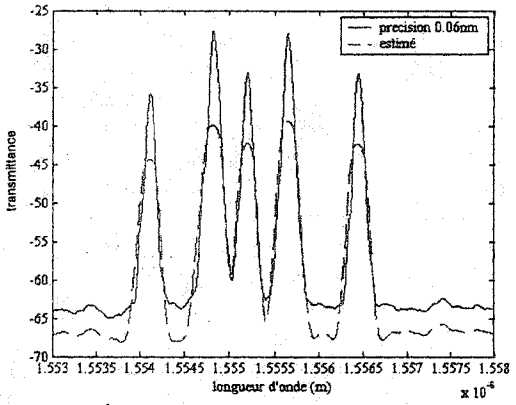


fig 3.20b : Étalonnage du système : l'estimation et la sortie du spectromètre de résolution 0.06nm

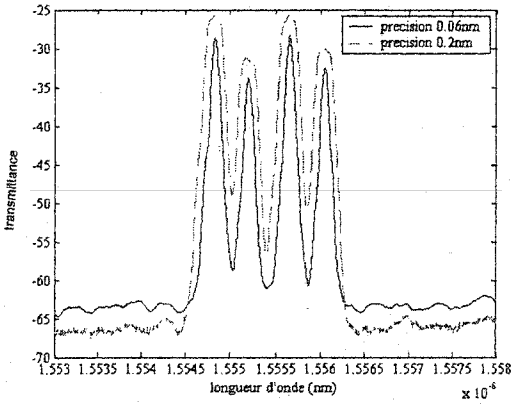


fig. 3.21a- Données de validation à l'aide de spectromètres de résolutions 0.06nm et 0.2nm (prélèvement 1)

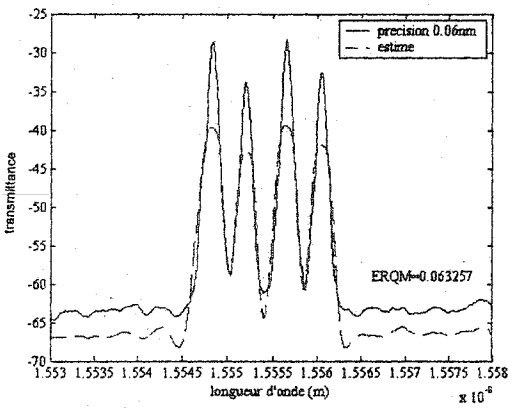


fig. 3.21b- L'estimation comparer à la sortie du spectromètre de résolution 0.06nm (prélèvement 1)

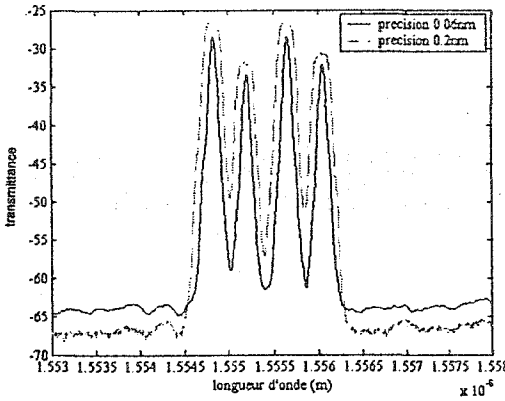


fig. 3.21c- Données de validation à l'aide de spectromètres de résolutions 0.06nm et 0.2nm (prélèvement 2)

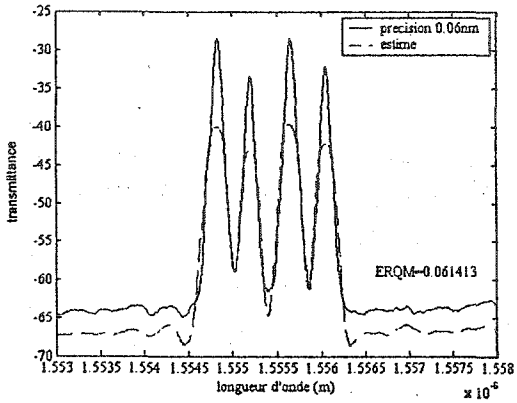


fig. 3.21d- L'estimation comparer à la sortie du spectromètre de résolution 0.06nm (prélèvement 2)

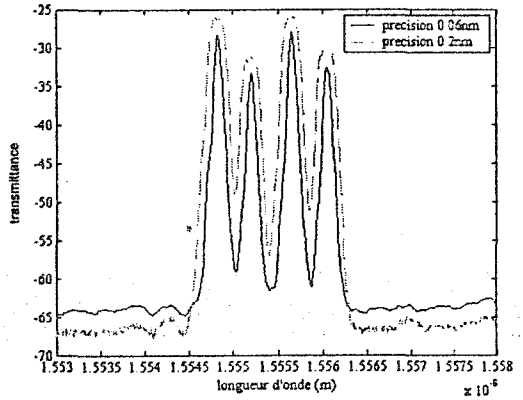


fig. 3.21e- Données de validation à l'aide de spectromètres de résolutions 0.06nm et 0.2nm (prélèvement 3)

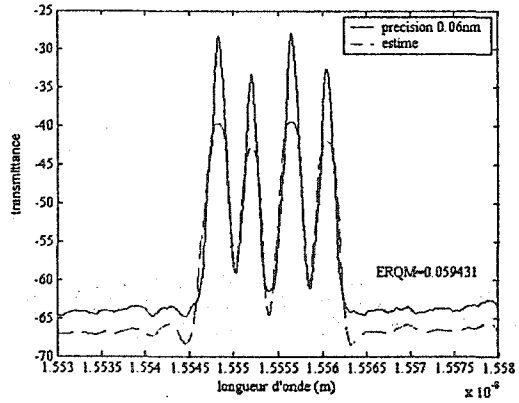


fig. 3.21f- L'estimation comparer à la sortie du spectromètre de résolution 0.06nm (prélèvement 3)

Figure 3.21 : Visualisation de données de validation et comparaison de l'estimé à la mesure donnée par le spectromètre de résolution 0.06nm pour trois prélèvements

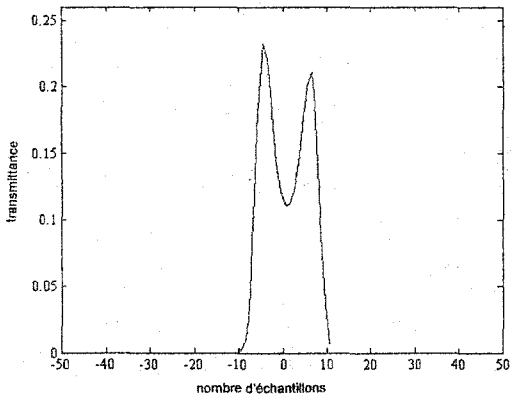


fig 3.22 : Réponse impulsionnelle du spectromètre de résolution 0.5nm trouver à l'aide de Gold

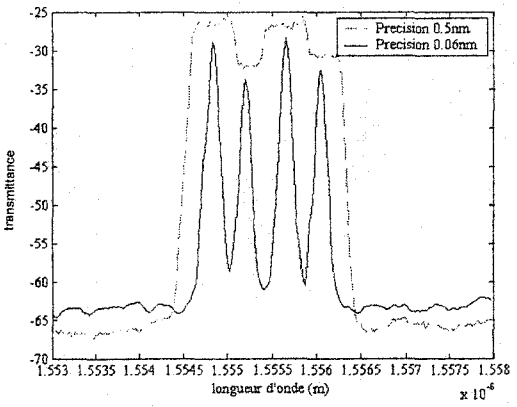


fig 3.23a : Données d'étalonnage à l'aide de spectromètres de résolutions 0.06nm et 0.5nm

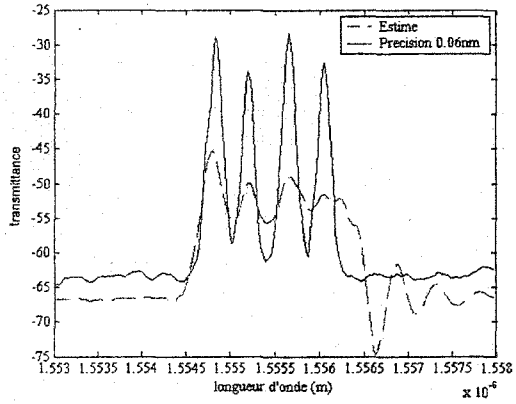


fig 3.23b : Étalonnage du système : l'estimation et la sortie du spectromètre de résolution 0.06nm

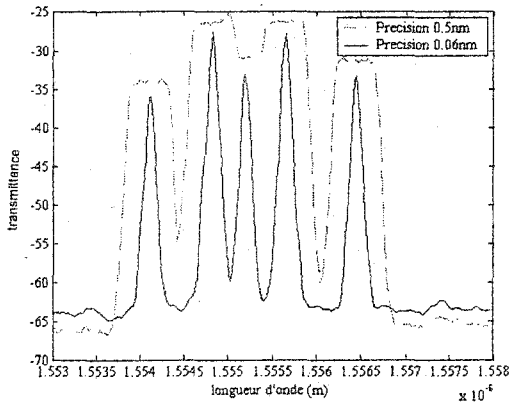


fig. 3.24a- Données de validation à l'aide de spectromètres de résolutions 0.06nm et 0.5nm (prélèvement 1)

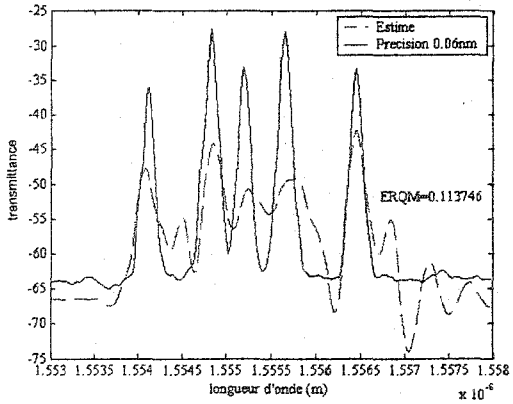


fig. 3.24b- L'estimation comparer à la sortie du spectromètre de résolution 0.06nm (prélèvement 1)

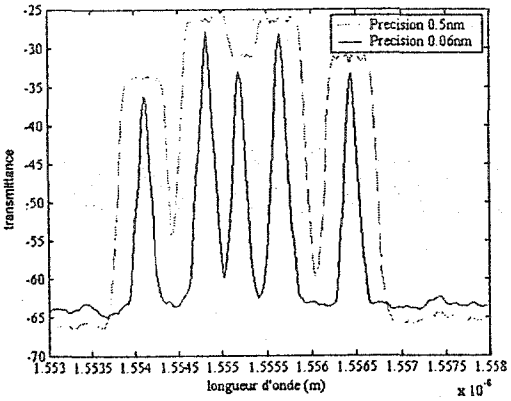


fig. 3.24c- Données de validation à l'aide de spectromètres de résolutions 0.06nm et 0.5nm (prélèvement 2)

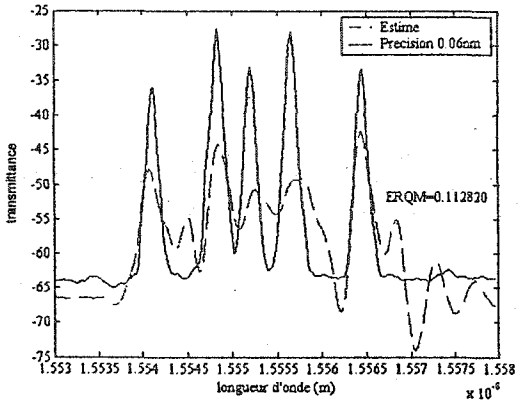


fig. 3.24d- L'estimation comparer à la sortie du spectromètre de résolution 0.06nm (prélèvement 2)

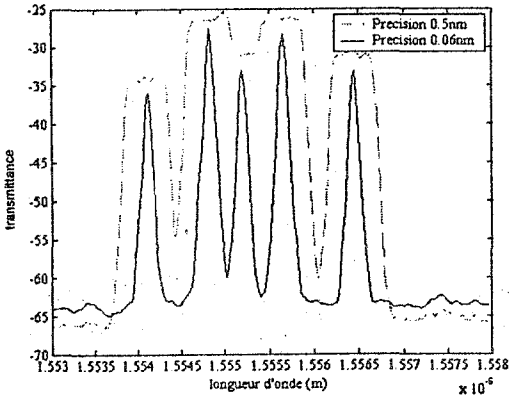


fig. 3.24e- Données de validation à l'aide de spectromètres de résolutions 0.06nm et 0.5nm (prélèvement 3)

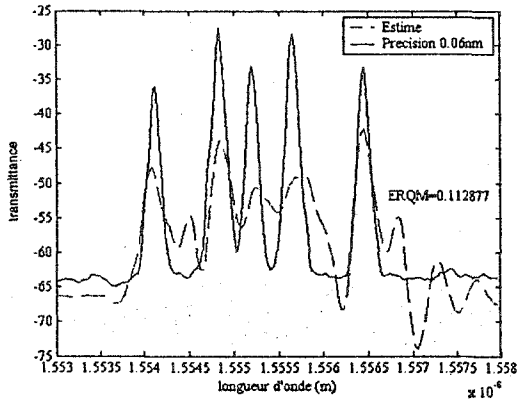


fig. 3.24f- L'estimation comparer à la sortie du spectromètre de résolution 0.06nm (prélèvement 3)

Figure 3.24 : Visualisation de données de validation et comparaison de l'estimé à la mesure donnée par le spectromètre de résolution 0.06nm pour trois prélèvements

3.4 Comparaison de la complexité de calcul entre le filtre H∞ et les algorithmes de référence

Le temps de calcul et la complexité d'un algorithme dépend étroitement du nombre d'opérations arithmétiques requises. De plus, pour un système stationnaire qui est toujours le cas dans la plupart des systèmes de mesure, l'équation de Riccati sera répétées N_k fois jusqu'à ce que le gain du filtre H∞ devient constant. Le tableau (3.4) illustre et compare le nombre d'additions et de multiplications de l'algorithme proposé avec celui des algorithmes de référence.

Les dimensions des matrices de l'équation d'état sont :

dim(A) = 2Mg x 2Mg

dim(B) = 2Mg x 1

dim(C) = 1 x 2Mg

Mg représente le nombre de fonctions exponentielles décroissantes (Eq. 3.7)

Le tableau 3.5 montre le nombre d'opérations arithmétiques pour les signaux synthétiques traitées dans le cas où M = N = 128 et Mg = 3. Le nombre d'itérations k_{iter} pour les algorithmes de Jansson et Van Cittert dépendent du niveau de bruit et peut aller jusqu'à 1000.

Ces tableaux montrent que pour un système de mesure stationnaire, l'algorithme basé sur le filtrage H∞ est beaucoup plus rapide que les algorithmes de référence et surtout si le nombre d'échantillons est plus grand.

De plus, l'algorithme donne des résultats meilleurs que le filtre de Kalman on peut le considérer comme un outil pour les applications temps réel qui exigent la rapidité de traitement et l'exactitude d'estimation.

Tableau 3.4 : Comparaison de la complexité de calcul entre le filtre H∞ et les algorithmes de référence

Algorithme	# +	# x
H∞	$(8Mg^2+4Mg)$ N	$(8Mg^2+8Mg)$ N
Kalman	$16Mg^2$ N	$(16Mg^2+8Mg)$ N
Jansson	(M+1) N k_{iter}	(MN+N-M+1) k_{iter}
Van Cittert	(M+3) N k_{iter}	(MN+3N-M+1) k_{iter}

Tableau 3.5 : Nombre d'opération pour le filtre H∞ et les algorithmes de référence

Algorithme	# +	# ×
H∞	10752	12288
Kalman	18432	21504
Jansson	16512 k_{iter}	16385k_{iter}
Van Cittert	16768 k_{iter}	16641 k_{iter}

D'autre part, le filtre proposé peut être un premier pas pour la détermination exacte des positions des pics ainsi leurs amplitudes en appelant des algorithmes itératifs tel que LMS [8][9]. Sur les éléments du vecteur d'amplitudes on peut même introduire la contrainte de positivité pour éliminer les éléments négatifs de ce dernier.

3.5 Réduction du nombre de paramètres de l'algorithme basé sur le filtre H∞

D'après la définition du filtre, le critère de la sous-optimalité est donné par l'équation (3.21) et (3.22).[3]

$$J(Q_k, p, W_k, V_k) = \frac{\sum_{k=0}^{N-1} \left\| x_k - \hat{x}_k \right\|^2_{Q(k)}}{\left\| z_0 - \hat{z}_0 \right\|^2_{p^{-1}} + \sum_{k=0}^{N-1} \left(\left\| w_k \right\|^2_{W_k^{-1}} + \left\| v_k \right\|^2_{V_k^{-1}} \right)} < 1/\gamma \qquad (3.21)$$

si $\hat{x}_0$ n'est pas connu.

La matrice initiale de **P**(k), solution de l'équation matricielle de Riccati, est **p**.

Ou

$$J(Q_k, W_k, V_k) = \frac{\sum_{k=0}^{N-1} \left\| x_k - \hat{x}_k \right\|^2_{Q(k)}}{\sum_{k=0}^{N-1} \left(\left\| w_k \right\|^2_{W_k^{-1}} + \left\| v_k \right\|^2_{V_k^{-1}} \right)} < 1/\gamma \qquad (3.22)$$

si $\hat{x}_0$ est connu.

La matrice initiale de **P**(k), solution de l'équation matricielle de Riccati, est la matrice zéros. La notation $\|u\|^2_U$ équivaut au triple produits matriciels $u^T.U.u$; et $\|u\|^2$ est la norme H_2 du vecteur **u**.

Si on considère la nouvelle mesure brute $y_1(\lambda) = y(\lambda) - y(0)$ ce qui entraîne à $y_1(0) = 0$, et $\hat{x}(0) = 0$, $\hat{x}_0$ est donc connu à l'avance et ainsi la fonction coût à optimiser

qui sera considérée est celle donnée par (3.22). La matrice d'initiation pour l'équation de Riccati est la matrice zéro. Des simulations ont été effectuées, ont donné des bons résultats. De plus, pour un signal contenant plusieurs piques la matrice zéros est la matrice qui donne la sous optimalité par rapport à p.

D'autre part, dans l'équation de Riccati, la trouve la matrice $\mathbf{Q}$ ($\mathbf{Q} = \overline{\mathbf{Q}}$) est toujours reliée (corrélés) directement avec γ par une multiplication.

Si $\mathbf{Q} = q\,\mathbf{I}$, avec $\mathbf{I}$ désigne la matrice identité avec les dimensions appropriées.

Soit $\gamma_1 = \gamma.\mathbf{Q}$ ou $\gamma_1 = (\gamma.q).\mathbf{I}$ soit aussi $\gamma_1 = \left(\frac{\gamma}{\alpha}\right).(\alpha.q).\mathbf{I}$

Ce qui veut dire que tout changement sur la matrice Q peut être simplifier par γ. On pose donc $\mathbf{Q} = \mathbf{I}$ et $\gamma_1 = \gamma.q$ sera la nouvelle limite pour le critère d'optimisation et l'équation (3.22) s'écrit

$$J(W_k, V_k) = \frac{\sum_{k=0}^{N-1} \left\| x_k - \hat{x}_k \right\|^2}{\sum_{k=0}^{N-1} \left(\left\| w_k \right\|^2_{W_k^{-1}} + \left\| v_k \right\|^2_{V_k^{-1}} \right)} < 1/\gamma \qquad (3.23)$$

en tenant compte de ce qui précède, la fonction coût s'écrit comme suit

$$J(W_k, V_k) = \frac{\sum_{k=0}^{N-1} \left\| x_k - \hat{x}_k \right\|^2}{\sum_{k=0}^{N-1} \left(\left\| w_k \right\|^2_{W_k^{-1}} + \left\| v_k \right\|^2_{V_k^{-1}} \right)} = \frac{\sum_{k=0}^{N-1} \left\| x_k - \hat{x}_k \right\|^2}{\sum_{k=0}^{N-1} \left(w_k^T . W_k^{-1} . w_k + v_k^T . V_k^{-1} . v_k \right)} \qquad (3.24)$$

Comme on a pris les matrices arbitraires comme un produit d'un scalaire et de la matrice identité c'est-à-dire $\mathbf{W} = W \,.\, \mathbf{I}$ et $\mathbf{V} = V \,.\, \mathbf{I}$, W et V sont des scalaires appartenant à $\mathbf{R}^{+*}$ afin que les matrices **W** et **V** soient définies positives.

Donc on a

$$J(W,V) = \frac{\sum_{k=0}^{N-1} \left\| \mathbf{x}_k - \hat{\mathbf{x}}_k \right\|^2}{\sum_{k=0}^{N-1} \left[W^{-1} (w_k^T . \mathbf{I} . w_k) + V^{-1} (v_k^T . \mathbf{I} . v_k) \right]} = \frac{\sum_{k=0}^{N-1} \left\| \mathbf{x}_k - \hat{\mathbf{x}}_k \right\|^2}{\sum_{k=0}^{N-1} \left(W^{-1} \left\| w_k \right\|^2 + V^{-1} \left\| v_k \right\|^2 \right)}$$

$$= \frac{\sum_{k=0}^{N-1} \left\| \mathbf{x}_k - \hat{\mathbf{x}}_k \right\|^2}{W^{-1} \sum_{k=0}^{N-1} \left(\left\| w_k \right\|^2 + \frac{V^{-1}}{W^{-1}} \left\| v_k \right\|^2 \right)} = \frac{\sum_{k=0}^{N-1} \left\| \mathbf{x}_k - \hat{\mathbf{x}}_k \right\|^2}{W^{-1} \sum_{k=0}^{N-1} \left(\left\| w_k \right\|^2 + \beta^{-1} \left\| v_k \right\|^2 \right)}$$

Avec $\beta = \frac{V}{W}$

Le critère d'optimisation est donné ainsi par l'équation ci-dessous

$$J(W,V) = \frac{\sum_{k=0}^{N-1} \left\| \mathbf{x}_k - \hat{\mathbf{x}}_k \right\|^2}{W^{-1} \sum_{k=0}^{N-1} \left(\left\| w_k \right\|^2 + \beta^{-1} \left\| v_k \right\|^2 \right)} < 1/\gamma$$

d'où

$$J(\beta) = \frac{\sum_{k=0}^{N-1} \left\| \mathbf{x}_k - \hat{\mathbf{x}}_k \right\|^2}{\sum_{k=0}^{N-1} \left(\left\| w_k \right\|^2 + \beta^{-1} \left\| v_k \right\|^2 \right)} < 1/\gamma \qquad (3.25)$$

avec $\bar{\gamma} = \gamma . W$ la nouvelle limite d'optimisation.

Ainsi, d'après ce qui précède, on a pu réduire aisément le nombre de paramètres arbitraires de l'algorithme proposé de 4 paramètres (matrices **Q**, **p**, W, V) à un seul paramètre scalaire β hormis γ.

Notons que la réponse impulsionnelle du système de mesure considéré est un gaussoïde de bande passante caractérisé par σ=4 (fig. 3.25).

Des simulations pratiques sur des données synthétiques ont été procédées pour valider notre démonstration théorique. Les simulations sont effectuées sur un signal à plusieurs piques (fig. 3.26), celui-ci est déjà utilisé dans les simulations avec la précédente méthode d'optimisation qui tient compte des matrices **Q**, **p**, W, V comme paramètres d'optimisation. (voir section 3.2)

Le résultat de simulation pour SNR=64.6dB est montré sur la figure 3.27 pour β optimum. La figure 3.28 montre la variation de l'erreur quadratique moyen relatif en fonction de β. Les erreurs quadratiques moyens relatifs à la sous-optimalité pour l'ancienne et la nouvelle méthode sont les même à 10^{-4} près. De plus, la variance des erreurs est très petite par rapport aux autres méthodes ce qui montre la robustesse de l'algorithme. Ceci, nous dispense de chercher les paramètres optimums à chaque fois et une seule optimisation suffit à long terme quelque soit le niveau de bruit.

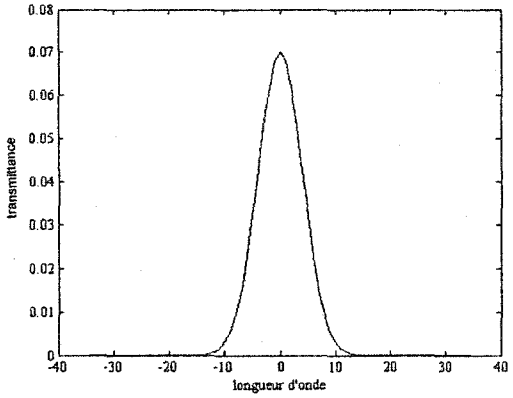


Figure 3.25 : Réponse impulsionnelle du système de mesure

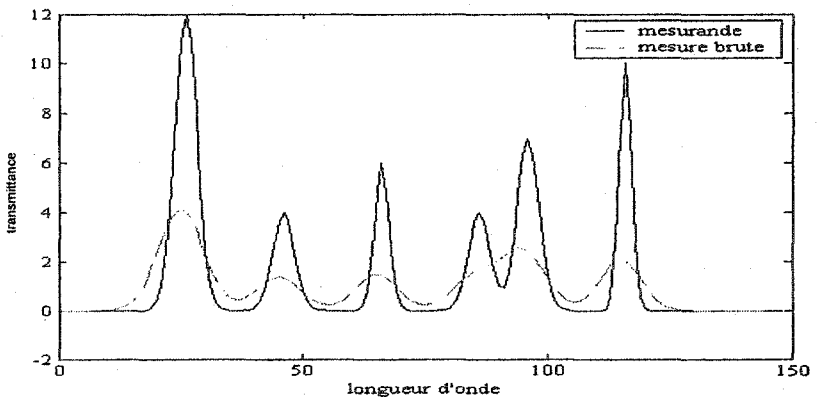


Figure 3.26 : Mesurande et la sortie du système de mesure pour SNR=64.6dB

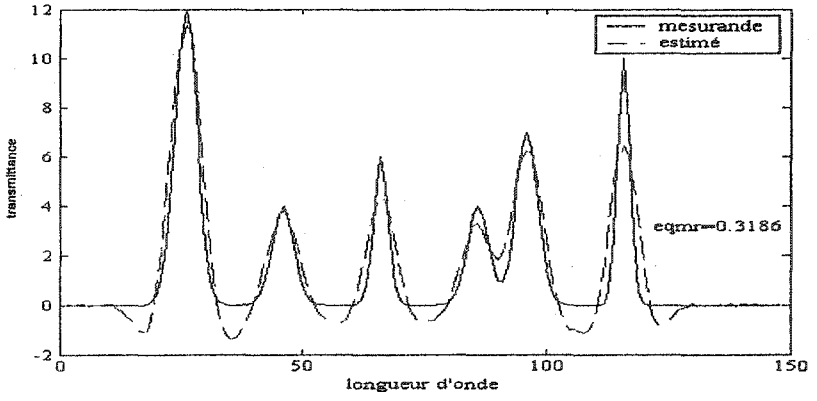


Figure 3.27 : Mesurande et son estimé pour SNR=64.6dB et β=0.0033

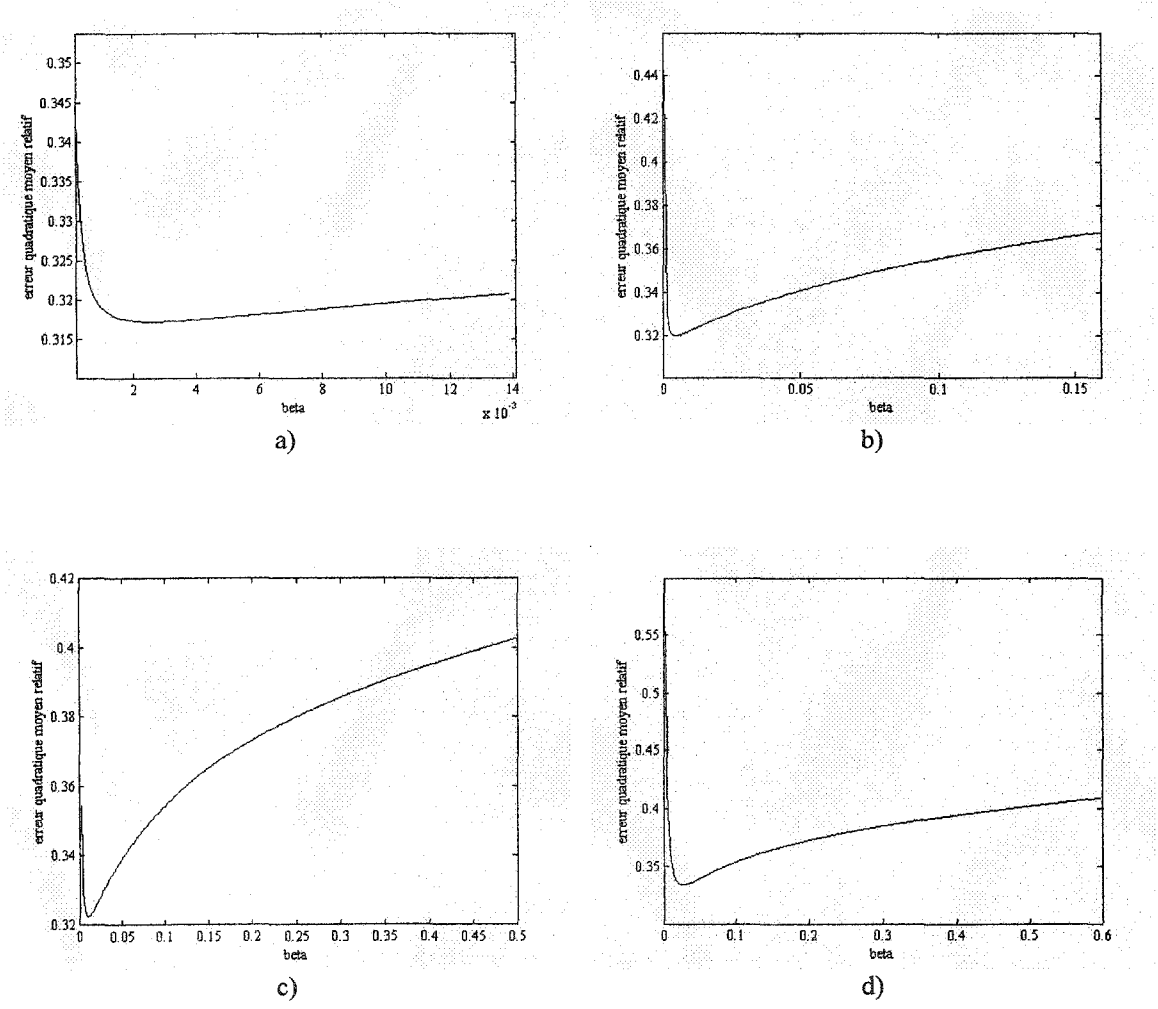


Figure 3.28 : Variation de l'erreur quadratique moyen relatif en fonction du paramètre β pour a) SNR=64.6dB b) SNR=54.6dB c) SNR=44.6dB d) SNR=34.6dB

Le tableau 3.6 compare le rapport $\frac{V_{opt}}{W_{opt}}$ de l'ancienne méthode et celui du paramètre β_{opt} pour la nouvelle méthode. La notation 'opt' veut dire le paramètre qui donne une erreur minimum d'estimation. On remarque que ces valeurs sont très proches, ce qui valide encore une fois la nouvelle méthode d'optimisation à paramètres réduits.

De plus, l'équation de Riccati connaîtra une diminution du nombre d'opérations arithmétiques.

Tableau 3.6 : Comparaison de différents valeurs $\frac{V_{opt}}{W_{opt}}$ de l'ancienne méthode et β_{opt} de la nouvelle méthode.

SNR (dB)	**64.6**	**54.6**	**44.6**	**34.6**
$\frac{V_{opt}}{W_{opt}}$	0.0035	0.0041	0.015	0.039
β_{opt}	0.0033	0.0048	0.011	0.037

Reformulant l'énoncé du filtre H∞ en tenant compte des ces nouveaux résultats :

Soit un système représenté par le modèle d'état suivant:

$$\begin{aligned} \mathbf{z}(k+1) &= \mathbf{A}(k)\mathbf{z}(k) + \mathbf{B}(k)\mathbf{w}(k) \\ \mathbf{y}(k) &= \mathbf{C}(k)\mathbf{z}(k) + \mathbf{v}(k) \end{aligned} \qquad k=0,1,\ldots. N-1, \quad \mathbf{z}(0)=\mathbf{z}_0 \qquad (3.26)$$

On s'intéresse à estimé une combinaison linéaire $\mathbf{x}(k)$ des vecteurs d'état donné tel que $\mathbf{x}(k)=\mathbf{L}(k).\mathbf{z}(k)=\mathbf{u}(k)$ ($\mathbf{L}$ est pris comme matrice identité). On suppose aussi que les bruits agissant sur la sortie et l'état du système de mesure v et w ont une énergie finie sur l'intervalle [0 , N-1]. La fonction coût est ainsi donnée par :

$$J(\beta) = \frac{\sum_{k=0}^{N-1} \left\| \mathbf{x}_k - \hat{\mathbf{x}}_k \right\|^2}{\sum_{k=0}^{N-1} \left(\left\| w_k \right\|^2 + \beta^{-1} \left\| v_k \right\|^2 \right)} \tag{3.27}$$

L'estimateur sous optimal doit satisfaire $J(\beta) < 1/\gamma$ Avec $\gamma>0$. Le modèle de l'estimateur est donné par l'équation d'état suivante

$$\begin{aligned} \hat{\mathbf{z}}(k+1) &= \mathbf{A}(k)\hat{\mathbf{z}}(k) + \mathbf{K}(k)(y(k) - \mathbf{C}(k)\hat{\mathbf{z}}(k)) \\ \hat{\mathbf{x}}(k) &= \mathbf{L}(k)\hat{\mathbf{z}}(k) \end{aligned} \tag{3.28}$$

Avec,

$$\mathbf{K}(k+1) = \beta^{-1}\mathbf{A}(k)\mathbf{P}(k)(\mathbf{I} - \gamma\mathbf{P}(k) + \beta^{-1}\mathbf{C}^T(k)\mathbf{C}(k)\mathbf{P}(k))^{-1}\mathbf{C}^T(k) \tag{3.29}$$

$$\begin{aligned} \mathbf{P}(k+1) = {} & \mathbf{A}(k)\mathbf{P}(k)\mathbf{A}^T(k) + \mathbf{B}(k)\mathbf{B}^T(k) + \\ & \mathbf{A}(k)\mathbf{P}(k)(\gamma - \beta^{-1}\mathbf{C}^T(k)\mathbf{C}(k))\left[\mathbf{I} - \mathbf{P}(k)(\gamma - \beta^{-1}\mathbf{C}^T(k)\mathbf{C}(k))\right]^{-1}\mathbf{P}(k)\mathbf{A}^T(k) \end{aligned} \tag{3.30}$$

Avec condition initiale P(0) = 0

3.6 Ajout de contrainte de positivité à l'algorithme proposé

On peut toujours améliorer la qualité de reconstitution on ajoutant d'autres contraintes à l'algorithme proposé, par exemple on a ajouté la contrainte de positivité qui se résume, après modification de l'équation d'état (3.28), en ce qui suit.

$$\hat{\mathbf{z}}(k+1) = \mathbf{A}(k)\hat{\mathbf{z}}(k) + \mathbf{K}(k)(y(k) - \mathbf{C}(k)\hat{\mathbf{z}}(k))$$
$$\text{si} \quad \hat{\mathbf{z}}(k) < 0 \quad \text{alors} \quad \hat{\mathbf{x}}(k) = \mathbf{L}(k)\hat{\mathbf{z}}(k) \tag{3.31}$$
$$\text{sinon} \quad \hat{\mathbf{x}} = 0$$

Le tableau 3.7 contient la moyenne de 50 réalisation de l'erreur moyen relative des moindre carrée suivant 4 niveaux de bruits sous les conditions de 3.3.1.2 avec σ=4, le tableau montre le gain de l'ajout de la contrainte de positivité.

Tableau 3.7 : Moyenne de 50 réalisations de l'erreur relative des moindres carrés obtenu par H∞ et H∞ avec contrainte de positivité (Hp∞).

Algorithme	**SNR (dB)**			
	64.6	**54.6**	**44.6**	**34.6**
Hp∞	0.2722	0.2721	0.2754	0.2990
H∞	0.3181	0.3201	0.3280	0.3441

La figure (3.29) montre les résultats de corrections obtenues par l'algorithme proposé pour SNR=38.6dB

On remarque que celui modifié est plus précis et on diminuer l'erreur d'estimation de 14.5% même si on aura une légère augmentation du temps de calcul.

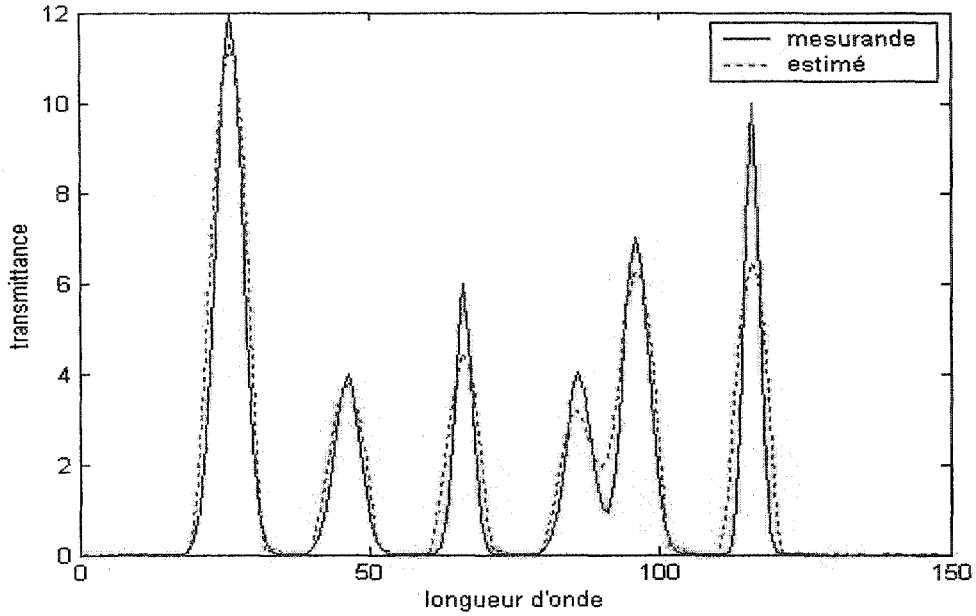


Figure 3.29 : Mesurande et son estimé pour SNR=64.6dB

Chapitre 4

SIMULATION DE L'ALGORITHME DANS UN SIMULATEUR DE PROCESSEUR À USAGE GÉNÉRALE ET PROPOSITION D'ARCHITECTURES ITGE SPÉCIALISÉES

4.1 Simulation de l'algorithme proposé dans un simulateur de processeur numérique à usage générale

L'implantation de l'algorithme proposé dans un processeur numérique de signaux DSP à usage générale constitue une étape fondamentale dans la réalisation de l'opération de reconstitution de mesurandes d'une part, et d'autre part elle permet d'analyser la faisabilité d'une architecture pour l'implantation en VLSI sous forme de FPGA, ASIC etc..., parmi ces points : la dynamique des signaux, l'effet de quantification et la possibilité de rééchelonnement des signaux, la vitesse de reconstitution ...

Le processeur numérique choisi est le DSP56309 de Motorola, il est basé sur une architecture Havard hautement parallèle et pipeliné offrant 255 MIPS et utilisant des mots

de 24 bits (bruit d'arrondi de 144dB). Une telle architecture désignée pour maximiser la capacité intensive en données pour les applications du DSP, notons que ce DSP fonctionne avec une arithmétique à point fixe. La configuration interne du DSP 563× est duale de nature, c'est qu'il y a deux espaces mémoires de données extensibles, deux unités arithmétiques et de génération d'adresses, une unité arithmétique et logique ayant deux accumulateurs, deux circuits décaleurs-limiteurs et un multiplicateur câblé.

Les organigrammes ci-dessous montre les différentes opérations/étapes du programme assembleur pour l'implantation de l'algorithme proposé (Fig. 4.1a et 4.1b). Ce programme s'exécute deux fois après un pivot du résultat intermédiaire. Le programme assembleur (voir annexe) a été assemblé et testé sur le simulateur de DSP56309 de Motorola. Notons bien que les données et les signaux ont été normalisées pour avoir une dynamique inclue dans l'intervalle [-1, 1] (voir 4.2).

Pour un nombre d'échantillons égale à 64, le nombre de cycles machine trouvé est 1.314.718 cycles en totalité. Avec un cycle machine de durée 10ns, le temps total de reconstitution est estimé à environ 13ms. De plus, la figure 4.2 montre le résultat de correction obtenue par l'algorithme implanté sur le simulateur de DSP en question.

4.2 Effet la quantification sur l'algorithme et échelonnement des signaux:

Un des points les plus importants à discuter avant d'aborder la conception du circuit intégré est l'étude de la quantification. Cette dernière permet de conclure sur le type de

représentation, le nombre de bits nécessaire pour représenter les données et la taille des composantes arithmétiques telle que le multiplieur accumulateur.

Une étude de quantification a été effectuée pour déterminer la largeur des mots nécessaire pour représenter les données et les blocs de calcul et ainsi assurer une bonne précision utilisant un calcul à virgule fixe et une représentation en complément à deux. Nous avons réalisé cette étude avec le logiciel Matlab, qui nous permet, dans un premier lieu, de simuler la fonctionnalité de l'algorithme en effectuant les calculs à virgule flottante. Par la suite nous avons modifié les programmes utilisés dans la première étape pour quantifier les signaux. Ainsi nous avons rajouté un programme de quantification de matrice et un autre pour la quantification des vecteurs. Ensuite nous avons réalisé un autre programme qui fait la multiplication matrice-matrice élément par élément et la quantification du résultat après chaque opération arithmétique élémentaire. Comme dernière étape nous avons réalisé un autre programme pour le calcul de l'erreur relative de quantification. Cette dernière est définie comme étant la différence entre le signal estimé avec un certain nombre de bits et le signal original (mesurande).

Tous les programmes utilisés durant cette étude sont présentés en annexe. La qualité de la reconstitution est évaluée selon l'erreur quadratique moyenne relative entre l'estimé et le signal à mesurer :

$$\varepsilon\left(\overset{o}{\mathbf{x}}, \hat{\mathbf{x}}_q\right) = \frac{\left\| \{\hat{\mathbf{x}}_{q,k}\} - \{\overset{o}{\mathbf{x}}_k\} \right\|_2}{\left\| \{\overset{o}{\mathbf{x}}_k\} \right\|_2} \tag{4.1}$$

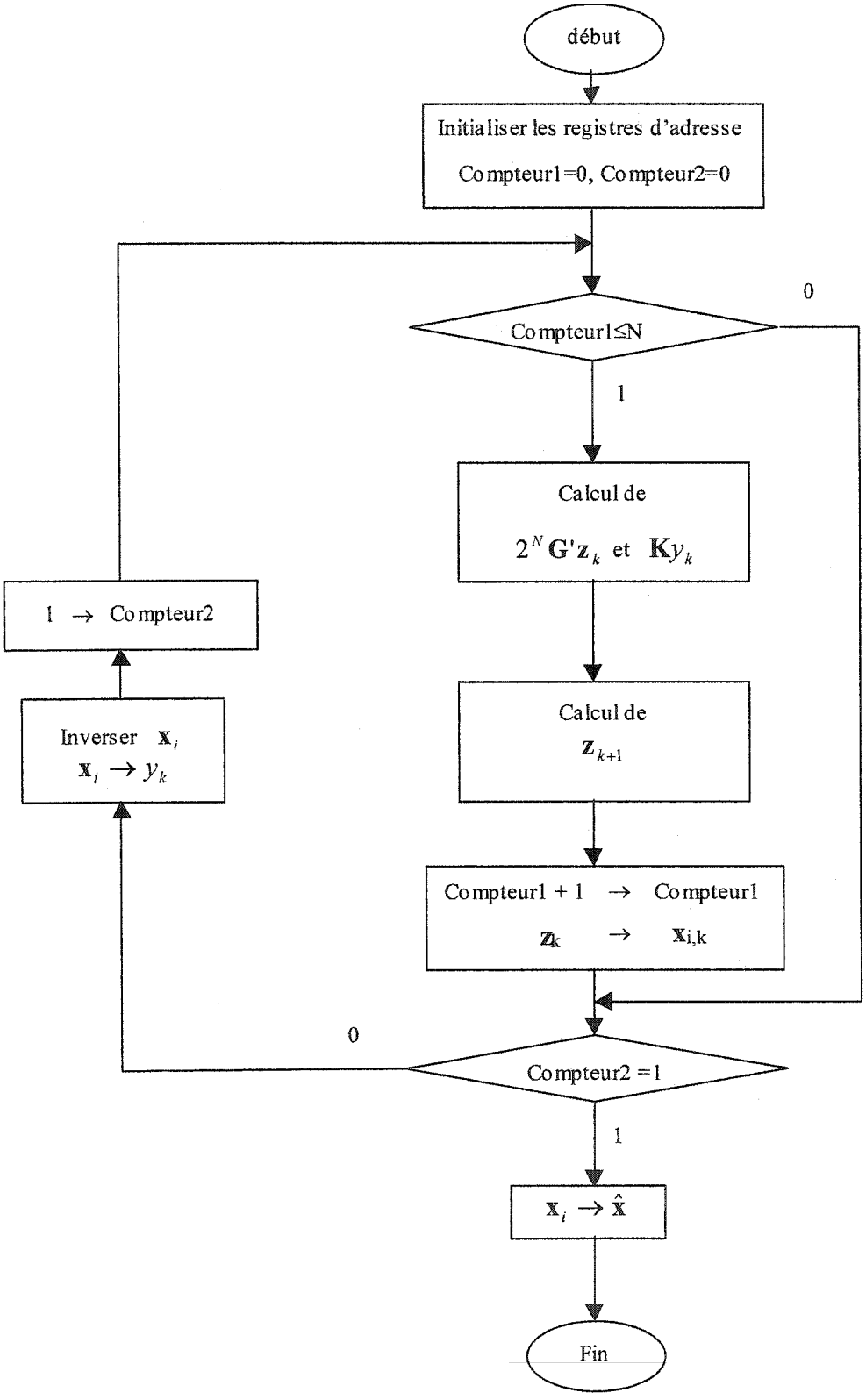


Figure 4.1a : Organigramme simplifié du programme assembleur de reconstitution

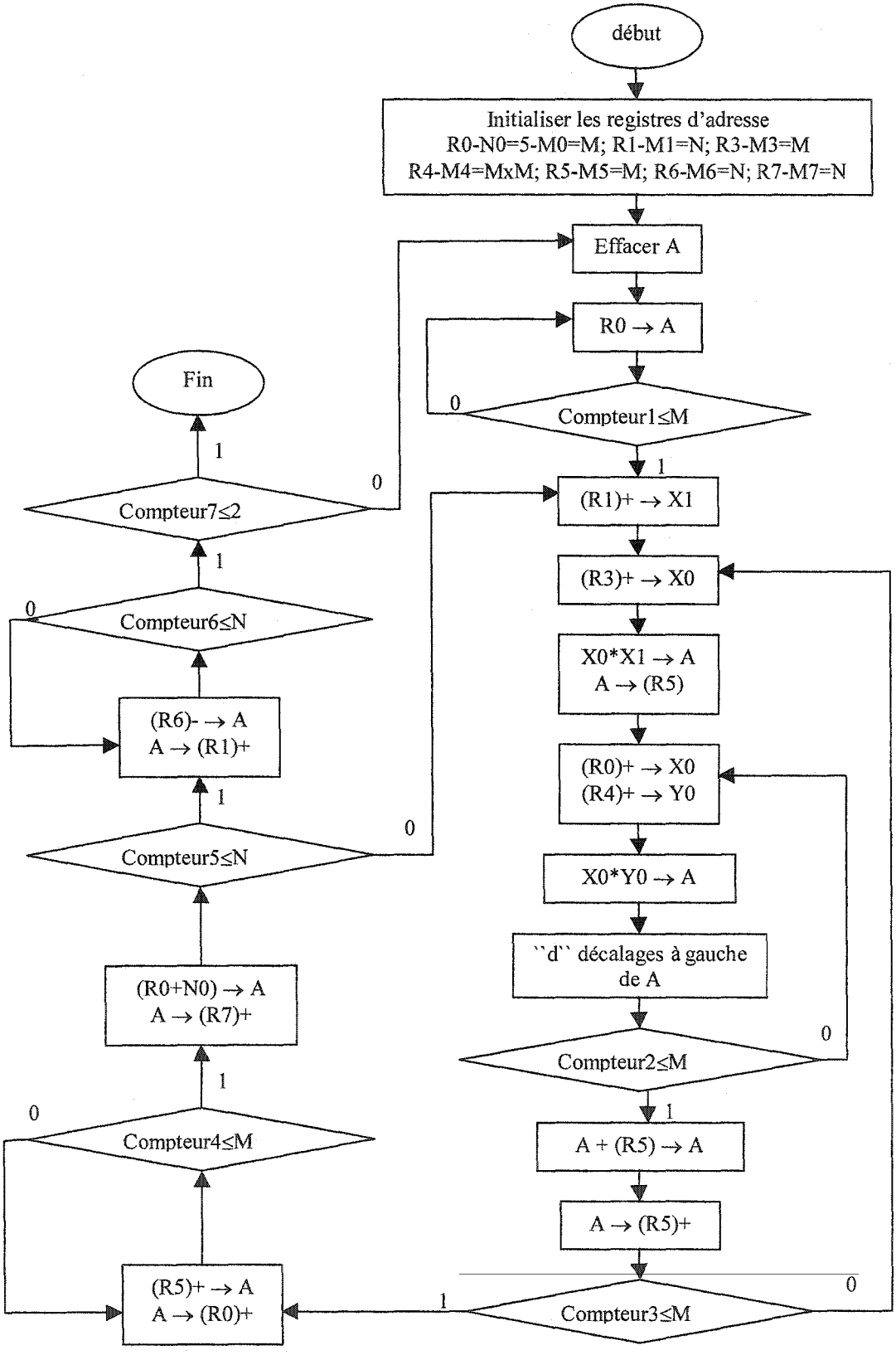


Figure 4.1b : Organigramme détaillé du programme assembleur de reconstitution

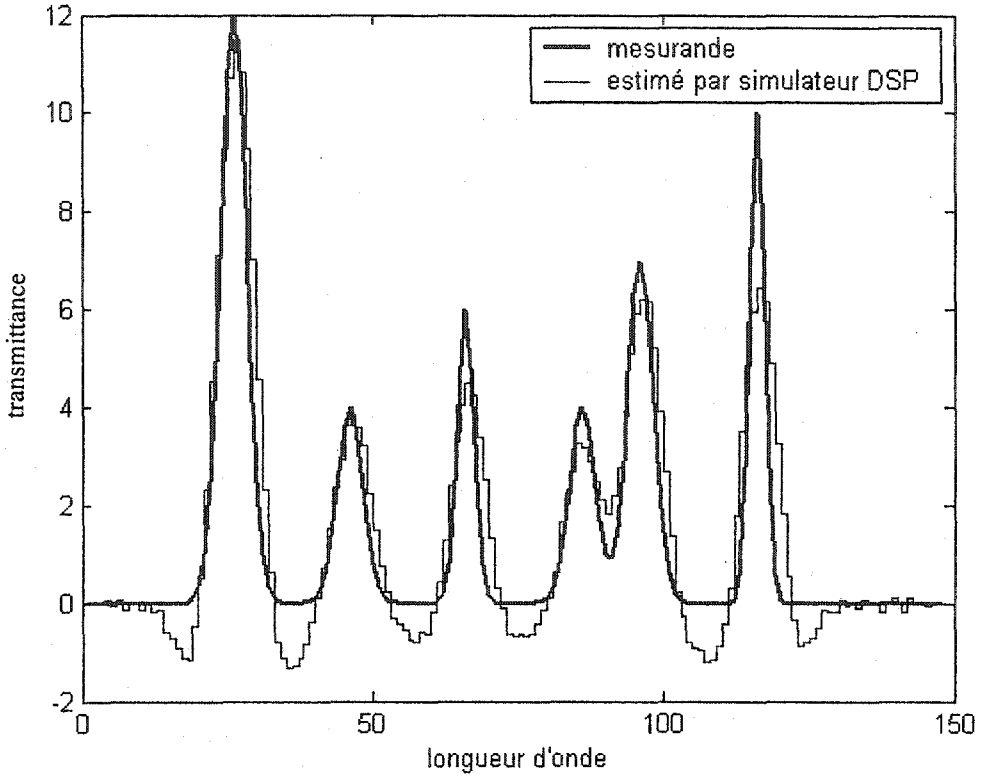


Figure 4.2 : Signal à mesurer et son estimé par l'algorithme tourné par le simulateur DSP56309, M=64 ; N=148, $\Delta\lambda$=1 et SNR=64.6dB

Notons bien que l'arithmétique utilisée est à complément à deux avec saturation. Les données sont échelonnée pour qu'ils soient tous entre les valeurs –1 et 1, pour cela on a procédé au changement de base pour la représentation d'état pour que tous les élément du vecteur **K** satisfait (4.2) et que tous les composants du vecteur d'état satisfait (4.3).

$$\left|\mathbf{K}_{ij}\right| \leq 1 \tag{4.2}$$

$$\left|\mathbf{z}_{i}\right| \leq 1 \tag{4.3}$$

G s'écrira comme produit de 2^N avec une matrice **G'** tel que les éléments de cette dernière satisfait la condition (4.4), ceci économise le temps de calcul en substituant la multiplication par un décalage adéquat.

$$\left|\mathbf{G'}_{ij}\right| \leq 1 \tag{4.4}$$

En effet, soit la représentation d'état de notre système de mesure de dimension 2Mg :

$$\begin{aligned} z(k+1) &= Az(k) + Bw(k) \\ y(k) &= Cz(k) + v(k) \end{aligned} \tag{4.5}$$

Il existe au moins une matrice de transformation **W** qui permet d'obtenir (4.5) sous forme diagonale ($\mathbf{W}_d$) sous les conditions (4.2) et (4.3).

La nouvelle représentation d'état du même système est la suivante :

$$\begin{aligned} \tilde{z}(k+1) &= \tilde{A}\tilde{z}(k) + \tilde{B}w(k) \\ y(k) &= \tilde{C}\tilde{z}(k) + v(k) \end{aligned} \tag{4.6}$$

Où

$$\tilde{\mathbf{A}} = \mathbf{W}^{-1}\mathbf{A}\mathbf{W}, \quad \tilde{\mathbf{B}} = \mathbf{W}^{-1}\mathbf{B}, \quad \tilde{\mathbf{C}} = \mathbf{C}\mathbf{W}. \tag{4.7}$$

Dans ce qui suit, nous écrirons la matrice $\mathbf{W}$ décomposée selon ses différentes colonnes, soit,

$$\mathbf{W} = \left[\mathbf{w}^{(1)} \quad \mathbf{w}^{(2)} \quad \ldots \quad \mathbf{w}^{(2Mg)}\right]$$

Soit λ_i les racines de l'équation caractéristique de $\mathbf{A}$ du système considéré observable, on cherche les matrices $\mathbf{W}_d$ qui permet d'obtenir la nouvelle représentation sous les conditions (4.2) et (4.3) tel que :

$$\tilde{\mathbf{A}} = \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & & & \\ 0 & 0 & \cdots & \lambda_{2Mg} \end{bmatrix}, \quad \tilde{\mathbf{B}} = \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_{2Mg} \end{bmatrix}, \quad \tilde{\mathbf{C}} = \begin{bmatrix} 1 & 1 & \cdots & 1 \end{bmatrix}$$

Explicitons $\tilde{\mathbf{C}} = \mathbf{C}\mathbf{W}_d$ soit $\begin{bmatrix} 1 & 1 & \cdots & 1 \end{bmatrix} = \mathbf{C}.\left[\mathbf{w}^{(1)} \quad \mathbf{w}^{(2)} \quad \ldots \quad \mathbf{w}^{(2Mg)}\right]$ c'est-à-dire

$$1 = \mathbf{C}.\mathbf{w}^{(i)} \quad \text{pour} \quad i = 1, 2, \ldots, 2Mg. \tag{4.7}$$

Explicitons $\mathbf{W}_d\tilde{\mathbf{A}} = \mathbf{A}\mathbf{W}_d$

$$\left[\mathbf{w}^{(1)} \quad \mathbf{w}^{(2)} \quad \ldots \quad \mathbf{w}^{(2Mg)}\right]\begin{bmatrix} \lambda_1 & 0 & \ldots & 0 \\ 0 & \lambda_2 & \ldots & 0 \\ \vdots & & & \\ 0 & 0 & \ldots & \lambda_{2Mg} \end{bmatrix} = \mathbf{A}\left[\mathbf{w}^{(1)} \quad \mathbf{w}^{(2)} \quad \ldots \quad \mathbf{w}^{(2Mg)}\right]$$

Alors $$\mathbf{w}^{(i)}\lambda_i = \mathbf{A}\mathbf{w}^{(i)} \quad \text{pour} \quad i = 1, 2, \ldots, 2Mg \tag{4.8}$$

On peut écrire :

$$\begin{cases} \mathbf{C}\mathbf{w}^{(i)}\lambda_i & = & \mathbf{C}\mathbf{A}\mathbf{w}^{(i)} \\ \mathbf{C}\mathbf{A}\mathbf{w}^{(i)}\lambda_i & = & \mathbf{C}\mathbf{A}^2\mathbf{w}^{(i)} \\ \vdots & & \\ \mathbf{C}\mathbf{A}^{2Mg-2}\mathbf{w}^{(i)}\lambda_i & = & \mathbf{C}\mathbf{A}^{2Mg-1}\mathbf{w}^{(i)} \end{cases} \tag{4.9}$$

D'après (4.7), la première ligne de (4.9) s'écrit :

$$\lambda_i = \mathbf{C}\mathbf{A}\mathbf{w}^{(i)}$$

la deuxième :

$$\lambda_i^2 = \mathbf{C}\mathbf{A}^2\mathbf{w}^{(i)}$$

$$\vdots$$

la dernière ligne :

$$\lambda_i^{2Mg-1} = \mathbf{C}\mathbf{A}^{2Mg-1}\mathbf{w}^{(i)}$$

Et sous forme matricielle peut s'écrire :

$$\overbrace{\begin{bmatrix} 1 & 1 & \cdots & 1 \\ \lambda_1 & \lambda_2 & \cdots & \lambda_{2Mg} \\ \lambda_1^2 & \lambda_2^2 & \cdots & \lambda_{2Mg}^2 \\ \vdots & \vdots & & \vdots \\ \lambda_1^{2Mg-1} & \lambda_2^{2Mg-1} & & \lambda_{2Mg}^{2Mg-1} \end{bmatrix}}^{\tilde{\mathbf{W}}_{obs}} = \overbrace{\begin{bmatrix} \mathbf{C} \\ \mathbf{C}\mathbf{A} \\ \mathbf{C}\mathbf{A}^2 \\ \vdots \\ \mathbf{C}\mathbf{A}^{2Mg-1} \end{bmatrix}}^{\mathbf{W}_{obs}} \overbrace{\cdot \begin{bmatrix} \mathbf{w}^{(1)} & \mathbf{w}^{(2)} & \cdots & \mathbf{w}^{(2Mg)} \end{bmatrix}}^{\mathbf{W}_d} \tag{4.10}$$

On reconnaît la matrice d'observabilité $\mathbf{W}_{obs}$ du système initial. $\tilde{\mathbf{W}}_{obs}$ sera la matrice d'observabilité du système transformé. Ainsi :

$$\mathbf{W}_d = \mathbf{W}_{obs}^{-1} \cdot \tilde{\mathbf{W}}_{obs} \tag{4.11}$$

La détermination de la matrice $\mathbf{W}_d$ a été fait à l'aide d'un programme Matlab se trouvant en annexe.

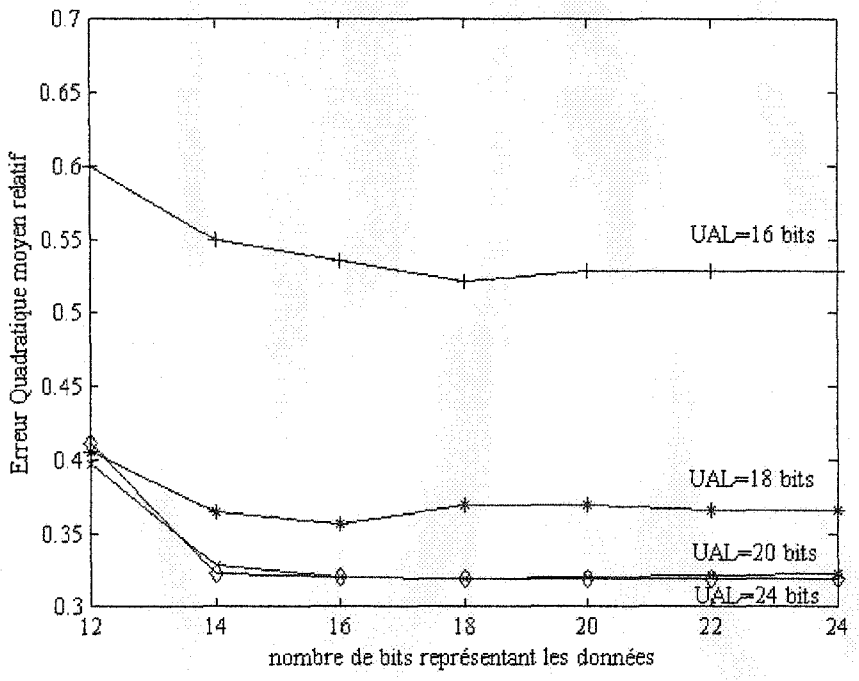
Les figures 4.3-a, 4.3-b, 4.3-c et 4.3-d donnent les résultats de quantification exprimés en terme d'erreur quadratique relative moyenne pour différents SNR. Ces résultats sont obtenus pour différents nombres de bits utilisés pour les blocs de calcul (UAL) et différents nombre de bits utilisés pour la représentation des données soit la matrice $\mathbf{G} = \mathbf{A} - \mathbf{KC}$, le vecteur de gain du filtre K, la mesure brute $\{\tilde{\mathbf{y}}_k\}$ et les variables d'état $\{\mathbf{z}_i\}$.

La figure 4.4-a et 4.4-b montrent les résultats d'estimation d'un signal de 6 piques pour deux niveaux de SNR. Le nombre de bits utilisés pour les blocs de calcul est 20 bits, le nombre de bits utilisés pour la représentation des données est 12 et 14 respectivement.

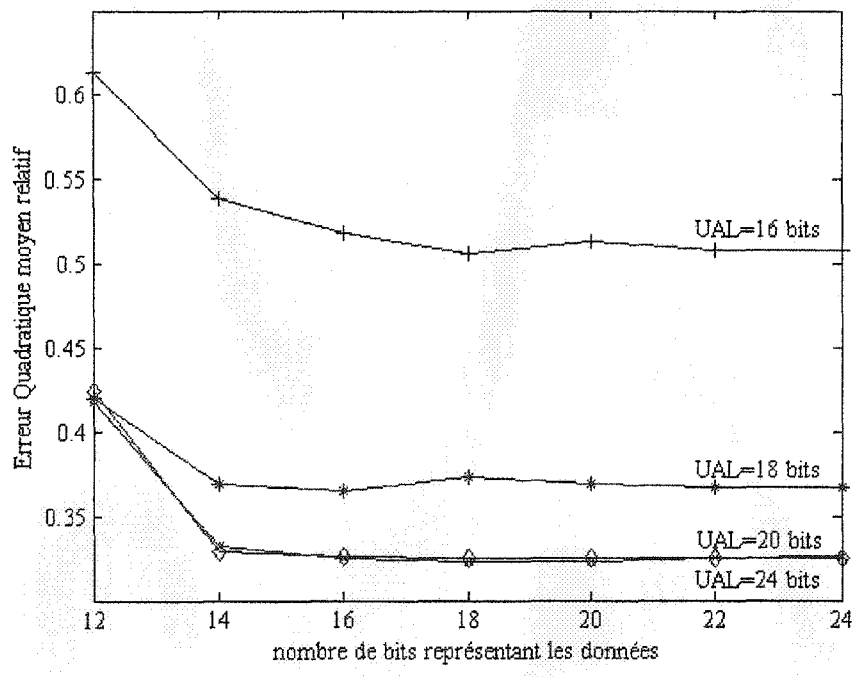
En conclusion, le nombre de bits suffisant pour concevoir les blocs de calcul (UAL) est 20 bits, de plus, pour représenter les données de mesure bruts 14 bits seront nécessaires.

Notons que des recherches précédents réalisées autour de l'algorithme de Kalman nécessite au moins 24 bits pour les blocs de calcul et 20 bits pour représenter les données[31][35][55], ce qui montre la puissance de notre méthode de quantification.

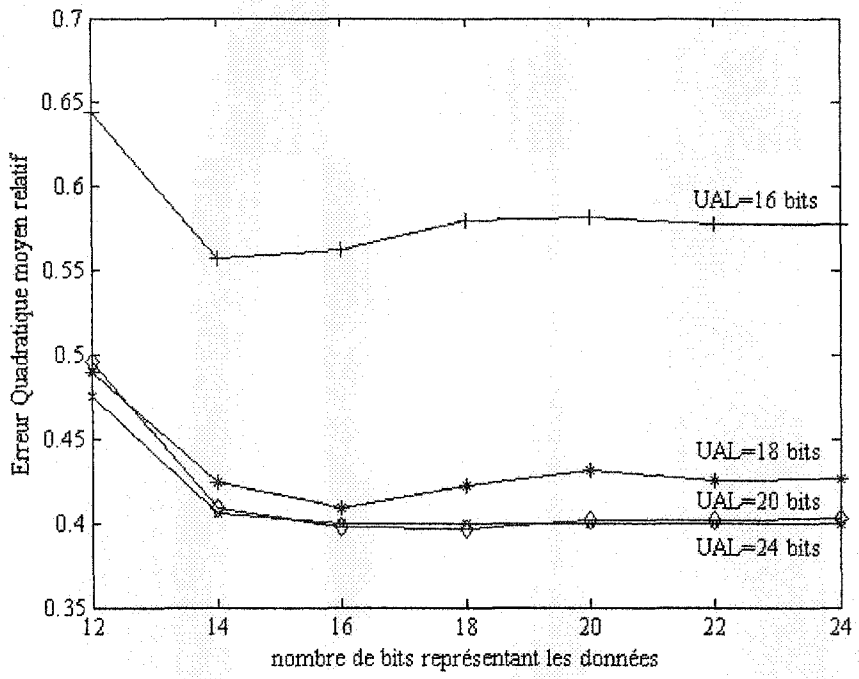
a)


Erreur Quadratique moyen relatif
0.7
0.65
0.6
0.55
0.5
0.45
0.4
0.35
0.3
12
14
16
18
20
22
24
nombre de bits représentant les données
UAL=16 bits
UAL=18 bits
UAL=20 bits
UAL=24 bits

b)


Erreur Quadratique moyen relatif
0.6
0.55
0.5
0.45
0.4
0.35
12
14
16
18
20
22
24
nombre de bits représentant les données
UAL=16 bits
UAL=18 bits
UAL=20 bits
UAL=24 bits

c)



d)

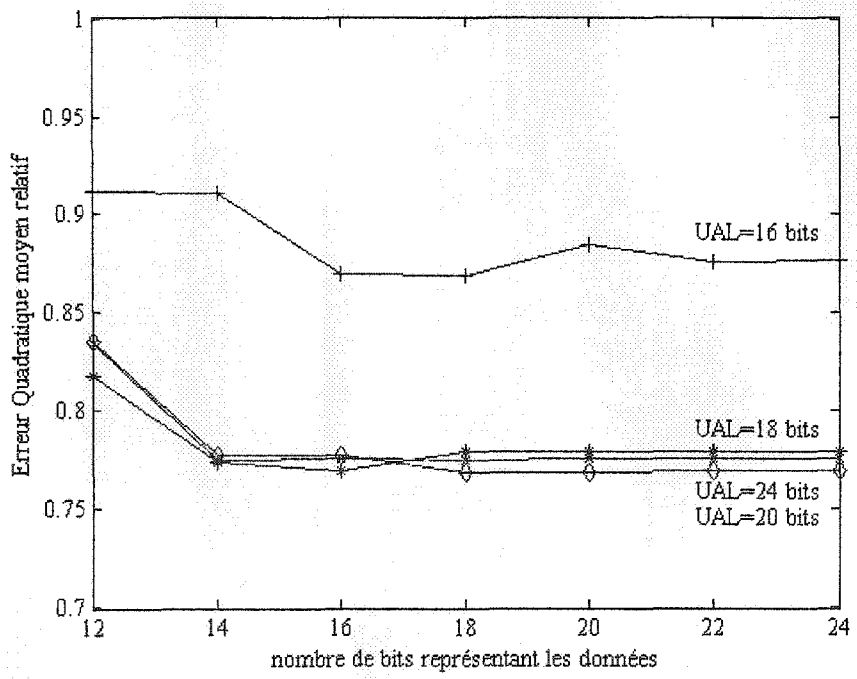


Figure 4.3 : Erreur quadratique moyen relatif de reconstitution en fonction de la longueur des mots représentant les données et la longueur des mots pour les blocs de calcul (UAL) suivant différents SNR a) 64.6dB b) 54.6dB c) 44.6dB d) 34.6dB.

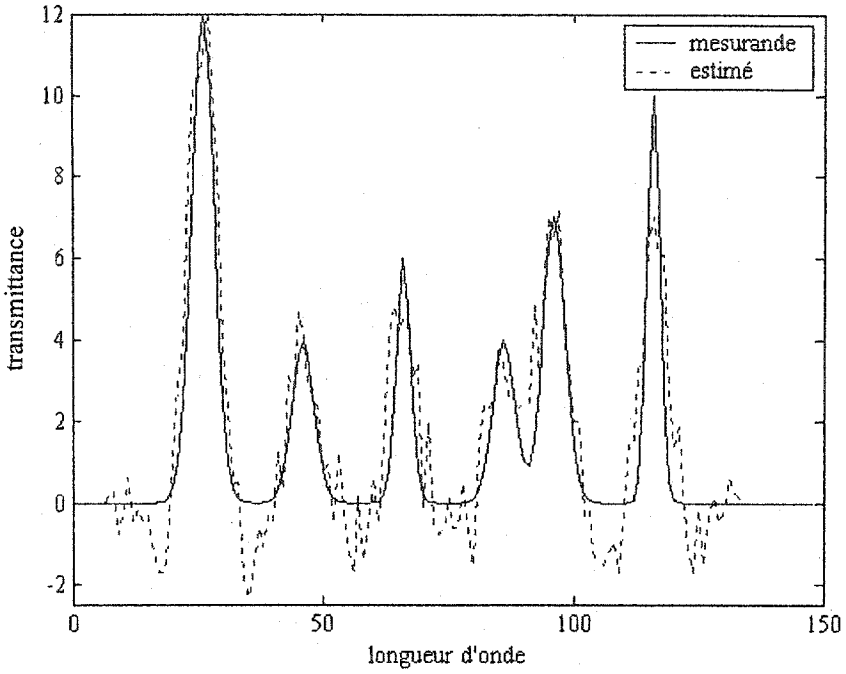


figure 4.4-a : Résultat de reconstitution en représentant les données sur 12 bits et en utilisant des blocs de calcul de 20 bits pour SNR=64.6dB.

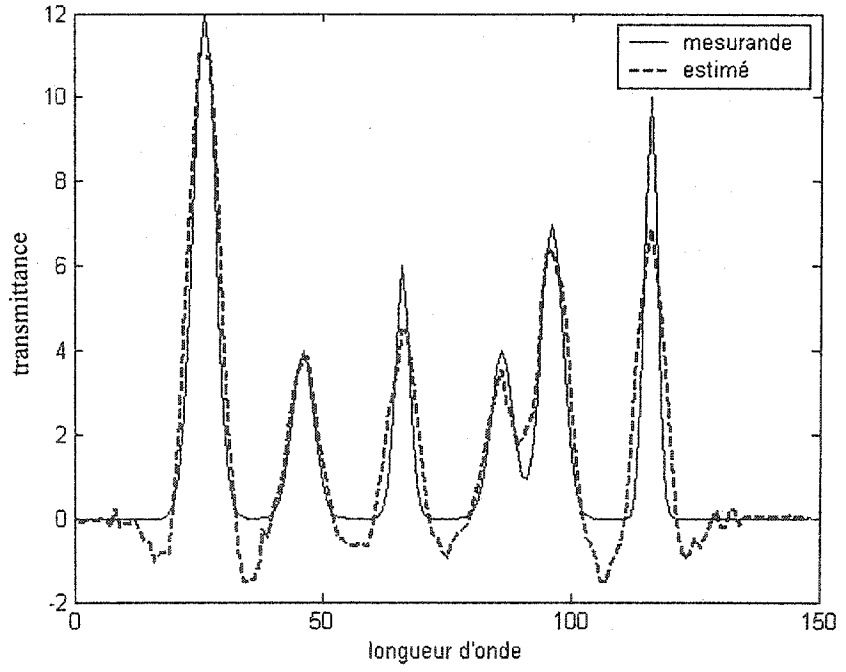


figure 4.4-b : Résultat de reconstitution en représentant les données sur 14 bits et en utilisant des blocs de calcul de 20 bits pour SNR=64.6dB.

4.3 Proposition d'une architecture ITGE dédiée

4.3.1 Architecture classique

Une première version d'un processeur spécialisé pour la reconstitution de signal mesuré basé sur le filtre en question est réalisé en première étape sans exploiter à fond le parallélisme inhérent au calcul. Cette première version a servi de base de travail pour la mise au point de notre seconde version. L'architecture de cette première version se présentait sur la figure 4.4. Le fonctionnement de cette architecture est similaire à celui du DSP 56309 mais avec moins de surface d'intégration sur silicium.

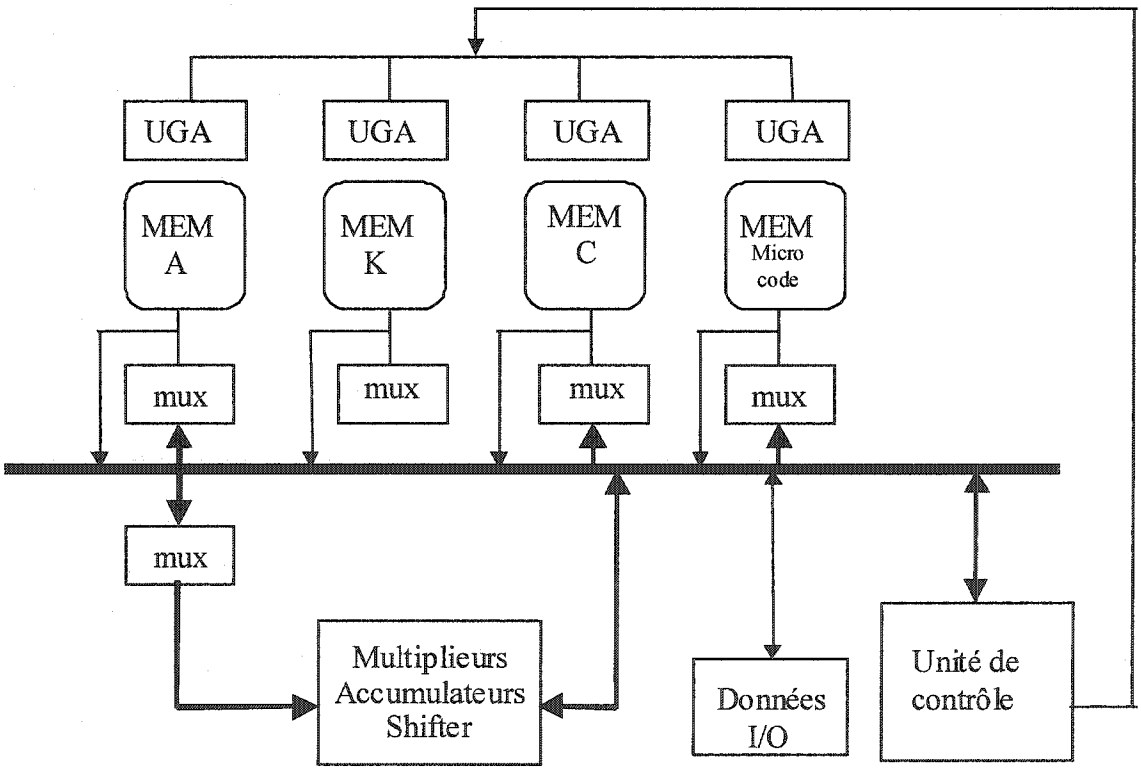


figure 4.5 : Première version du processeur

Dans cette architecture, les mémoire servent au stockage des matrices et paramètres de reconstitution et aussi les micro instructions du programme de reconstitution. Elles disposent chacune de leur propre bus de données. Les multiplexeurs sont utilisés pour lier les mémoires à l'unité arithmétique et logique (UAL). Cette dernière est constituée d'au moins deux multiplieurs, deux registres accumulateurs et un décaleur. Pour éviter les délais dans l'adressage, chaque mémoire a son unité de génération d'adresse. L'unité de contrôle contient tous les contrôles sur les module du ASIC.

4.3.2 Réseaux systoliques

4.3.2.1 Introduction

Pour augmenter les performances des processeurs, l'approche traditionnelle consiste à rechercher une augmentation de la vitesse des composants et à minimiser les périodes d'horloge. Aujourd'hui, cette approche est remise en cause. Tout d'abord l'industrie des semi conducteurs est d'une manière générale s'intéresse à développer des technologies à très haute densité d'intégration que de réaliser des circuits ultra rapides. Ensuite, on a atteint des limites physiques qui semblent incontournables jusqu'à l'arrivée sur le marché des technologies très rapides de la prochaine génération (à l'arséniure de gallium, GaAs) ; de plus, le prix à payer pour gagner un ordre de grandeur sur la vitesse des composants en utilisant ces nouvelles technologies sera vraisemblablement prohibitif pour la plupart des applications.

Plutôt que de compter seulement sur l'augmentation de la vitesse des composants, une autre possibilité est d'opter pour des systèmes parallèles qui pourront être implantés

efficacement sous forme de circuits intégrés: la complexité de la conception de ces circuits doit rester dans le cadre des meilleures possibilités industrielles.

De fait, la complexité des circuits intégrés disponibles à l'heure actuelle rend possible la réalisation à un faible coût de tels systèmes parallèles. Deux restrictions cependant [15] :

- il s'agit de processeurs spécialisés que l'on associe à un processeur hôte de type conventionnel ;

- la classe d'application est bien délimitée : les problèmes où le volume de calculs à effectuer priment largement sur les transferts de données à réaliser.

Le modèle systolique, introduit en 1978 par Kung et Leiserson, s'est révélé être un outil puissant pour la conception de processeurs intégrés spécialisés. En un mot, une architecture systolique est agencée en forme de réseau. Ces réseaux se composent d'un nombre de cellules élémentaires identiques et localement interconnectées appeler processeur élémentaire. Chaque cellule reçoit des données en provenance des cellules voisines, effectue un calcul simple, puis transmet les résultats, toujours aux cellules voisines, un temps de cycle plus tard.

Les cellules évoluent en parallèle, en principe sous le contrôle d'une horloge globale (synchronisme total) : plusieurs calculs sont effectués simultanément sur le réseau, et on peut "pipeliner" la résolution de plusieurs instances du même problème sur le réseau. La dénomination "systolique" provient d'une analogie entre la circulation des flux de données dans le réseau et celle du sang humain, l'horloge qui assure la synchronisation constituant le "coeur" du système.

4.3.2.2 Pourquoi des architectures systoliques ?

Concurrence et communication

Plusieurs calculs sont effectués sur une même donnée à l'intérieur du réseau : une fois qu'une donnée en provenance de la mémoire est lue par le réseau, elle passe de cellule en cellule et peut donc être utilisée plusieurs fois. Ce mode de calcul est à l'opposé du fonctionnement basé sur un accès mémoire à chaque utilisation d'une donnée, caractéristique des architectures séquentielles traditionnelles.

En conséquence, un réseau systolique peut être étendu (en joignant des cellules) pour traiter un problème coûteux en nombre d'opérations sans qu'il faille imposer pour autant une augmentation correspondante du débit de la mémoire. Cette propriété confère aux réseaux systoliques un avantage majeur sur les architectures traditionnelles.

Simplicité et régularité

Réaliser des circuits spécialisés à un coût raisonnable est une préoccupation majeure pour les concepteurs ITGE : le coût de conception d'un tel circuit doit être assez bas pour pouvoir être amorti sur un faible volume de production. Les architectures systoliques sont composées à partir d'un petit nombre de cellules de base, premier avantage sur une architecture composée d'une grande variété de cellules complexes. Deuxième avantage, l'interconnexion locale et régulière des cellules facilitent grandement l'implantation topologique de celles ci sur le circuit.

Enfin, il importe de dire un mot de la testabilité et de la forte résistance aux pannes des architectures systoliques. Les mêmes arguments que précédemment jouent en leur faveur dans ces domaines :

- pour le test : si les cellules sont identiques, il suffit d'en tester une seule;

- pour la résistance aux pannes : considérons un circuit comportant plusieurs rangées de cellules. On peut prévoir des mécanismes d'interrupteurs permettant de contourner les cellules qui se révéleraient défaillantes après réalisation. Il s'agit alors de reprogrammer le réseau en n'utilisant que les cellules valides, ce qui est facilité par la localité des connexions entre cellules.

Calculs intensifs

Les systèmes ITGE sont adaptés à l'implantation d'algorithmes "compute-bound" (c'est-à-dire où le nombre de calculs élémentaires est plus grand que le nombre de données en entrée-sortie) plutôt qu'à des problèmes "I/O-bound" (où la situation est inversée) car le nombre de ports d'entrée-sortie est limité. Par exemple, la multiplication de deux matrices de taille n, qui nécessite $O(n^3)$ multiplications et additions pour $O(n^2)$ données, est un problème "compute-bound", tandis que l'addition de deux matrices de taille n, qui nécessite n^2 additions pour $3n^2$ opérations d'entrée-sortie est un problème "I/O-bound".

Les caractéristiques des architectures systoliques conduisent dans la plupart des problèmes "compute-bound" à des calculs en temps réel, c'est-à-dire où les sorties sont délivrées au même rythme que les entrées. De fait, de telles architectures se sont révélées très performantes pour la résolution de nombreux problèmes où le volume de calcul est très important, et le traitement est régulier.

Peu de problèmes ont résisté à la systolisation

Même si tous les algorithmes proposés dans la littérature ne sont pas implémentés (ou implémentables) en ITGE, un tour d'horizon des réseaux existants a le mérite de montrer l'étendue du champ d'application de l'algorithmique systolique :

♦ traitement du signal et de l'image : filtrage; convolution 1D et 2D; corrélation; lissage par médiane 1D et 2D; transformée de Fourier discrète; projections géométriques; encodage/décodage pour les corrections d'erreurs; ...

♦ arithmétique matricielle : multiplication matrice-vecteur, matrice-matrice; triangularisation (résolution de systèmes linéaires, calcul de l'inverse d'une matrice); décomposition QR (calcul aux moindres carrés, inversion de matrice de covariance); problèmes aux valeurs propres; ...

♦ applications non numériques :

- structures de données : piles, files d'attente, recherche dans un dictionnaire, tri, etc.

- graphes et algorithmes géométriques : fermeture transitive, arbre de degré minimum, composantes connexes, enveloppes convexes, etc.

- manipulation de chaînes de caractères : occurrences d'un mot, plus longue sous-suite, reconnaissance de langages réguliers, etc.

- programmation dynamique,

- opérations sur des bases de données relationnelles,

- algèbre des polynômes : multiplication, division euclidienne, PGCD, etc.

- arithmétique entière et dans des corps finis : multiplication, division, PGCD, etc.

- simulation de type Monte-Carlo.

4.3.3 Proposition d'une nouvelle architecture

Afin d'atteindre une vitesse de traitement supérieure à celles des processeurs existants en plus d'une meilleure qualité de reconstitution de mesurandes, il faut profiter au maximum du parallélisme des équations de l'algorithme proposé. Les données sont représentées sur un bus de 14 bits, l'unité arithmétique et logique fonctionne à base de 20 bits. Les matrices d'état et le gain de Kalman sont supposés stockés dans des mémoires internes et ont une dimension de 2Mg=6.

Grâce à la nature récursive de l'équation d'état du filtre (3.14), les architectures systoliques sont très bien adaptées pour leurs implantation.

De manière générale, le processeur se subdivise en ses principales composants qui sont (figure 4.4) :

- le bloc de contrôle;
- le bloc mémoires et leurs compteurs;
- l'unité arithmétique et logique (UAL) et registres d'adresses/données.

Le bloc mémoire

La taille de mémoire est un facteur très important à tenir en compte dans la conception d'une architecture spécialisées. Ainsi, l'objectif serait de minimiser la taille des mémoires de stockage des matrices d'état A et C et le gain du filtre K. Comme on a un vecteur d'état de dimension 2Mg (égale à 6 pour notre cas), donc on prévoit (2Mg×2Mg)+2Mg+2Mg=48 cases mémoire présentées sur 14 bits. Pour optimiser

l'espace mémoire l'équation (4.12) serait utilisée. Dans ce cas, les nouveaux matrices à stocker sont $\mathbf{G'}= 2^{-n}\,\mathbf{G}$ et $\mathbf{K}$ et les cases mémoires sont en nombre de 42 présentées sur 14 bits, avec n vaut 4.

$$\begin{aligned}\hat{\mathbf{z}}(k+1) &= \big(\mathbf{A}(k)-\mathbf{K}(k)\mathbf{C}(k)\big)\hat{\mathbf{z}}(k)+\mathbf{K}(k)y(k)\\ &= 2^{n}\mathbf{G'}(k)\hat{\mathbf{z}}(k)+\mathbf{K}(k)y(k)\end{aligned} \tag{4.12}$$

Notons que les matrices G' et K satisfaisaient les conditions (4.2) et (4.4).

Le processeur proposé donc contiendra deux mémoires G' et K adressées par un compteur cyclique qui balaie toutes les adresses possibles en ordre croissant. Comme la taille des matrices est 2Mg, alors le compteur cyclique compte jusqu'à 2Mg et se remet à zéro jusqu'à activation d'un signal ``STOP``. La taille de la mémoire G' est 2Mg×2Mg cases présenté sur 14 bits, tandis que la taille de K est 2Mg.

De plus, une troisième mémoire de type RAM est utilisée pour stocker le vecteur de reconstitution intermédiaire $\hat{\mathbf{x}}_{int}$. Si le nombre d'échantillon du mesurande est égale par exemple N=64, on doit prévoir une mémoire de 64 cases de 14 bits.

En conclusion, le processeur doit contenir trois mémoires : deux pour les matrices G' et K et l'autre de type RAM. (fig. 4.5)

Le pointeur commun d'adresse des mémoires G' et K est un compteur cyclique de longueur 2Mg. Tandis que la mémoire RAM est pointée par un compteur / décompteur de taille N ou être conçu comme un pile ``dernier entrant premier sortant`` (LIFO).

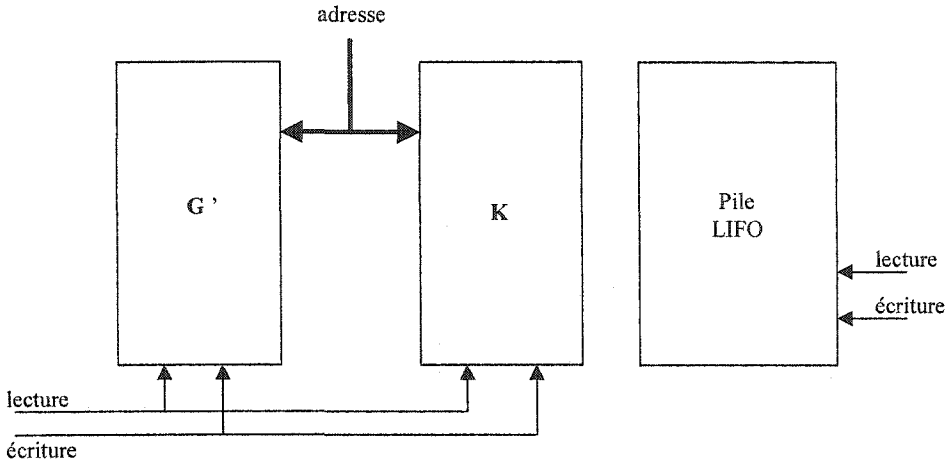


Figure 4.6 : Structure des mémoires

Une autre façon de concevoir la mémoire de stockage des matrices, serait de stocker les transposées des matrices K et G' sur la même mémoire qui aura 42 cases de 14 bits (fig. 4.6), cela nous permet de diminuer le nombre de processeurs élémentaires de 2Mg+1 à 2Mg.

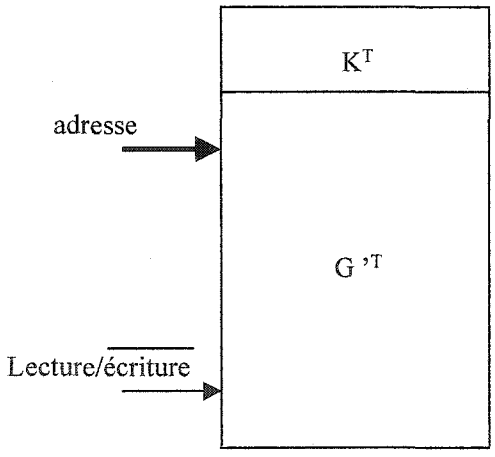


Figure 4.7 : Structure de la mémoire englobant G' et K

Les pointeurs des mémoires

Le compteur pointant les mémoires G' et K, est de type synchrone de taille 2Mg avec un RAZ et un signal ``STOP``. Le compteur pointeur du mémoire RAM est aussi synchrone de taille N avec un RAZ et un signal ``STOP``.

Structure de bus

Afin de réduire le coût de la mise en œuvre du processeur, il faut choisir une structure de bus minimale pour soutenir l'ensemble des transferts de données à l'intérieur et à l'extérieur du le processeur.

Les transferts de données dans le processeur sont : le chargement des mémoires et registres, l'entrée des données de mesure (échantillons à reconstituer) issues du convertisseur analogique / numérique (A/N), ainsi que la sortie de l'échantillon reconstitué. À l'intérieur du processeur, les flots de données englobent les transferts entre les mémoires extérieures et l'UAL.

Le transfert se fait en deux phases :

- phase d'initialisation : cette phase est celle pendant laquelle les deux mémoires G' et K du processeur sont chargées. Toutes les données sont issues de l'extérieur et y arrivent sous forme de mots de 14 bits, d'où la présence d'un port de données de 14 bits. Un bus interne de données de 14 bits sert à acheminer ces données vers la bonne mémoire de destination.

- phase de traitement : cette phase est celle pendant laquelle notre algorithme de reconstitution est en cours d'exécution, le bus interne sert maintenant à l'entrée de la

mesure issu de l'extérieur. un second bus a pour rôle de convoyer toute données en provenance de l'UAL.

Le bloc de contrôle

Le bloc de contrôle a pour rôle d'assurer le fonctionnement de l'ensemble selon un graphe d'état préalablement conçu. Ce bloc doit avoir les caractéristiques suivantes :

- pouvoir commander et synchroniser une boucle de calcul;
- pouvoir effectuer des sauts et des retours de sous-taches;
- pouvoir générer les adresses;
- pouvoir être arrêté de l'extérieur du processeur.

L'unité arithmétique et logique

Sa structure nous est dictée par l'algorithme parallélisé. Pour avoir l'opération de multiplication par une multiple de 2, suite à l'équation $G = 2^n G'$, il suffit de répéter le décalage à gauche n fois.

La figure 4.7 montre l'architecture systolique proposée sous forme d'un réseau linéaire de 2Mg=6 processeurs élémentaires (PE). La tâche du processeur est de trouver le vecteur d'état $\hat{\mathbf{z}}(k+1) = 2^n \mathbf{G}'(k)\hat{\mathbf{z}}(k) + \mathbf{K}(k)y(k)$ après un certain nombre de cycle horloge bien déterminé, et d'extraire l'estimé qui égale à $\hat{\mathbf{z}}(5)$ à chaque itération. Notons que chaque itération dure 2Mg+1 cycles d'horloge(latence du réseau), pour avoir le premier résultat on prévoit 2Mg+2 cycles d'horloge.

Le réseau est utilisé deux fois, la première pour calculer le vecteur de reconstitution intermédiaire et la seconde, pour calculer l'estimé du mesurande.

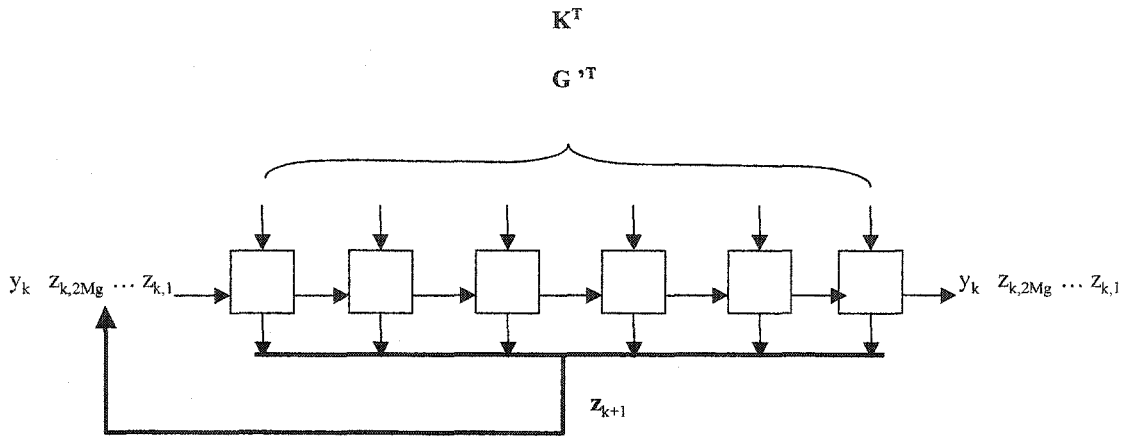


Figure 4.8 : Architecture systolique proposée

Les cellules du réseau sont constituées d'un multiplieur, un additionneur accumulateur, 2 registres et un décaleur à gauche de n fois (n=4). Ce décaleur ne fonctionne pas quand le compteur de la mémoire devient 2Mg+1. En effet, durant les 2Mg cycles de chaque itération, c'est le produit $\hat{\mathbf{z}}_1(k+1,i) = 2^n \mathbf{G}'(i,k)\hat{\mathbf{z}}(k,i)$ qui s'exécute, tandis qu'au dernier cycle horloge de chaque itération, on calcule $\hat{\mathbf{z}}(k+1,i) = \hat{\mathbf{z}}_1(k+1,i) + \mathbf{K}(k,i)y(k)$, (fig. 4.8)

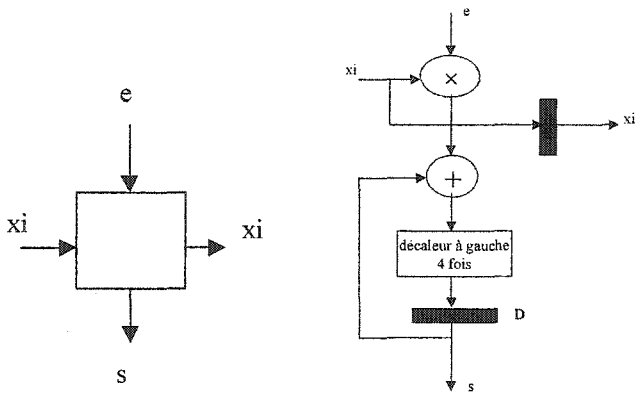


Figure 4.9 : Structure des processeurs élémentaires

les entrées ($\tilde{y}_k$) / sorties ($\hat{x}_k$) sont acheminées à l'extérieur sur le même bus (14 broches). En effet, l'échantillonnage de la mesure brute y_k se fait à chaque itération (2Mg+1 cycles d'horloge). Le bus sera donc disponible durant les autres cycles pour acheminer l'estimé, cette sortie sera verrouiller par un ``Latch`` jusqu'à la prochaine sortie et ceci nous fait économiser 14 broches (fig. 4.9), de plus le bus d'entrée sera en haute impédance. Cette technologie est utilisée par la compagnie Intel dans la conception de ces micro processeurs 8086.

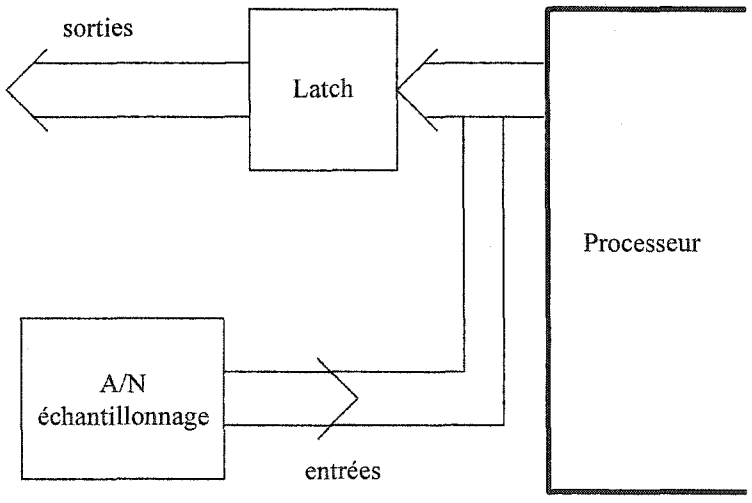


Figure 4.10 : Acheminement des entrées et sorties sur le même bus

Pour un grand nombre d'échantillons de la mesure brute N, on ne peut que concevoir une pile (stockage du vecteur de reconstitution intermédiaire) à l'extérieur du processeur. Une broche serait ainsi ajoutée pour commander la pile et un multiplexeur. Durant la première phase, le processeur transmet le vecteur du résultat de reconstitution intermédiaire sur la pile et le multiplexeur sélectionne la mesure brute $\tilde{y}_k$ comme entrée

pour le processeur, le bloc D_ctrl achemine vers la mémoire les données du bus provenant du Latch.

Durant la deuxième phase, la mémoire est en mode lecture, le bus provenant du Latch vers le bloc D_ctrl sera mis en haute impédance et les données ($\hat{\mathbf{x}}_{int}$) issues de la pile seront transmis vers le multiplexeur. Ce dernier met en haute impédance le bus qui achemine $\tilde{y}_k$ et laisse le vecteur $\hat{\mathbf{x}}_{int}$ acheminé vers l'entrée du processeur. Notons que dans cette phase l'estimé sera disponible à la sortie du latch et sera utilisé selon l'application.(fig. 4.10)

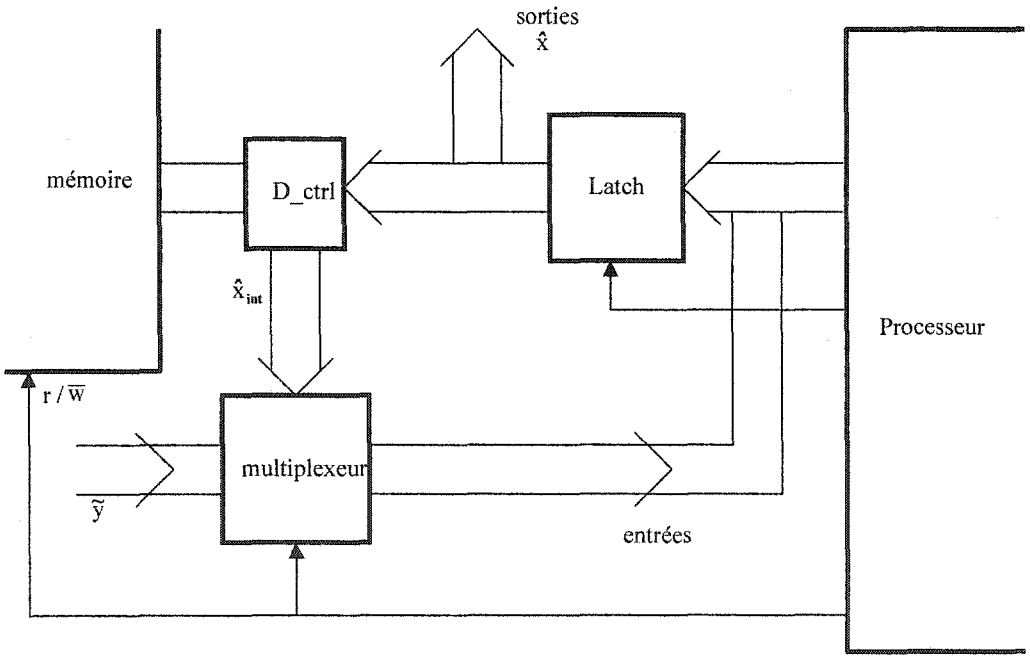


Figure 4.11 : Environnement du processeur dans le cas ou la mémoire est conçu à l'extérieur

l'architecture générale du processeur proposé avec mémoire pile intégrée est représentée sur la figure 4.11. Il contient :

- (2Mg+1)×(2Mg)=42 cases mémoires présentées sur 14 bits pour le stockage des matrices **G'** et **K**,
- une mémoire de type pile LIFO de taille 64,
- un bloc de contrôle,
- unité arithmétique et logique sous forme d'un réseau systolique linéaire de 2Mg cellules.

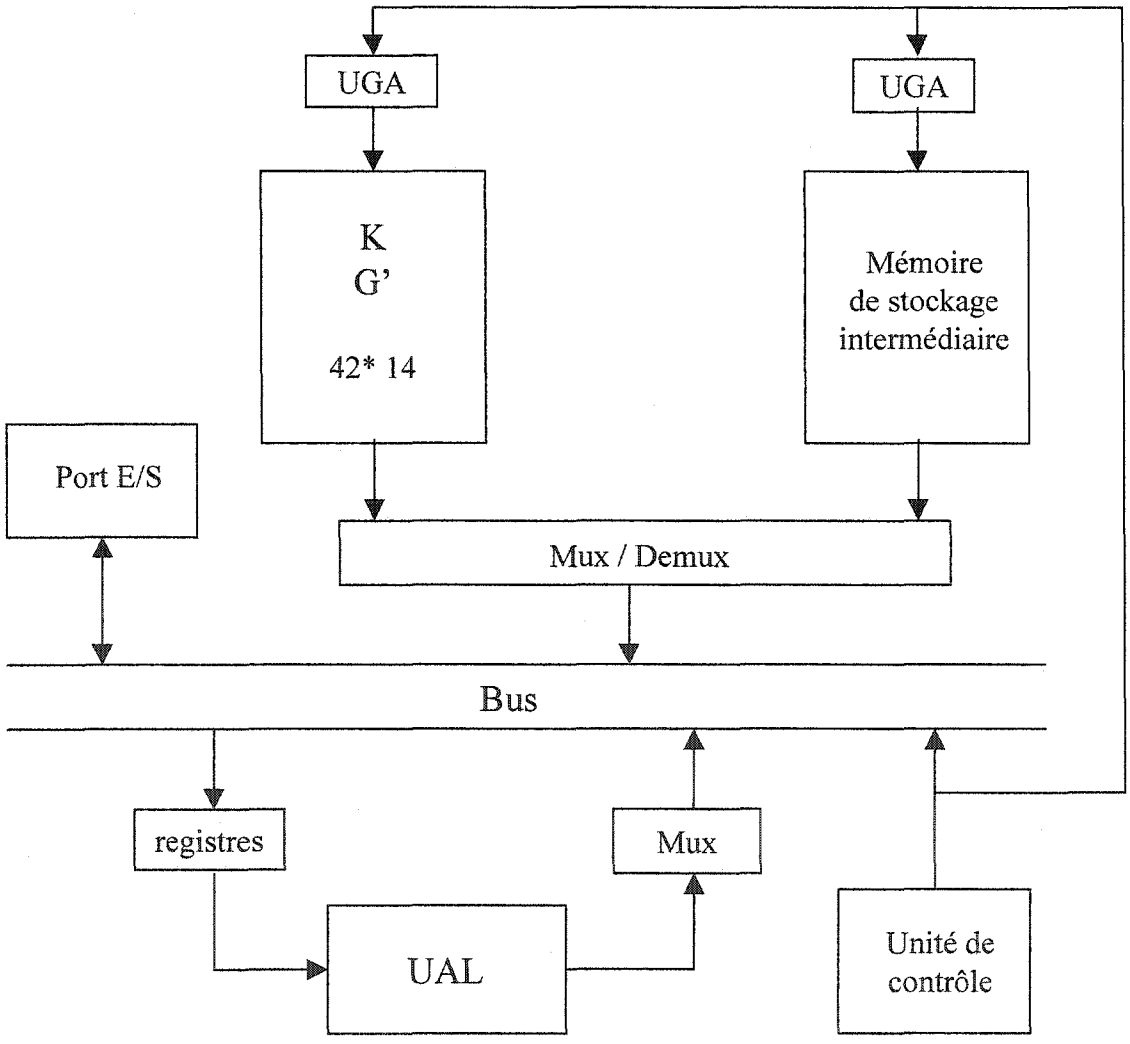


Figure 4.12 : Architecture globale du processeur proposé

4.3.4 Modélisation de l'architecture

Le langage VHDL et l'environnement de peakVHDL

Le langage de modélisation VHDL (Very High Speed Integrated Circuit Hardware Description Language), est un langage très modulaire, puissant et général. On ne va pas y écrire de longues descriptions, mais des unités plus petites et hiérarchisées. C'est également un langage moderne où l'on utilise une grande partie de l'expérience acquise en génie matériel et en génie logiciel. Notons également que les développements de description VHDL sont prévus pour pouvoir être exécutés en parallèle par des unités complètes. De plus, puisque les unités de compilation sont séparées, les modèles sont conçus pour être réutilisables.

La méthodologie du top-down design a été adoptée pour faciliter la conception. Ainsi l'architecture a été décomposée en plusieurs blocs simples qui sont utilisés fréquemment avec une simple interface; par conséquent, une réduction substantielle en terme de temps de développement de l'architecture est atteinte. Le contrôle dans cette architecture est simple. Un contrôle systolique est réalisé puisque tout les cellules font la même chose avec un simple décalage.

Nous avons modélisé le processeur grâce aux outils du logiciel PeakVHDL Design Suite de Accolade. Le langage VHDL utilisé par le simulateur est celui définie par le standard IEEE 1076-1993; le standard IEEE 1164 est totalement supporté ; le paquetage 1076.3 (numeric_std) est supporté et inclue dans IEEE.LIB; le paquetage Synopsys std_logic_arith et autres paquetages Synopsys sont aussi inclues dans IEEE.LIB,

accompagnée avec le logiciel. Le logiciel HDL Designer-Pro de Mentor Graphics a été aussi utilisé dans notre modélisation des processeurs élémentaires. Le processeur a été testé avec des données synthétiques et les résultats sont en annexe. La synthèse n'a pas été fait vu que le logiciel ne supporte pas ce module.

Résultats et discussion

L'utilisation de circuits à applications spécifiques pour résoudre les équations du filtre H∞ est bien justifiée. L'implantation d'opérations arithmétiques très complexes nécessite une grande capacité de calcul. Le nombre de ces opérations est fonction de la taille du modèle d'état, du nombre d'échantillons à traiter et de la taille de fonction de transfert du système de mesure.

La modularité des réseaux systoliques permet d'adapter les architectures à plusieurs types et tailles des problèmes. En plus, il est possible de systoliser plusieurs problèmes , car le modèle systolique se prête bien au calcul numérique (multiplication matrice-vecteur, convolution, déconvolution ...).

Le nombre de bits suffisant pour concevoir les blocs de calcul (UAL) est 20 bits, de plus, pour représenter les données 14 bits est suffisant ce qui permettre la réduction de la surface d'intégration.

La transformation de l'équation du système à une forme matricielle assure un grande économie en terme de surface d'intégration, puissance consommée et temps de calcul. Les performances du réseau systolique ont été démontrées pour faire la multiplication matrices-vecteur.

La répartition du calcul sur les cellules élémentaires permet de réduire le temps de calcul d'un problème d'ordre $O(N^2)$ pour un calcul séquentiel à un temps d'exécution d'ordre O(N) en utilisant un réseau de N cellules.

Le stockage des résultats intermédiaires entre les deux passages constitue un faiblesse de l'algorithme du point de vue implantation en silicium si N est grand, car il faut disposer d'une mémoire externe avec une taille assez grande généralement, cette solution est applicable pour des applications non rapide ou qui ne nécessite pas le traitement en temps réel.

CONCLUSION

L'objectif de ce mémoire est d'effectuer une étude algorithmique d'un filtre H∞ et monter l'apport de ce filtre dans la résolution des problèmes de reconstitution des systèmes de mesure. De plus l'un des objectifs de ce projet est l'analyse du développement et de l'implantation de l'algorithme dans un processeur numérique à usage général (DSP 56309 de Motorola), ainsi la proposition d'une architecture d'un processeur dédié pour la reconstitution de mesurandes dynamiques.

Nos principaux objectifs ont été atteints, en l'occurrence :

- élaboration d'un algorithme de reconstitution de mesurandes dynamiques basé sur le filtrage H∞ ;
- étude de l'algorithme proposé en se basant sur des données afin de vérifier ses performances (exactitude, complexité et temps de calcul). La comparaison de l'algorithme proposé avec les algorithmes de référence a montré la puissance de l'algorithme en terme de qualité et complexité de calcul. L'algorithme proposé est beaucoup plus rapide que les algorithmes de référence et surtout si le nombre d'échantillons est plus grand. L'algorithme proposé donne des résultats meilleurs que l'algorithme basé sur le filtre de Kalman. On peut donc l'utiliser dans les applications temps réel qui exigent la rapidité et l'exactitude de reconstitution;

- validation de l'algorithme en utilisant des données réelles spectrométriques. Ces résultats montrent que l'algorithme proposé donne de bons résultats de reconstitution et peut être utilisé comme une première étape dans la procédure de détermination les positions des pics du spectre et leurs amplitudes;
- réduction de la complexité de l'algorithme en réduisant le nombre de paramètres libres de 4 à un seul paramètres β et en diminuant le nombre d'opérations arithmétiques;
- étude de quantification a été effectuée pour déterminer la largeur des mots nécessaires pour représenter les données et les blocs de calcul et ainsi assurer une bonne exactitude utilisant un calcul à virgule fixe et une représentation en complément à deux. Pour les blocs de calcul (UAL), le nombre de bits suffisant est de 20 bits, et pour représenter les données 14 bits seront suffisants;
- la conception d'un algorithme optimisé pour un processeur numérique à usage général;
- proposition d'une architecture du processeur spécialisé pour la reconstitution de signaux basé sur le filtrage H∞ en utilisant le parallélisme dans l'algorithme;
- la conception, la modélisation VHDL et la simulation dans l'environnement peakVHDL de Accolade et HDL Designer-Pro de Mentor Graphics validant l'architecture en question;

En comparaison aux autres algorithmes de référence pour la reconstitution de signaux spectrométriques, nous somme arrivé à la conclusion que le filtrage H∞ offre d'excellents résultats de reconstitution avec une rapidité de traitement.

L'élaboration de ce mémoire m'a permis de me familiariser avec les principaux algorithmes de reconstitution de mesurandes, l'optimisation LQ du filtrage H∞, ainsi qu'aux différents outils de conception ITGE.

Ce mémoire représente une contribution au développement du filtrage H∞ appliqué aux systèmes de mesure spectrométriques et d'architectures en technologie ITGE pour la reconstitution de mesurandes.

BIBLIOGRAPHIE

[1] M.Ben Slima.
Algorithmes de reconstitution de mesurandes dynamiques avec régularisation multiple. Thèse de doctorat, INRS-Télécommunications, décembre 1995.

[2] X. Yu, C. S. Hsu, R. H. Bamberger, S. J. Reeves.
H∞ Deconvolution filter design and its application in image restoration.
Proc. IEEE Int. Conf. Acoustic, Speech & Signal Process. 1995

[3] X. Shen.
Discrete H∞ filter desing with application to speech enhancement.
Proc. IEEE Int. Conf. Acoustic, Speech & Signal Process. 1995.

[4] Uri Shaked, Yahali Theodor.
A frequency domain approach to the problems of H∞ - Minimum error state estimation and decomvolution.
IEEE Transactions on Signal Processing, vol40. N^0 12, Déc. 1992.

[5] X. Shen.
Robust H∞ Filtering for Continuous Time Varying Uncertain Systems with Deterministic Input Signals.
Proc.IEEE Int. Conf. Acoustic, Speech & Signal Process.1995.

[6] M. Ben Slima, R. Z. Morawski, and A. Barwicz.
A spline-based variational method with constraints for spectrophotometric data correction.
IEEE Trans. Instr. & Meas., vol. IM-41, pp. 786-790, Dec. 1992.

[7] D. Massicotte, R.Z. Morawski, A.Barwicz.
Kalman-filter-based algorithm of spectrometric data correction, part I : An iterative algorithm of deconvolution.
IEEE Trans .Instr.&Meas., Vol.46 ,No3, pp.678-684,1997.

[8] G. Demoment and D. Saint-Felix.
Déconvolution en temps réel par une méthode sous-optimale rapide.
Ondes et Signal, no. 76, pp. 133-147, 1983.

[9] P. A. Jansson.
Deconvolution with Application in spectroscopy.
Academic Press, London, 1984.

[10] Coordonnateur : Alain Oustaloup
La Robustesse : Analyse et Synthèse de Commandes Robustes.
Paris : Hermès , 1994

[11] John C. Doyle, Keith Glover, Pramod P. Khargonekar, Bruce A. Francis.
State-Space Solution to Standard H_2 and H∞ Control Problems.
IEEE Transactions on Automatic Control, vol 34. N^0 8, Aout 1989.

[12] A. Zaafouri, A. Kochbati, M. Ksouri.
Techniques LMI pour la commande d'un Système Électromécanique Incertain.

[13] Krishan M. Nagpal, Pramod P. Khargonekar.
Filtring and smouthing in H∞ Setting.
IEEE Transactions on Automatic Control, vol 36. N^0 2, Février 1991.

[14] P.H. Sydenham.
Handbook of Measurement Science. Vol. 2.
Wiley, Chichester, 1983.

[15] L. Finkelstein.
Measurement and instrumentation science – an analytical review.
Measurement Journal of IMEKO (Special Issue), vol. 14, no 1, 1994.

[16] A. Barwicz.
System approach to electric measurement.
in Proc. IEEE Trans. Instr. Meas., Techn., conference, (Irvine, 14-16 May 1993) pp. 397-402, 1993.

[17] R.Z. Morawski.
Unified approach to measurand reconstitution.
IEEE Trans. Instr. Meas., vol. IM-43, pp. 226-231, Apr. 1994.

[18] R. Z. Morawski, M. Ben Slima, M. Milewski, and A. Barwicz.
Application of digital signal processor for correction of spectrophotometric data.
IEEE Trans. Instr. & Meas., vol. IM-42, pp. 778-782, Jun. 1993.

[19] A. Bennia and N. S. Nahman.
Deconvolution of causal pulse and transient data.
in Proc. IEEE Instr. & Meas. Techn. Conference, (San José, 13-15 Fev. 1990), pp. 251-254, 1990.

[20] L. Matthies.
Stereo vision for planetery rovers: Stochastic modeling to near real-time implementation.
Int. Journal of Computer Vision, vol. 8, no. 1, pp. 71-91, 1992.

[21] Y. G. Biraud.
Les méthodes de déconvolutions et leurs limitations fondamentales.
Revue de Physique appliquée, vol. 11, pp. 203-214, Mar. 1976.

[22] R. Prost and R. Goutte.
Deconvolution when the convolution kernel has no inverse.
IEEE Trans. Acoust., Speech & Signal Processing, vol. ASSP-25, pp. 542-549, Dec. 1977.

[23] R. Z. Morawski.
Basic problems of measurement signal reconstruction.
in Proc. Australien Inst. and Meas. Conference, (Adelaide 14-16 Nov. 1989), 1989.

[24] R. Z. Morawski and A. Miekina.
Noise reduction in spectrophotometric data processing using Tikhonov's regularisation.
in Proc. 1st Int. IMEKO-TC4 Symp. on Noise in Elec. Meas., (Coms, 19-21 Jun. 1986), pp. 217-222, 1986.

[25] M. I. Sezan et al.
Regularized signal restoration using the theory of convex projections.
in Proc. IEEE ICASSP Conference, (Texas, 6-9 Apr. 1987), pp. 36.4.1- 36.4.4, 1987.

[26] J. K. Tugnait.
Constrained signal restoration via iterated extended kalman filtering.
IEEE Trans. Acoust., Speech & Signal Processing, vol. 32, no. 2, pp. 472-475, 1985.

[27] R. W. Schafer et al.
Constrained iterative restoration algorithms.
Proc. IEEE, vol. 69, pp. 432-450, Apr. 1981.

[28] A. Barwicz, J. L. Dion, and R. Z. Morawski.
Calibration of an electronic measuring system for ultrasonic analysis of solutions.
IEEE Trans. Instr. & Meas., vol. IM-39, pp. 1030-1033, Dec. 1990.

[29] A. Guershaoui et al.
Deconvolution des données du types sismiques: étude comparative de certaines méthodes
in 11ieme Colloque sur le Traitement du Signal et des Images, (Nice, 1-5 Jun. 1987), pp. 725-728, 1987.

[30] A. Miekina and R. Z. Morawski.
Incorporation of the positivity constraint into a Tikhonov-method based algorithm of measurand reconstruction.

in Proc. IMEKO-TC1& TC7 Colloquium on State and Advances of Meas.& Instr. Sci., (London, 8-10 Sept. 1993), pp. 299-304, 1993.

[31] D. Massicotte, R. Z. Morawski, and A. Barwicz.
Incorporation of a positivity constraint into a Kalman-filter-based algorithm for correction of spectrophotometric data
in Proc. IEEE Instr. & Meas. Techn. Conference, (New York, 12-14 May 1992), pp. 590-593, 1992.

[32] M. Ben Slima, R. Z. Morawski, and A. Barwicz.
A recursive spline-based algorithm for spectrophotometric data correction.
in Proc. IEEE Instr. & Meas. Techn. Conference, (Irvine, 18-20 May 1993), pp. 500-503, 1993.

[33] H. C. Andrews and B. R. Hunt.
Digital Image Restoration. Prentice Hall, Englewood
Cliffs N.J., 1977.

[34] A. Barwicz, M. Ben Slima, D. Massicotte, R. Z. Morawski, and C. Thellen.
Improving the resolution of analytical instrument in environmental labortories.
in Proc. IEEE Instr. & Meas. Techn. Conference, (Hamamatsou, 10-14 May 1994), pp. 454-457, 1994.

[35] D.Massicotte.
Une approche à l'implantation en technologie VLSI d'une classe d'algorithmes de reconstitution de signaux.
Ph.D. Thèse, Ecole Polytechnique de montreal,1995.

[36] M.Newell and J.Rasure.
A VLSI System for Real-Time Linear Operations and Transforms.
IEEE Trans. On Signal Processing, Vol. 39,No.8, pp.1914-1917,August 1991.

[37] H. T. Kung.
Why Systolic Architectures.
IEEE Trans. Computer, pp.37-46,January 1982.

[38] M. R. Azimi, T. Lu et E. M. Nebot.
Parallel and sequential block Kalman filtering and their implementation using systolic arrays.
IEEE Trans. On Signal Processing, Vol .39,Iss.1,pp.137-147,1991.

[39] D.Bursky.
Reprogrammable IC takes on graphics, video, and audio.
Electronic Design, Vol.43,Iss: 21,pp.133-4,136-7,Oct 1995.

[40] P. Pirsch; J.Kneip; K.Ronner.

Parallelization resources of image processing algorithms and their maping on a programmable parallel videosignal processor.
1995 IEEE Symposium on Circuits and Systems, Vol. 1, pp.562-5,1995.

[41] P. H. Van Cittert.
Zum Einuss der Spaltbreit auf die Intensitatsverteilung in Spektrallinien II.
Z. Physik, vol. 69, pp. 298-308, 1931.

[42] E.Ercanli et C. Parachristou.
A register file and scheduling model for application specific processor synthesis.
33rd Design Automation Conference, pp. 35-40,1996.

[43] R. Yung et N.C. Wilhelm.
Caching processor general registers.
International conference on computer Design : VLSI in Computers and Processors, pp. 307-12.

[44] Z. Tang; C.Zhang; L.Sifei; T.Yu.
A new architecture for branch-intensive loops.
Processings. Advances in Parallel and Distributing Computing, pp. 241-6.

[45] S.Dutta; A.Wolf; W.Wolf et K.J.O'Connor.
Design issues for very-long-instruction-word VLSI video signal processors.
VLSI Signal Processing Conference, IX, pp.95-104.

[46] H.Qing et H.C.Huan.
RNIW : a novel general-purpose DSP architecture.
1996 IEEE International Conference on Acoustics, Speech, and Signal Processing Conference Proceedings, pp. 3302-5,Vol.6.

[47] M.Moonen; P.Van Dooren et J.Vandewalle.
Combined Jacobi-type algorithms in signal processing.
SVD and Signal Processing, Il. Algorithms, Analysis and Application, Amsterdam, Netherlands, 25-27 June 1990.

[48] Barwicz, w.i. Mitchell
Systolic array for Kalman filtering with algorithm-based fault to tolerance.
SDV and Signal Processing, II. Algorithms, Analysis and Application, Amsterdam, Netherlands, 25-27 June 1990.

[49] S.Y Kung et J.N. Hwang.
Systolic array designs for Kalman filtering.
IEEE Trans. On signal Processing, Vol. 39, pp. 171-82, Jan. 91.

[50] A. Barwicz.
System Approach to Electrical Measurements.

Conference Record IEEE IMTC/93, Irwine, california, May 18-20, pp. 397-402, 1993.

[51] D. Massicotte, M.A. santerre, Y. Savaria and A. Barwicz.
Structure de calculs parallèles pour le filtrage de Kalman dans la reconstitution de signaux.
CCECE'92, Toronto, pp. TM4.16.1-TM4.16.4, 1992.

[52] M.Kunt et al.
Techniques modernes de traitement numérique des signaux.
Presse polytechniques et universitaires romandes, vol. 1, 1991.

[53] M. Béllanger.
Traitement numérique du signal-théorie et pratique.
Masson, 1990.

[54] J.L Hennessey ET D. A. Patterson.
Architectures des ordinateurs : une approche quantitative.
McGRAW-HULL 1992.

[55] M. Ben Slima, L. Szczecinski, D. Massicotte, A. Barwicz, and R.Z. Morawski.
Algorithmic Specification of an Specialized Processor For Spectrometric Applications.
IEEE Instrum & Meas. Technology Conference (IMTC/97), Ottawa, Canada, pp. 90-95, 19-21 May 1997.

[55] P. Quinton et Y. Robert.
Algorithmes et architectures systoliques.
Masson, 1989.

[56] Intel.
I860-64 Bit Microprocessor, Assembler and Linker
Reference Manual, 1989.

[57] G. Demoment.
Déconvolution des signaux.
Tech. Rep. L2S 20/84, Laboratoires des signaux et systèmes CNRS/ESE, France, 1985.

[58] A. N. Tikhonov and V. Y. Arsenine.
Solutions of Ill-Posed Problems.
Wiely/Winston, London, 1977.

[59] J. Hadamard.
Sur les problèmes aux dérivées partielle et leur signification physique.
Univ. Princeton, 1902.

[60] A. N. Tikhonov and V. Y. Arsenine.
Solutions of Ill-Posed Problems.
Wiely/Winston, London, 1977.

[61] R. Z. Morawski and A. Miekina.
Noise reduction in spectrophotometric data processing using Tikhonov's regularisation.
in Proc. 1st Int. IMEKO-TC4 Symp. on Noise in Elec. Meas., (Coms, 19-21 Jun. 1986), pp. 217{222, 1986.

[62] D. L. Phillips.
A technique for the numerical solution of certain integral equations of the first kind.
J. Ass. Comput. Mach., vol. 9, pp. 84-97, 1962.

[63] S. Twomey.
On the numerical solution of Fredholm integral equations of the first kind by inversion of the linear systems produced by quadrature.
J. Ass. Comput. Mach., vol. 10, pp. 97-101, 1963.

[64] B. R. Hunt.
The application of constrained least-squares estimation to image restoration by digital computer.
IEEE Trans. Computers, vol. 22, no. 9, pp. 805-812, 1973.

[65] G. Demoment.
Image reconstruction and restoration - overview of common estimation structures and problems.
IEEE Trans. Acoust., Speech & Signal Processing, vol. 37, pp. 2024-2036, Dec. 1989.

[66] C. Chi and J. R. Mendel.
Performance of minimum variance deconvolution filter.
in Proc. IEEE ICASSP Conference, (San Diego, 19-21 Mar. 1984), pp. 28B.2.1-28B.2.4, 1984.

[67] J. K. Tugnait.
Constrained signal restoration via iterated extended kalman filtering.
IEEE Trans. Acoust., Speech & Signal Processing, vol. 32, no. 2, pp. 472-475, 1985.

[68] P. Moulin.
An adaptive spline method for signal restoration.
in Proc. IEEE ICASSP Conference, (San Fransisco, 23-26 Mar. 1992), pp. 169-172, 1992.

[69] P. B. Crilly.
A quantitative evaluation of various iterative deconvolution algorithms.
IEEE Trans. Instr. & Meas., vol. IM-40, pp. 558-562, Jun. 1991.

[70] L. Finkelstein.
Fundamental concept of measurement : Definitions and scales.
Measurement & Control, vol. 8, Mar. 1975

[71] P.H. Sydenham.
Handbook of Measurement Science.Vol. 1.
Wiley, Chichester, 1982.

Annexe A

Programmes Matlab

```
function c=conv_sp(a,b);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%   cyclical convolution of the vectors a &b
%%%   which must have the same length
%%%   using FFT
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

if length(a)~=length(b)
   error(' lenghts must be equal');
end

A=fft(a);
B=fft(b);
C=A.*B;
c=real(ifft(C));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [xrec,tx]=gold(y_d,G,NbIter,ty,x0);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Gold iterative method (M. Ben Slima)      %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
tx=ty;
G=fftshift(G);

[N,M]=size(G);
if N~=length(y_d), error('wrong input data, dimension do not agree');
end
if M==1
  CONV=1;   % convolution via fft
else
  CONV=0;   % convolution via matrix mutiplication
end

if CONV, VecN = flipud(conv_sp(flipud(y_d),G));
else,    VecN = G'*y_d; end
VecN = VecN .* (VecN > 0);

if nargin==4
  VecS = VecN;       % start value for iterations
else
  VecS=x0;
end
i=1;

while (NbIter>0)
  if CONV
    z = conv_sp(VecS,G);
    z = flipud(conv_sp(flipud(z),G));
  else
    z= G*VecS;
    z= G'*z;
  end
  VecS = VecS .* (VecN./(z));        % iterations Gold
%  plot(VecS);
```

```
  NbIter=NbIter-1;
%  i=i+1
end
xrec=VecS;
%disp('finished');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

conv01;
er=0;
nb_exp=1;
for kk=1:nb_exp
%for I=400:25:800
I=1000;
y_d=y1;G=g';ty=[0:127];x0=y_d;

[estime,tx]=gold(y_d,G,I,ty,x0);
er0=x1-estime;
er1=norm(er0)/norm(x1);
er=cat(1,er,er1);
end;
end;
disp('ERREUR QUADRATIQUE MOYEN RELATIF = ')
sum(er)/nb_exp
%-----------------------------------------------------------------------
figure(1);
plot(estime,'r');hold on
plot(x1);hold off

figure(2)
plot(400:25:800,er(2:length(er)))
beep

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%function [xrec,tx,i]=jansson_L(y_d,G,I,ty,method);
conv01;
%calib_m01_006; N=length(x);
calib_m01_02;y1=y;;g_02;g=cat(1,g,zeros(936,1));g=g';
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%   Jansson iterative method  with positivity        %
%%   constraints   (M. Ben Slima)                     %
%%                                                    %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
er=0;
nb_exp=1;
for kk=1:nb_exp
%for I=300:25:1500%2:5:500
I=150;
y_d=y1;G=g';ty=[0:N-1];method='L2';
%-----------------------------------------------------------------------
tx=ty;
G=fftshift(G);
[N,M]=size(G);
if N~=length(y_d), error('wrong input data, dimension do not agree');
end
```

```
if M==1
  CONV=1;
  x=flipud(conv_sp(flipud(y_d),G));
else
  CONV=0;
  x=G'*y_d;
end
x=zeros(size(y_d));
NbIter=I; Ifix=1;
z=0;z_old=0;z_old2=0;z_old3=0;NN=0;R=0; N2=0;
i=0;j=0;ir=0;

while (NbIter>0)
  if (CONV==1)
     dy = (y_d - conv_sp(x,G));
     gr = flipud(conv_sp(flipud(dy),G));
  else
     dy = (y_d - G*x);
     gr = G'*dy;
  end

  if ~Ifix
    NN_old=NN;
    NN=norm(x);
    R_old=R;  R=norm(dy);

    if i>1,
      z_old4=z_old3; z_old3=z_old2; z_old2=z_old;  z_old=z;
      z=log(NN/NN_old)/log(R/R_old);
    end
    if (z<z_old)&(z_old<z_old2)&(z_old2<z_old3)&(z_old3<-1)
      break
    end
  end

  N2_old=N2; N2=gr'*gr;

  if j==0, dd=gr;
  else, dd=gr+(N2/N2_old)*dd; end

  if any(x), d=dd.*x;
  else, d=dd; end

  if (d'*gr<0)
    ir=ir+1; j=-1;
    if any(x), d=gr.*x;
    else d=gr; end
  end

  if CONV, Gd=conv_sp(G,d);
  else, Gd=G*d; end
  lambda = dy'*Gd/(Gd'*Gd);

  x = x + lambda*d;
  NbIter = NbIter - 1;
  i=i+1;j=j+1;
  if (j==N), j=0; end
```

```
end
xrec=x;
%-------------------------------------------------------------------------
er0=x1-xrec;
er1=norm(er0)/norm(x1);
er=cat(1,er,er1);
end;
end;
disp('ERREUR QUADRATIQUE MOYEN RELATIF = ')
sum(er)/nb_exp
%-------------------------------------------------------------------------
chercher=0;
if chercher==1
pcc=[300:25:1500];%300:25:2000
Iopt=pcc(1);kk=2;er2=er(2);
for k=3:length(er)
    if er2>=er(k-1)
        kk=k-1;er2=er(k-1);
    end;
end;
Iopt=pcc(kk);  %pcc(k-1) correspond à er(k)
end;
%-------------------------------------------------------------------------
figure(1);
plot(xrec,'r');hold on
plot(x1);hold off

figure(2)
plot(300:25:1500,er(2:length(er)))
beep

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%            Filtre de Kalman                       %
%                                                   %
%                  Été 2002                         %
%                                                   %
%            Fahim  El abidi                        %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear all;clc;
er=0;
dt=1;          %Pas d'échantillonnage
%DEFINITION DE LA FONCTION DE TRANSFERT DU SYSTÈME DE MESURE:
sigma=8;
M=128;         % nombre de points d'echantillonage de g(t)
T1=dt*(M/2); % [T2,T1] est l'intervalle de g(t)
T2=dt*(M/2-1);
t=[-T2:dt:T1];
g=0.0525*exp(-(t).^2/2/sigma^2);

%DEFINITION DE LA GRANDEUR a MESURER x(t):
N=128;        % nombre de points d'echantillonage de x(t)
T=dt*(N-1);
t=[0:dt:T];
x=2*exp(-(t-45).^2/2/2.25^2)+6*exp(-(t-65).^2/2/2.25^2);
%x=1.*(t>=2);
x=x';
```

```
%MESURE BRUTE DU SYSTÈME DE MESURE
for hh=1:50
y=conv(g,x)*dt;%y=y';
%AJOUT DU BRIUT BLANC SUR LA SORTIE DU SYSTÈME
SNR=64.6; % RAPPORT SIGNAL/BRUIT
py=sqrt(sum(y.^2)/length(y));
s=py*10.^(-SNR/20);
v=randn(length(y),1);
v=v*s/std(v);
y=y+v;

% L'ETAT DU ½ SYSTÈME G+/- EST D'ORDRE 3:
de=6;          %dimention de l'état
z_int=zeros(de,length(y));

%DEFINITION DES MATRICE D'ÉTAT DU G+/-
a1=0.19261168004182;
a2=-0.19128919874023;
a3=0.02069599842750;
b1=0.13277647492450;
b2=0.43454750843267;
b3=2.81913305961804;

p1=exp(-b1*dt); p2=exp(-b2*dt); p3=exp(-b3*dt);
n3=a1+a2+a3;
n2=-(a1*p2+a1*p3+a2*p1+a2*p3+a3*p1+a3*p2);
n1=a1*p2*p3+a2*p1*p3+a3*p1*p2;
d0=-p1*p2*p3;
d1=p1*p2+p1*p3+p2*p3;
d2=-p1-p2-p3;

A=[-d2 -d1 -d0 n3 n2 n1; 1 0 0 0 0 0; 0 1 0 0 0 0; 0 0 0 1 0 0; 0 0 0 1
0 0; 0 0 0 0 1 0];
B=[0;0;0;1;0;0];
C=[-d2 -d1 -d0 n3 n2 n1];

K=zeros(de,1); %GAIN DU FILTRE
%gain=[0; 0; 0; 0; 0; 0];
%DEFINITION DES PARAMETRES DU FILTRE DE KALMAN

%for root_beta=0:0.1:100
beta=100;    % P0=beta*Matrice_Identité
p0=beta*eye(de);
p1=zeros(de);
p2=zeros(de);
root_beta=12.5;

ecar_type_nu=s;            %c'est l'écart type de nu
R=ecar_type_nu.^2;         %variance de nu
ecar_type_w=root_beta*s;   %c'est l'écart type de w
Q=diag((ecar_type_w)^2*ones(1,1));

%gain=zeros(de,1);
for k=1:30,
   p1=A*p0*A'+B*Q*B';
   K=p1*C'*inv(C*p1*C'+R);
```

```
   p2=(eye(de)-K*C)*p1;
   p0=p2;
   %gain=cat(2,gain,K);
end;

%CALCUL DU Z_int
for k=1:length(y)-1,
   z_int(:,k+1)=A*z_int(:,k)+K*(y(k+1)-C*A*z_int(:,k));
end;
%---------------z_int=symetrie(z_int)---------------------
z_int=fliplr(z_int);

z_estime=zeros(de,length(z_int)); %Estimé de l'état du système complet
for k=1:length(z_int)-1,
   z_estime(:,k+1)=A*z_estime(:,k)+K*(z_int(4,k+1)-C*A*z_estime(:,k));
end;

z_estime=fliplr(z_estime);
estime=z_estime(5,floor(M/2):length(z_int)-floor(M/2));
estime=estime';
%estime=estime.^6;x=x.^6;
erreur=x-estime;

%ERREUR RELATIF QUADRATIQUE MOYEN
er1=norm(erreur)/norm(x);
%disp('ERREUR QUADRATIQUE MOYEN RELATIF = '); disp(er1);
%disp('------------------');

er=cat(1,er,er1);
end;

figure(3);
plot(x);hold on;
plot(estime,'r');hold off;
title('X(bleur) ESTIMÉ(rouge)');
figure(1);
plot(z_int(5,:));
title('Z_INT');
figure(2);
plot(x);hold on;
plot(y(floor(M/2)+1:length(y)-floor(M/2)),'g');hold off;
title('X(bleur) Y(vert)');
figure(4);
plot(erreur);
title('ERREUR=x-estimé');

%figure(5)
%plot(0:0.1:100,er(2:size(er)),'k')
%title('ERREUR=f(beta)');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%    Filtre Hinf pour g non-gausoidale                  %
%                                                       %
%                Été 2002                               %
%                                                       %
%            Fahim  El abidi                            %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
clear all;
clc;

eqm0=1000;
M=64;
dt=25/128;
T1=dt*M;
gama=1.1;
n=128;
N=n;
z0=zeros(M,1);

max=N;
Ai=6;
a=0.7;
b=0.9;
al=exp(-a*dt);
bl=exp(-b*dt);
k=1;
%DEFINITION DE LA FONCTION DE TRANSFERT DU SYSTÈME DE MESURE:
for t=0:dt:T1,
   g(k)=Ai*(exp(-a*t)-exp(-b*t));
   k=k+1;
end;

%DEFINITION DE LA GRANDEUR À MESURER x(t):
t=0;
for k=1:N,
   x(k)=2*exp(-6*(t-5.5)^2)+6*exp(-6*(t-8)^2);
   t=t+dt;
end;
x=x';
yd=conv(g,x)*dt;
z_estime=zeros(3,max);

A=[al+bl -al*bl Ai*(al-bl); 1 0 0; 0 0 1];
B=[0; 0; 1];
C=[al+bl -al*bl Ai*(al-bl)];

L=1/dt*[0 0 0; 0 0 0; 0 0 1];
eqm=zeros(1);
Q=1e-6*eye(3);
p0=0.002*eye(3);
W=0.57e-4;
V=1.8e-10;%1.8e-4
Qb=L'*Q*L;

SNR=58.6;
ss=size(yd);
py=sqrt(sum(yd.^2)/ss(1));
s=py*10.^(-SNR/20);

for ii=1:1%50
v=randn(size(yd),1);
v=v*s/std(v);

P=0;%p0;
```

```
y=yd+v;
K=zeros(3,1);

for k=1:size(y)-1,

   K=A*P*inv(eye(3)-gama*Qb*P+C'*inv(V)*C*P)*C'*inv(V);
   z_estime(:,k+1)=A*z_estime(:,k)+K*(y(k)-C*z_estime(:,k));
   P2=A*P*A'+B*W*B'+A*P*(gama*Qb-C'*inv(V)*C)*inv(eye(3)-P*(gama*Qb-
C'*inv(V)*C))*P*A';
   P=P2;

   y_est(k)=C*z_estime(:,k);
end;

estime1=L*z_estime(:,3:size(y)-M+2);
estime=estime1(3,:);
%estime=L*z_estime;
erreur=x-estime';
er(ii)=sqrt(sum(erreur.^2)/sum(x.^2));
end;

figure(1);
plot(g);

figure(2);
plot(x);hold on;
plot(y,'g');

figure(7);
plot(x);hold on;
plot(estime,'r');

figure(4);
plot(erreur);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%    Filtre Hinf pour g gausoidale                       %
%                                                        %
%                  Été 2002                              %
%                                                        %
%            Fahim El abidi                              %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear all;clc;
er=0;

dt=1;          %Pas d'échantillonnage

%DEFINITION DE LA FONCTION DE TRANSFERT DU SYSTÈME DE MESURE:
sigma=8;
M=128;         % nombre de points d'echantillonage de g(t)
T1=dt*(M/2); % [T2,T1] est l'intervalle de g(t)
T2=dt*(M/2-1);
t=[-T2:dt:T1];
g=0.0525*exp(-(t).^2/2/sigma^2);

%DEFINITION DE LA GRANDEUR a MESURER x(t):
```

```
N=128;        % nombre de points d'echantillonage de x(t)
T=dt*(N-1);
t=[0:dt:T];
x=2*exp(-(t-45).^2/2/2.25^2)+6*exp(-(t-65).^2/2/2.25^2);
%x=1.*(t>=2);
x=x';

%DEFINITION DES MATRICE D'ÉTAT DU G+/-
a1=0.19261168004182;
a2=-0.19128919874023;
a3=0.02069599842750;
b1=0.13277647492450;
b2=0.43454750843267;
b3=2.81913305961804;
%a1=0.2209655; a2=-0.226789; a3=-0.010961;
%b1=0.1339261; b2=0.3720299; b3=5.8959598;
p1=exp(-b1*dt); p2=exp(-b2*dt); p3=exp(-b3*dt);
k1=(a1/b1)*(1-p1);k2=(a2/b2)*(1-p2);k3=(a3/b3)*(1-p3);
n2=k1+k2+k3;
n1=-(k1*p2+k1*p3+k2*p1+k2*p3+k3*p1+k3*p2);
n0=k1*p2*p3+k2*p1*p3+k3*p1*p2;
d2=-p1-p2-p3;
d1=p1*p2+p1*p3+p2*p3;
d0=-p1*p2*p3;

A=[-d2 -d1 -d0 n2 n1 n0; 1 0 0 0 0 0; 0 1 0 0 0 0; 0 0 0 1 0 0; 0 0 0 1
0 0; 0 0 0 0 1 0];
B=[0;0;0;1;0;0];
C=[-d2 -d1 -d0 n2 n1 n0];

%MESURE BRUTE DU SYSTÈME DE MESURE
er_cum=zeros(51,1);
er=0;
y=conv(g,x)*dt;%y=y';
%AJOUT DU BRIUT BLANC SUR LA SORTIE DU SYSTÈME
SNR=64.6; % RAPPORT SIGNAL/BRUIT
py=sqrt(sum(y.^2)/length(y));
s=py*10.^(-SNR/20);
v=randn(length(y),1);
v=v*s/std(v);
y=y+v;

% L'ETAT DU ½ SYSTÈME G+/- EST D'ORDRE 3:
de=6;         %dimention de l'état
z_int=zeros(de,length(y));

K=zeros(de,1); %GAIN DU FILTRE
%gain=[0; 0; 0; 0; 0; 0];
%DEFINITION DES PARAMETRES DU FILTRE H(inf)
gama=1.0089e4;%10.085e3 pour que les pic sont atteinds
L=eye(de);
Q=0.5e-4*eye(de);
Qb=L'*Q*L;
p0=0.00032145*eye(de);%0.001 % condition initiale =0 ou non
0.000232145
P=p0;
```

```
W=0.6098;%0.609711;%0.0061057;%augmenter / diminuer pic --- LISSAGE DU
courbe
V=1.25e-2;%0.76e-4; % augmanter /diminuer pic

%CALCUL DU GAIN DU FILTRE EN PREMIER LIEU
for k=1:30,
   K=A*P*inv(eye(de)-gama*Qb*P+C'*inv(V)*C*P)*C'*inv(V);
   P2=A*P*A'+B*W*B'+A*P*(gama*Qb-C'*inv(V)*C)*inv(eye(de)-P*(gama*Qb-
C'*inv(V)*C))*P*A';
   P=P2;
   %gain=cat(2,gain,K);
end;

%temp=fliplr(y');ym=temp';
%CALCUL DU Z_int
for k=1:length(y)-1,
   z_int(:,k+1)=A*z_int(:,k)+K*(y(k)-C*z_int(:,k));
end;

z_int=fliplr(z_int);

z_estime=zeros(de,length(z_int)); %Estimé de l'état du système complet

for k=1:length(z_int)-1,
   z_estime(:,k+1)=A*z_estime(:,k)+K*(z_int(5,k)-C*z_estime(:,k));
end;

Af=A-K*C; Bf=K; Cf=[0 0 0 0 1 0];

z_estime=fliplr(z_estime);
estime=z_estime(5,floor(M/2):length(z_int)-floor(M/2));
estime=estime';
erreur=x-estime;

%ERREUR RELATIF QUADRATIQUE MOYEN
er1=norm(erreur)/norm(x);
disp('ERREUR QUADRATIQUE MOYEN RELATIF = '); disp(er1);
disp('-----------------');

er=cat(1,er,er1);

figure(3);
title('X(bleur) ESTIMÉ(rouge)');
plot(x);hold on;
plot(estime,'r--');hold off;
figure(1);
title('Z_INT');
plot(z_int(5,:));
figure(2);
title('X(bleur) Y(vert)');
plot(x);hold on;
plot(y(floor(M/2)+1:length(y)-floor(M/2)),'g-.');hold off;
figure(4);
title('ERREUR=x-estimé');
plot(erreur);
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%    Filtre Hinf pour g non-gausoidale         %
%                                              %
%       avec un seul parametre Beta            %
%                                              %
%                  Été 2002                    %
%                                              %
%            Fahim El abidi                    %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear all;clc;
er=0;

dt=1;          %Pas d'échantillonnage
%DEFINITION DE LA FONCTION DE TRANSFERT DU SYSTÈME DE MESURE:
sigma=4;
M=128;         % nombre de points d'echantillonage de g(t)
T1=dt*(M/2); % [T2,T1] est l'intervalle de g(t)
T2=dt*(M/2-1);
t=[-T2:dt:T1];
g=0.07*exp(-(t).^2/2/sigma^2);

%DEFINITION DE LA GRANDEUR a MESURER x(t):
N=148;         % nombre de points d'echantillonage de x(t)
T=dt*(N-1);
t=[0:dt:T];
x=4*exp(-(t-45).^2/2/2.25^2)+6*exp(-(t-65).^2/2/1.5^2)+12*exp(-(t-
25).^2/2/2.25^2)+4*exp(-(t-85).^2/2/2.25^2)+7*exp(-(t-
95).^2/2/2.25^2)+10*exp(-(t-115).^2/2/1.25^2);
x=x';

%DEFINITION DES MATRICE D'ÉTAT DU G+/-
Ai=6;
a=0.7;
b=0.9;
al=exp(-a*dt);
bl=exp(-b*dt);

A=[al+bl -al*bl Ai*(al-bl); 1 0 0; 0 0 1];
B=[0; 0; 1];
C=[al+bl -al*bl Ai*(al-bl)];

%MESURE BRUTE DU SYSTÈME DE MESURE
er=0;nb_exp=1;
for hh=1:nb_exp
y=conv(g,x)*dt;%y=y';
%AJOUT DU BRIUT BLANC SUR LA SORTIE DU SYSTÈME
SNR=64.6; % RAPPORT SIGNAL/BRUIT
py=sqrt(sum(y.^2)/length(y));
s=py*10.^(-SNR/20);
v=randn(length(y),1);
v=v*s/std(v);
y=y+v;

% L'ETAT DU ½ SYSTÈME G+/- EST D'ORDRE 3:
de=3;          %dimention de l'état

gama=1.5e-4;%1.353e-6;
```

```
beta=0.0033;%0.0371;%0.0115;%0.004825;%0.001875;%0.003577;
i_beta=1/beta;

z_int=zeros(de,length(y));
K=zeros(de,1); %GAIN DU FILTRE

%CALCUL DU GAIN DU FILTRE EN PREMIER LIEU
P=zeros(de);
for k=1:30,
   K=i_beta*A*P*inv(eye(de)-gama*P+i_beta*C'*C*P)*C';
   P2=A*P*A'+B*B'+A*P*(gama*eye(de)-i_beta*C'*C)*inv(eye(de)-
P*(gama*eye(de)-i_beta*C'*C))*P*A';
   P=P2;
   %gain=cat(2,gain,K);
end;

%temp=fliplr(y');ym=temp';
%CALCUL DU Z_int
for k=1:length(y)-1,
   z_estime(:,k+1)=A*z_estime(:,k)+K*(y(k)-C*z_estime(:,k));
end;

estime=z_estime(2,floor(M/2):length(z_int)-floor(M/2));
estime=estime';
erreur=x-estime;

%ERREUR RELATIF QUADRATIQUE MOYEN
er1=norm(erreur)/norm(x);
%disp('ERREUR QUADRATIQUE MOYEN RELATIF = '); disp(er1);
%disp('----------------------');

er=cat(1,er,er1);
end;
end;
disp('ERREUR QUADRATIQUE MOYEN RELATIF = ')
sum(er)/nb_exp
disp('--------------------');

chercher=0;
if chercher==1
pcc=[0.02:0.0001:0.04];
Popt=pcc(1);kk=2;er2=er(2);
for k=3:length(er)
    if er2>=er(k-1)
        kk=k-1;er2=er(k-1);
    end;
end;
Popt=pcc(kk);  %pcc(k-1) correspond à er(k)
end;

figure(3);
plot(x);hold on;
plot(estime,'r--');hold off;
XLABEL('longueur d''onde')
YLABEL('niveau')

figure(1);
```

```
plot(z_int(2,:));
XLABEL('longueur d''onde')
YLABEL('niveau')

figure(2);
plot(x);hold on;
plot(y(floor(M/2)+1:length(y)-floor(M/2)),'g--');hold off;
XLABEL('longueur d''onde')
YLABEL('niveau')

figure(4);
plot(erreur);
XLABEL('longueur d''onde')
YLABEL('niveau')

figure(5);
plot(er(2:length(er)),'k');

figure(6);
plot(-T2:dt:T1,g);
XLABEL('longueur d''onde')
YLABEL('niveau')

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%    Filtre Hinf pour g gausoidale                   %
%                                                    %
%       avec un seul parametre Beta                  %
%                                                    %
%                  Été 2002                          %
%                                                    %
%            Fahim El abidi                          %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear all;clc;
er=0;

dt=1;          %Pas d'échantillonnage

%DEFINITION DE LA FONCTION DE TRANSFERT DU SYSTÈME DE MESURE:
sigma=4;
M=128;         % nombre de points d'echantillonage de g(t)
T1=dt*(M/2); % [T2,T1] est l'intervalle de g(t)
T2=dt*(M/2-1);
t=[-T2:dt:T1];
g=0.07*exp(-(t).^2/2/sigma^2);

%DEFINITION DE LA GRANDEUR a MESURER x(t):
N=148;        % nombre de points d'echantillonage de x(t)
T=dt*(N-1);
t=[0:dt:T];
x=4*exp(-(t-45).^2/2/2.25^2)+6*exp(-(t-65).^2/2/1.5^2)+12*exp(-(t-
25).^2/2/2.25^2)+4*exp(-(t-85).^2/2/2.25^2)+7*exp(-(t-
95).^2/2/2.25^2)+10*exp(-(t-115).^2/2/1.25^2);
x=x';

%DEFINITION DES MATRICE D'ÉTAT DU G+/-
a1=0.55935028722113;
a2=-0.65183425463092;
```

```
a3=0.12845413693272;
b1=0.29762584883795;
b2=0.60792790476436;
b3=7.49779435196885;
p1=exp(-b1*dt); p2=exp(-b2*dt); p3=exp(-b3*dt);
k1=(a1/b1)*(1-p1);k2=(a2/b2)*(1-p2);k3=(a3/b3)*(1-p3);
n2=k1+k2+k3;
n1=-(k1*p2+k1*p3+k2*p1+k2*p3+k3*p1+k3*p2);
n0=k1*p2*p3+k2*p1*p3+k3*p1*p2;
d2=-p1-p2-p3;
d1=p1*p2+p1*p3+p2*p3;
d0=-p1*p2*p3;

A=[-d2 -d1 -d0 n2 n1 n0; 1 0 0 0 0 0; 0 1 0 0 0 0; 0 0 0 1 0 0; 0 0 0 1
0 0; 0 0 0 0 1 0];
B=[0;0;0;1;0;0];
C=[-d2 -d1 -d0 n2 n1 n0];

%MESURE BRUTE DU SYSTÈME DE MESURE
er=0;nb_exp=1;
for hh=1:nb_exp
y=conv(g,x)*dt;%y=y';
%AJOUT DU BRIUT BLANC SUR LA SORTIE DU SYSTÈME
SNR=64.6; % RAPPORT SIGNAL/BRUIT
py=sqrt(sum(y.^2)/length(y));
s=py*10.^(-SNR/20);
v=randn(length(y),1);
v=v*s/std(v);
y=y+v;

% L'ETAT DU ½ SYSTÈME G+/- EST D'ORDRE 3:
de=6;          %dimention de l'état

%gain=[0; 0; 0; 0; 0; 0];
%DEFINITION DES PARAMETRES DU FILTRE H(inf)
%for gama=80000:1:90000;%1.0082e4:0.00001e4:1.015e4
%for beta=0.00025:0.001:0.2%0.02:0.0001:0.04%0.00001:0.0001:0.014
gama=1.5e-4;%1.353e-6;
beta=0.0033;%0.0371;%0.0115;%0.004825;%0.001875;%0.003577;
i_beta=1/beta;

z_int=zeros(de,length(y));
K=zeros(de,1); %GAIN DU FILTRE

%CALCUL DU GAIN DU FILTRE EN PREMIER LIEU
P=zeros(de);
for k=1:30,
   K=i_beta*A*P*inv(eye(de)-gama*P+i_beta*C'*C*P)*C';
   P2=A*P*A'+B*B'+A*P*(gama*eye(de)-i_beta*C'*C)*inv(eye(de)-
P*(gama*eye(de)-i_beta*C'*C))*P*A';
   P=P2;
   %gain=cat(2,gain,K);
end;

%temp=fliplr(y');ym=temp';
%CALCUL DU Z_int
for k=1:length(y)-1,
```

```
   z_int(:,k+1)=A*z_int(:,k)+K*(y(k)-C*z_int(:,k));
end;
%--------------z_int=symetrie(z_int)--------------
z_int=fliplr(z_int);

z_estime=zeros(de,length(z_int)); %Estimé de l'état du système complet

for k=1:length(z_int)-1,
   z_estime(:,k+1)=A*z_estime(:,k)+K*(z_int(5,k)-C*z_estime(:,k));
end;

z_estime=fliplr(z_estime);
estime=z_estime(5,floor(M/2):length(z_int)-floor(M/2));
estime=estime';
erreur=x-estime;

%ERREUR RELATIF QUADRATIQUE MOYEN
er1=norm(erreur)/norm(x);
%disp('ERREUR QUADRATIQUE MOYEN RELATIF = '); disp(er1);
%disp('----------------------------');

er=cat(1,er,er1);
end;
end;
disp('ERREUR QUADRATIQUE MOYEN RELATIF = ')
sum(er)/nb_exp
disp('--------------------------');

chercher=0;
if chercher==1
pcc=[0.02:0.0001:0.04];
Popt=pcc(1);kk=2;er2=er(2);
for k=3:length(er)
    if er2>=er(k-1)
        kk=k-1;er2=er(k-1);
    end;
end;
Popt=pcc(kk);  %pcc(k-1) correspond à er(k)
end;

figure(3);
plot(x);hold on;
plot(estime,'r--');hold off;
XLABEL('longueur d''onde')
YLABEL('niveau')

figure(1);
plot(z_int(5,:));
XLABEL('longueur d''onde')
YLABEL('niveau')

figure(2);
plot(x);hold on;
plot(y(floor(M/2)+1:length(y)-floor(M/2)),'g--');hold off;
XLABEL('longueur d''onde')
YLABEL('niveau')
```

```
figure(4);
plot(erreur);
XLABEL('longueur d''onde')
YLABEL('niveau')

figure(5);
plot(er(2:length(er)),'k');

figure(6);
plot(-T2:dt:T1,g);
XLABEL('longueur d''onde')
YLABEL('niveau')
```

Annexe B

Programmes Assembleur

```
          ;-----------------------------------------
          ; programme assembleur pour DSP 563x
          ; filtrage H inf
          ; utilisant la transformation: g = g+ * g-
          ; nombre d'echantillons est 64
          ; preparer par : Fahim  El abidi
          ; Genie Electrique Automne 2002
          ;-----------------------------------------


     org  p:$c00000
M    equ 6
N    equ 64
M_M  equ M*M ; dimension de G = M*M
     move #$0,R0
     move #$4,N0
     move #M-1,M0
     move #$100,R1
     move #N-1,M1
     move #$200,R3
     move #M-1,M3

     move #$100,R4
     move #M_M-1,M4
     move #$200,R5
     move #M-1,M5
     move #$0,R7
     move #N-1,M7
     move #N-1,R6
     move #N-1,M6

label9    do   #2,end9

     CLR  A
label2    rep  #M
     move A,X:(R0)+
label8    do   #N,end8
label3    move X:(R1)+,X1


label5    do   #M,end5
     move X:(R3)+,Y1
     mpy  Y1,X1,B        ;calcul de K*yk  ;move   A,Y:(R5)
     nop            ;---------------------------
```

```
label6    do   #M,end6
     move X:(R0)+,X0     Y:(R4)+,Y0
     mpy  X0,Y0,A        ;X:(R0)+,X0    Y:(R4)+,Y0
     lsl  A
     lsl  A
     lsl  A
     lsl  A
end6 nop
     add  B,A  ;B contient R5
     nop
     move A,Y:(R5)+
end5 nop

label7    do   #M,end7
     move Y:(R5)+,A
     nop
     move A,X:(R0)+
end7 nop
     move X:(R0+N0),A
     nop
     move A,Y:(R7)+
end8 nop            ; N echantillons

label1    do   #N,end1
     move Y:(R6)-,A
     nop
     move A,X:(R1)+
end1 nop
     move #$7f,R6

end9 nop       ; les 2 boucle pour g+ et g-
     stop
```

Annexe C

Programmes VHDL et design du processeur sur HDL Designer-Pro
de Mentor Graphics

```
;-----------------------------------------
;-- programmes VHDL pour filtrage H inf
;-- utilisant la transformation: g = g+ * g-
;--
;-- preparer par : Fahim  El abidi
;-- Genie Electrique Automne 2002
;-----------------------------------------


library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;

entity seq4 is
port(
     h: in std_logic;
     d: in std_logic_vector(15 downto 0);
     q: out std_logic_vector(15 downto 0)
     );
end seq4;

architecture archi of seq4 is

signal test: std_logic_vector(15 downto 0);
begin
     process
     begin
          wait until h='1'and h'event;
          test<=d;
     end process;
     q<=test;
end archi;


library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;

entity seq4bis is
port(
     h: in std_logic;
     d: in std_logic_vector(31 downto 0);
     q: out std_logic_vector(31 downto 0)
     );
end seq4bis;

architecture archi of seq4bis is

signal test: std_logic_vector(31 downto 0);--
:="00000000000000000000000000000000";
begin
     process
```

```
      begin
            wait until h='1' and h'event;
            test<=d;
      end process;
      q<=test;
end archi;


library ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_arith.all;

ENTITY addition2 IS
PORT (
      x : IN std_logic_vector (31 DOWNTO 0);-- pour le test 2 downto 0
      clk : in std_logic;
      a : OUT std_logic_vector (31 DOWNTO 0) -- pour le test 2 downto 0
      );
END addition2;

ARCHITECTURE archi OF addition2 IS

signal s, sa : std_logic_vector(31 downto
0):="00000000000000000000000000000000";

BEGIN
som : process (x, sa)
begin
      s <= (signed(x) + signed(sa));
end process som;

latch : process (clk)
begin
    if (clk='1' and clk'event) then
            sa <= s;
      end if;
end process latch;
a <= sa;
end archi;


library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;

entity addi1 is
port( a, b, ri: in std_logic;
      s, r: out std_logic
      );
end addi1;

architecture flot1 of addi1 is
```

```
signal sig: std_logic;
begin
      sig<=a xor b;
      s<= sig xor ri;
      r<= (a and b) or (sig and ri);
end flot1;


library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;

use work.addi1;

entity addi_n is
generic (n: natural :=32); -- pour le test:n=3
port (
      x, y: in std_logic_vector(n downto 1);
      rin: in std_logic:='0';
      sn: out std_logic_vector(n downto 1);
      rn: out std_logic
      );
end addi_n;

architecture struc_n of addi_n is

component addi1 is
port( a, b, ri: in std_logic;
      s, r: out std_logic
      );
end component;

signal c: std_logic_vector(1 to n+1);--initialisaion!

begin
      label1: for i in 1 to n generate
            instance: addi1 port map (x(i), y(i), c(i), sn(i), c(i+1));
      end generate;
      c(1)<=rin;
      rn<=c(n+1);
end struc_n;


library ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_arith.all;

ENTITY addition IS
PORT (
      x : IN std_logic_vector (31 DOWNTO 0);-- pour le test 2 downto 0
      clk : in std_logic;
      a : OUT std_logic_vector (31 DOWNTO 0) -- pour le test 2 downto 0
```

```
      );
END addition;

ARCHITECTURE archi OF addition IS

signal s, sa, xx : std_logic_vector(31 downto
0):="00000000000000000000000000001000";

BEGIN
som : process
begin
      s <= (signed(x) + signed(sa)); assert false report "addition
effectuer" severity error;
      xx <= x;
      wait on x, sa;
end process som;

latch : process (clk)
begin
    if (clk='1') then
          sa <= s;
      end if;
end process latch;
a <= sa;
end archi;


library ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_arith.all;

ENTITY multiplication IS
PORT (
      x,y :IN std_logic_vector (15 DOWNTO 0);-- pour le test 2 downto 0
      p :OUT std_logic_vector (31 DOWNTO 0) -- pour le test 5 downto 0
      );
END multiplication;

ARCHITECTURE archi OF multiplication IS
BEGIN
process (x,y)
VARIABLE dtemp : std_logic_vector(31 DOWNTO 0); --5 downto 0
BEGIN
      dtemp := (signed(x) * signed(y));
    p <= dtemp(31 downto 0); -- 5 downto 0
end process;
end archi;


library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
```

```
package pack is
     type LIGNE is array (0 to 5) of std_logic_vector(15 downto 0);
     type MAT is array (0 to 10, 0 to 5) of std_logic_vector(15 downto
0);
end pack;


library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use work.all;

entity pe2 is
     port (
          e, xi : in std_logic_vector(15 downto 0):="0000000000000000";
          clk : in std_logic;
          s : out std_logic_vector(15 downto 0);
          xo : out std_logic_vector(15 downto 0)
     );
end pe2;

architecture archi of pe2 is

component multiplication is
PORT (
     x,y :IN std_logic_vector (15 DOWNTO 0);
     p :OUT std_logic_vector (31 DOWNTO 0)
     );
end component;

component addition2 is
PORT (
     x : IN std_logic_vector (31 DOWNTO 0);-- pour le test 2 downto 0
     clk : in std_logic;
     a : OUT std_logic_vector (31 DOWNTO 0) -- pour le test 2 downto 0
     );
end component;

component seq4 is -- registre 16 bits
port(
     h: in std_logic;
     d: in std_logic_vector(15 downto 0);
     q: out std_logic_vector(15 downto 0)
     );
end component;

signal xi1 :std_logic_vector(15 downto 0);
signal sm, accum :std_logic_vector(31 downto 0);
signal ss :std_logic_vector(15 downto 0):="0000000000000000";
```

```
begin
      xi1 <= xi after 2 ns;
      label1: seq4 port map (clk, xi1, xo);
      label2: multiplication port map (xi1, e, sm);
      label3: addition2 port map (sm, clk, accum);
      ss <= accum(15 downto 0); -- attention on doit prendre les 20 bits
de poid fort!!!
      s <= ss; -- ceci est pour lasimulation
end archi;


library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use work.pack.all;

entity mem is
port (
      clk : in std_logic;
      s_mem0,
      s_mem1,
      s_mem2,
      s_mem3,
      s_mem4,
      s_mem5 : out std_logic_vector(15 downto 0)
      );
end mem;

architecture archi of mem is
begin
process (clk)
variable adr : integer range 0 to 10 :=0;
variable m : MAT := (
            ("0000000000000011", "0000000000000000", "0000000000000000",
"0000000000000000", "0000000000000000", "0000000000000000"),
            ("0000000000000000", "0000000000000000", "0000000000000000",
"0000000000000000", "0000000000000000", "0000000000000000"),
            ("0000000000000000", "0000000000000001", "0000000000000000",
"0000000000000000", "0000000000000000", "0000000000000000"),
            ("0000000000000000", "0000000000000000", "0000000000000000",
"0000000000000000", "0000000000000000", "0000000000000000"),
            ("0000000000000000", "0000000000000000", "0000000000000010",
"0000000000000000", "0000000000000000", "0000000000000000"),
            ("0000000000000000", "0000000000000000", "0000000000000000",
"0000000000000000", "0000000000000000", "0000000000000000"),
            ("0000000000000000", "0000000000000000", "0000000000000000",
"0000000000000110", "0000000000000000", "0000000000000000"),
            ("0000000000000000", "0000000000000000", "0000000000000000",
"0000000000000000", "0000000000000000", "0000000000000000"),
            ("0000000000000000", "0000000000000000", "0000000000000000",
"0000000000000000", "0000000000000100", "0000000000000000"),
```

```
          ("0000000000000000", "0000000000000000", "0000000000000000",
"0000000000000000", "0000000000000000", "0000000000000000"),
          ("0000000000000000", "0000000000000000", "0000000000000000",
"0000000000000000", "0000000000000000", "0000000000000101")
                         );
begin
     if (clk = '1') then
          s_mem0 <= m(adr, 0);
          s_mem1 <= m(adr, 1);
          s_mem2 <= m(adr, 2);
          s_mem3 <= m(adr, 3);
          s_mem4 <= m(adr, 4);
          s_mem5 <= m(adr, 5);
          adr := adr +1;
          if (adr = 11) then
               adr := 0;
          end if;
     end if;
end process;
end archi;


library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use work.pe;
use work.pack.all;

entity reseau is
port (
     horloge : in std_logic;
     etat_pr : in std_logic_vector(15 downto 0);
     entree1 :  in std_logic_vector(15 downto 0);
     entree2 :  in std_logic_vector(15 downto 0);
     entree3 :  in std_logic_vector(15 downto 0);
     entree4 :  in std_logic_vector(15 downto 0);
     entree5 :  in std_logic_vector(15 downto 0);
     entree6 :  in std_logic_vector(15 downto 0);
     etat_sv1 : in std_logic_vector(15 downto 0);
     etat_sv2 : in std_logic_vector(15 downto 0);
     etat_sv3 : in std_logic_vector(15 downto 0);
     etat_sv4 : in std_logic_vector(15 downto 0);
     etat_sv5 : in std_logic_vector(15 downto 0);
     etat_sv6 : in std_logic_vector(15 downto 0)
     );
end reseau;

architecture arch of reseau is

component pe is
     port (
          e, xi : in std_logic_vector(15 downto 0):="0000000000000000";
```

```
            clk : in std_logic;
            s : out std_logic_vector(15 downto 0);
            xo : out std_logic_vector(15 downto 0)
      );
end component;

signal c2, c3, c4, c5, c6, c7 : std_logic_vector(15 downto
0):="0000000000000000";
signal c22, c33, c44, c55, c66 : std_logic_vector(15 downto
0):="0000000000000000";
signal clk1, clk2, clk3, clk4, clk5, clk6 : std_logic;

begin

      clk1 <= horloge;
      clk2 <= horloge;
      clk3 <= horloge;
      clk4 <= horloge;
      clk5 <= horloge;
      clk6 <= horloge;
      label1 : pe port map ( entree1, etat_pr, clk1, etat_sv1, c2 );
      c22 <= transport c2 after 2 ns;
      label2 : pe port map ( entree2, c22, clk2, etat_sv2, c3 );
      c33 <= c3;-- after 2 ns;
      label3 : pe port map ( entree3, c33, clk3, etat_sv3, c4 );
      c44 <= c4;-- after 2 ns;
      label4 : pe port map ( entree4, c44, clk4, etat_sv4, c5 );
      c55 <= c5;-- after 2 ns;
      label5 : pe port map ( entree5, c55, clk5, etat_sv5, c6 );
      c66 <= c6;-- after 2 ns;
      label6 : pe port map ( entree6, c66, clk6, etat_sv6, c7 );

end arch;


library ieee;
use ieee.std_logic_1164.all;
-- use ieee.numeric_std.all;      -- Note: uncomment this if you use
                                  -- IEEE standard signed or unsigned
types.
-- use ieee.std_logic_arith.all; -- Note: uncomment/modify this if you
use
                                  -- Synopsys signed or unsigned types.
use std.textio.all;
use work.seq4;

entity t_reg is
end t_reg;

architecture stimulus of t_reg is

component seq4 is
```

```
port(
     h: in std_logic;
     d: in std_logic_vector(15 downto 0);
     q: out std_logic_vector(15 downto 0)
     );
end component;

constant PERIOD: time := 4 us;
signal d, q : std_logic_vector(15 downto 0);
signal h : std_logic;
signal done: boolean := false;

begin
    ABC: seq4 port map (h, d, q );


    CLOCK1: process
     variable clktmp: std_ulogic := '1';
    begin
        clktmp := not clktmp;
        h <= clktmp;
           if done = true then
           wait;
        end if;
         wait for PERIOD/2;
    end process CLOCK1;

    STIMULUS1: process
    begin
          d<="0000000000000000"; wait for 7 us;
          d<="0000000000000111"; wait for 10 us;
          d<="0000000000000011"; wait for 10 us;
          d<="0000000000001010"; wait for 6 us;

        done <= true;     -- Turn off the clock
        wait;             -- Suspend simulation
    end process STIMULUS1;

end stimulus;


library ieee;
use ieee.std_logic_1164.all;
-- use ieee.numeric_std.all;      -- Note: uncomment this if you use
                                  -- IEEE standard signed or unsigned
types.
-- use ieee.std_logic_arith.all; -- Note: uncomment/modify this if you
use
                                  -- Synopsys signed or unsigned types.
use std.textio.all;
use work.seq4bis;
```

```
entity t_regbis is
end t_regbis;

architecture stimulus of t_regbis is

component seq4bis is
port(
     h: in std_logic;
     d: in std_logic_vector(31 downto 0);
     q: out std_logic_vector(31 downto 0)
     );
end component;

constant PERIOD: time := 4 us;
signal d, q : std_logic_vector(31 downto 0);
signal h : std_logic;
signal done: boolean := false;

begin
    ABC: seq4bis port map (h, d, q );


    CLOCK1: process
     variable clktmp: std_ulogic := '1';
    begin
        clktmp := not clktmp;
        h <= clktmp;
           if done = true then
           wait;
        end if;
         wait for PERIOD/2;
    end process CLOCK1;


    STIMULUS1: process
    begin
           d<="00000000000000000000000000000000"; wait for 7 us;
           d<="00000000000000000000000000001111"; wait for 10 us;
           d<="00000000000000000000000000001011"; wait for 10 us;
           d<="00000000000000000000000000001000"; wait for 6 us;

        done <= true;     -- Turn off the clock
        wait;             -- Suspend simulation
    end process STIMULUS1;

end stimulus;


library ieee;
use ieee.std_logic_1164.all;
-- use ieee.numeric_std.all;      -- Note: uncomment this if you use
```

```
                                    -- IEEE standard signed or unsigned
types.
-- use ieee.std_logic_arith.all; -- Note: uncomment/modify this if you
use
                                    -- Synopsys signed or unsigned types.
use std.textio.all;
use work.addi1;

entity t_additionneur1bit is
end t_additionneur1bit;

architecture stimulus of t_additionneur1bit is

component addi1 is
port( a, b, ri: in std_logic;
      s, r: out std_logic
      );
end component;

signal a, b, ri, s, r : std_logic;

begin
    ABC: addi1 port map (a, b, ri, s, r);

    STIMULUS1: process
    begin
           a<='0'; b<='0'; ri<='0'; wait for 2 us;
           a<='0'; b<='1'; ri<='0'; wait for 2 us;
           a<='1'; b<='0'; ri<='0'; wait for 2 us;
           a<='1'; b<='1'; ri<='0'; wait for 2 us;
           a<='0'; b<='0'; ri<='1'; wait for 2 us;
           a<='0'; b<='1'; ri<='1'; wait for 2 us;
           a<='1'; b<='0'; ri<='1'; wait for 2 us;
           a<='1'; b<='1'; ri<='1'; wait for 2 us;

        wait;              -- Suspend simulation
    end process STIMULUS1;

end stimulus;


library ieee;
use ieee.std_logic_1164.all;
-- use ieee.numeric_std.all;      -- Note: uncomment this if you use
                                    -- IEEE standard signed or unsigned
types.
-- use ieee.std_logic_arith.all; -- Note: uncomment/modify this if you
use
                                    -- Synopsys signed or unsigned types.
use std.textio.all;
use work.addi_n;
```

```
entity t_add20bit is
end t_add20bit;

architecture stimulus of t_add20bit is

component addi_n is
generic (n: natural);
port (
      x, y: in std_logic_vector(n downto 1);
      rin: in std_logic:='0';
      sn: out std_logic_vector(n downto 1);
      rn: out std_logic
      );
end component;

signal x, y, sn: std_logic_vector(2 downto 0);
signal rin, rn: std_logic:='0';

begin
    ABC: addi_n generic map (3) port map (x, y, rin, sn, rn);

    STIMULUS1: process
    begin
          x<="000"; y<="000"; wait for 2 us;
          x<="000"; y<="001"; wait for 2 us;
          x<="000"; y<="010"; wait for 2 us;
          x<="000"; y<="011"; wait for 2 us;
          x<="000"; y<="100"; wait for 2 us;
          x<="000"; y<="101"; wait for 2 us;
          x<="000"; y<="110"; wait for 2 us;
          x<="000"; y<="111"; wait for 2 us;

          x<="001"; y<="000"; wait for 2 us;
          x<="001"; y<="001"; wait for 2 us;
          x<="001"; y<="010"; wait for 2 us;
          x<="001"; y<="011"; wait for 2 us;
          x<="001"; y<="100"; wait for 2 us;
          x<="001"; y<="101"; wait for 2 us;
          x<="001"; y<="110"; wait for 2 us;
          x<="001"; y<="111"; wait for 2 us;

          x<="010"; y<="000"; wait for 2 us;
          x<="010"; y<="001"; wait for 2 us;
          x<="010"; y<="010"; wait for 2 us;
          x<="010"; y<="011"; wait for 2 us;
          x<="010"; y<="100"; wait for 2 us;
          x<="010"; y<="101"; wait for 2 us;
          x<="010"; y<="110"; wait for 2 us;
          x<="010"; y<="111"; wait for 2 us;

          x<="011"; y<="000"; wait for 2 us;
          x<="011"; y<="001"; wait for 2 us;
```

```
            x<="011"; y<="010"; wait for 2 us;
            x<="011"; y<="011"; wait for 2 us;
            x<="011"; y<="100"; wait for 2 us;
            x<="011"; y<="101"; wait for 2 us;
            x<="011"; y<="110"; wait for 2 us;
            x<="011"; y<="111"; wait for 2 us;

            x<="100"; y<="000"; wait for 2 us;
            x<="100"; y<="001"; wait for 2 us;
            x<="100"; y<="010"; wait for 2 us;
            x<="100"; y<="011"; wait for 2 us;
            x<="100"; y<="100"; wait for 2 us;
            x<="100"; y<="101"; wait for 2 us;
            x<="100"; y<="110"; wait for 2 us;
            x<="100"; y<="111"; wait for 2 us;

            x<="101"; y<="000"; wait for 2 us;
            x<="101"; y<="001"; wait for 2 us;
            x<="101"; y<="010"; wait for 2 us;
            x<="101"; y<="011"; wait for 2 us;
            x<="101"; y<="100"; wait for 2 us;
            x<="101"; y<="101"; wait for 2 us;
            x<="101"; y<="110"; wait for 2 us;
            x<="101"; y<="111"; wait for 2 us;

            x<="110"; y<="000"; wait for 2 us;
            x<="110"; y<="001"; wait for 2 us;
            x<="110"; y<="010"; wait for 2 us;
            x<="110"; y<="011"; wait for 2 us;
            x<="110"; y<="100"; wait for 2 us;
            x<="110"; y<="101"; wait for 2 us;
            x<="110"; y<="110"; wait for 2 us;
            x<="110"; y<="111"; wait for 2 us;

            x<="111"; y<="000"; wait for 2 us;
            x<="111"; y<="001"; wait for 2 us;
            x<="111"; y<="010"; wait for 2 us;
            x<="111"; y<="011"; wait for 2 us;
            x<="111"; y<="100"; wait for 2 us;
            x<="111"; y<="101"; wait for 2 us;
            x<="111"; y<="110"; wait for 2 us;
            x<="111"; y<="111"; wait for 2 us;


        wait;               -- Suspend simulation
    end process STIMULUS1;

end stimulus;


library ieee;
use ieee.std_logic_1164.all;
```

```
-- use ieee.numeric_std.all;      -- Note: uncomment this if you use
                                  -- IEEE standard signed or unsigned
types.
use ieee.std_logic_arith.all;     -- Note: uncomment/modify this if you
use
                                  -- Synopsys signed or unsigned types.
use std.textio.all;
use work.addition;

entity t_add_bis is
end t_add_bis;

architecture stimulus of t_add_bis is

component addition is
PORT (
     x : IN std_logic_vector (31 DOWNTO 0);-- pour le test 2 downto 0
     clk : in std_logic;
     a : OUT std_logic_vector (31 DOWNTO 0) -- pour le test 2 downto 0
     );
end component;

signal x, a : std_logic_vector(31 downto 0);
constant PERIOD: time := 2 us;
signal h : std_logic;
signal done: boolean := false;

begin
    ABC: addition port map (x, h, a);

     CLOCK1: process
     variable clktmp: std_ulogic := '1';
     begin
        clktmp := not clktmp;
        h <= clktmp;
        if done = true then
         wait;
        end if;
        wait for PERIOD/2;
     end process CLOCK1;

    STIMULUS1: process
    begin
          x<="00000000000000000000000000000000"; wait for 5 us;
          x<="00000000000000000000000000000001"; wait for 4.003 us;
          x<="00000000000000000000000000000000"; wait for 6 us;


          done <= true;     -- Turn off the clock
          wait;             -- Suspend simulation
    end process STIMULUS1;
```

```
end stimulus;


library ieee;
use ieee.std_logic_1164.all;
-- use ieee.numeric_std.all;      -- Note: uncomment this if you use
                                  -- IEEE standard signed or unsigned
types.
use ieee.std_logic_arith.all;     -- Note: uncomment/modify this if you
use
                                  -- Synopsys signed or unsigned types.
use std.textio.all;
use work.multiplication;

entity t_multip is
end t_multip;

architecture stimulus of t_multip is

component multiplication IS
PORT (
     x,y :IN std_logic_vector (2 DOWNTO 0);
     p :OUT std_logic_vector (5 DOWNTO 0)
     );
end component;

signal x, y : std_logic_vector(2 downto 0);
signal p : std_logic_vector(5 downto 0);

begin
    ABC: multiplication port map (x, y, p);

    STIMULUS1: process
    begin
           x<="000"; y<="000"; wait for 2 us;
           x<="000"; y<="001"; wait for 2 us;
           x<="000"; y<="010"; wait for 2 us;
           x<="000"; y<="011"; wait for 2 us;
           x<="000"; y<="100"; wait for 2 us;
           x<="000"; y<="101"; wait for 2 us;
           x<="000"; y<="110"; wait for 2 us;
           x<="000"; y<="111"; wait for 2 us;

           x<="001"; y<="000"; wait for 2 us;
           x<="001"; y<="001"; wait for 2 us;
           x<="001"; y<="010"; wait for 2 us;
           x<="001"; y<="011"; wait for 2 us;
           x<="001"; y<="100"; wait for 2 us;

           x<="001"; y<="101"; wait for 2 us;
           x<="001"; y<="110"; wait for 2 us;
           x<="001"; y<="111"; wait for 2 us;
```

```
x<="010"; y<="000"; wait for 2 us;
x<="010"; y<="001"; wait for 2 us;
x<="010"; y<="010"; wait for 2 us;
x<="010"; y<="011"; wait for 2 us;
x<="010"; y<="100"; wait for 2 us;
x<="010"; y<="101"; wait for 2 us;
x<="010"; y<="110"; wait for 2 us;
x<="010"; y<="111"; wait for 2 us;

x<="011"; y<="000"; wait for 2 us;
x<="011"; y<="001"; wait for 2 us;
x<="011"; y<="010"; wait for 2 us;
x<="011"; y<="011"; wait for 2 us;
x<="011"; y<="100"; wait for 2 us;
x<="011"; y<="101"; wait for 2 us;
x<="011"; y<="110"; wait for 2 us;
x<="011"; y<="111"; wait for 2 us;

x<="100"; y<="000"; wait for 2 us;
x<="100"; y<="001"; wait for 2 us;
x<="100"; y<="010"; wait for 2 us;
x<="100"; y<="011"; wait for 2 us;
x<="100"; y<="100"; wait for 2 us;
x<="100"; y<="101"; wait for 2 us;
x<="100"; y<="110"; wait for 2 us;
x<="100"; y<="111"; wait for 2 us;

x<="101"; y<="000"; wait for 2 us;
x<="101"; y<="001"; wait for 2 us;
x<="101"; y<="010"; wait for 2 us;
x<="101"; y<="011"; wait for 2 us;
x<="101"; y<="100"; wait for 2 us;
x<="101"; y<="101"; wait for 2 us;
x<="101"; y<="110"; wait for 2 us;
x<="101"; y<="111"; wait for 2 us;

x<="110"; y<="000"; wait for 2 us;
x<="110"; y<="001"; wait for 2 us;
x<="110"; y<="010"; wait for 2 us;
x<="110"; y<="011"; wait for 2 us;
x<="110"; y<="100"; wait for 2 us;
x<="110"; y<="101"; wait for 2 us;
x<="110"; y<="110"; wait for 2 us;
x<="110"; y<="111"; wait for 2 us;

x<="111"; y<="000"; wait for 2 us;
x<="111"; y<="001"; wait for 2 us;
x<="111"; y<="010"; wait for 2 us;
x<="111"; y<="011"; wait for 2 us;
x<="111"; y<="100"; wait for 2 us;
x<="111"; y<="101"; wait for 2 us;
```

```
            x<="111"; y<="110"; wait for 2 us;
            x<="111"; y<="111"; wait for 2 us;


        wait;              -- Suspend simulation
    end process STIMULUS1;

end stimulus;


library ieee;
use ieee.std_logic_1164.all;
-- use ieee.numeric_std.all;      -- Note: uncomment this if you use
                                  -- IEEE standard signed or unsigned
types.
-- use ieee.std_logic_arith.all; -- Note: uncomment/modify this if you
use
                                  -- Synopsys signed or unsigned types.
use std.textio.all;
use work.pe;

entity t_pe is
end t_pe;

architecture stimulus of t_pe is

component pe is
     port (
          e, xi : in std_logic_vector(15 downto 0):="0000000000000000";
          clk : in std_logic;
          s : out std_logic_vector(15 downto 0);
          xo : out std_logic_vector(15 downto 0)
     );
end component;

constant PERIOD: time := 4 us;
signal e, xi : std_logic_vector(15 downto 0);
signal xo, s : std_logic_vector(15 downto 0);
signal h : std_logic;
signal done: boolean := false;

begin
    ABC: pe port map (e, xi, h, s, xo );


    CLOCK1: process
     variable clktmp: std_ulogic := '1';
    begin
        clktmp := not clktmp;
        h <= clktmp;
           if done = true then
           wait;
```

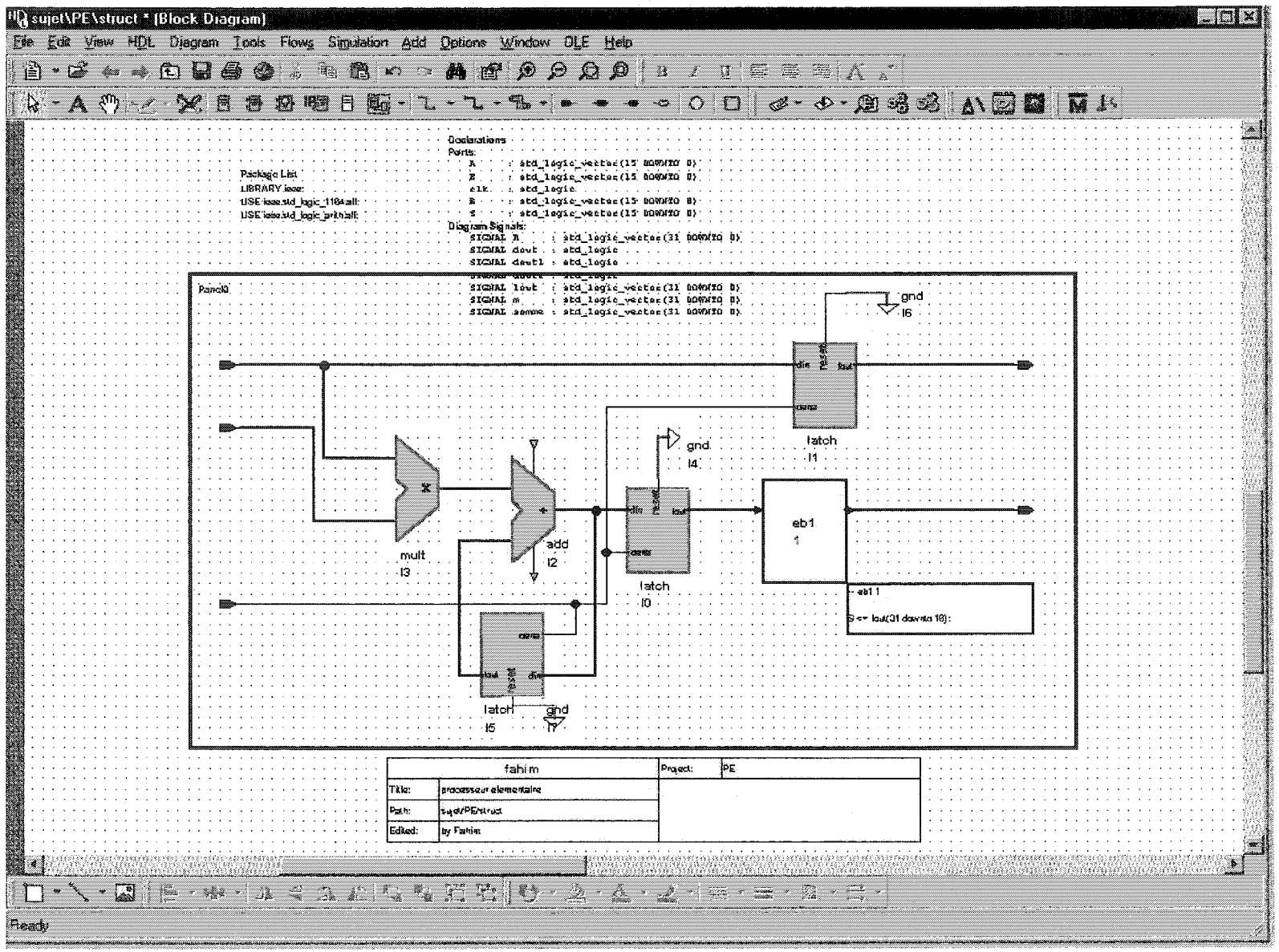
```
        end if;
         wait for PERIOD/2;
    end process CLOCK1;

    STIMULUS1: process
    begin
           e<="0000000000000010"; xi<="0000000000000001"; wait for 2 us;
           xi<="0000000000000010"; wait for 4 us;
           xi<="0000000000000011"; wait for 4 us;
           xi<="0000000000000100"; wait for 4 us;

        done <= true;     -- Turn off the clock
        wait;             -- Suspend simulation
    end process STIMULUS1;

end stimulus;


library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;      -- Note: uncomment this if you use
                               -- IEEE standard signed or unsigned
types.
use ieee.std_logic_arith.all; -- Note: uncomment/modify this if you use
                               -- Synopsys signed or unsigned types.
use std.textio.all;
use work.mem;

entity t_m is
end t_m;

architecture stimulus of t_m is

component mem is
port (
     clk : in std_logic;
     s_mem0,
     s_mem1,
     s_mem2,
     s_mem3,
     s_mem4,
     s_mem5 : out std_logic_vector(15 downto 0)
     );
end component;

constant PERIOD: time := 4 us;
signal s_mem0,  s_mem1,    s_mem2, s_mem3, s_mem4, s_mem5 :
std_logic_vector(15 downto 0);
signal h : std_logic;
signal done: boolean := false;

begin
```

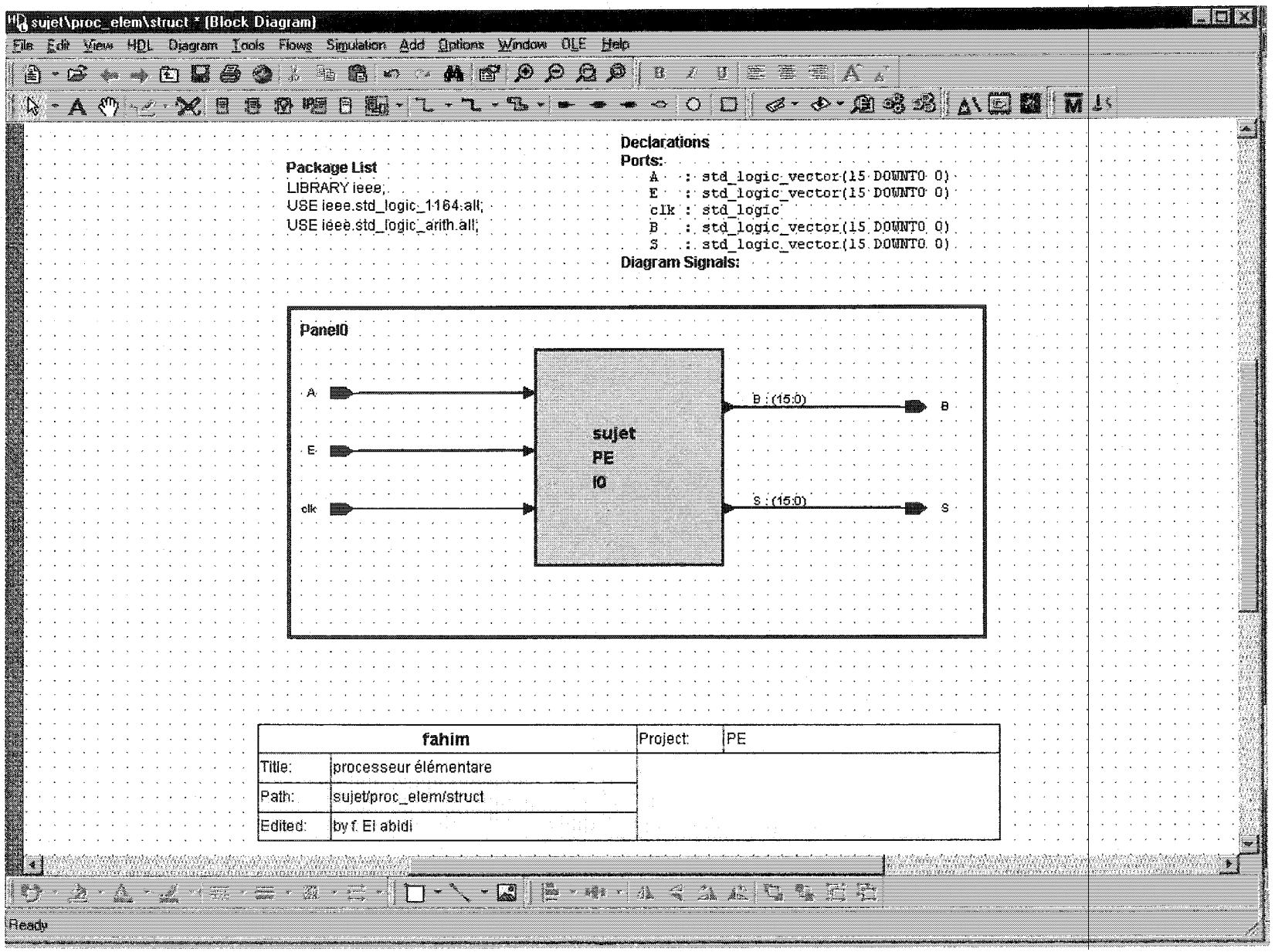
```
    ABC: mem port map (h, s_mem0,     s_mem1,    s_mem2, s_mem3, s_mem4,
s_mem5);


    CLOCK1: process
     variable clktmp: std_ulogic := '1';
    begin
        clktmp := not clktmp;
        h <= clktmp;
         wait for PERIOD/2;
    end process CLOCK1;

    STIMULUS1: process
    begin
          wait for 16 us;

        done <= true;     -- Turn off the clock
        wait;             -- Suspend simulation
    end process STIMULUS1;


end stimulus;


library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use work.reseau;
use work.pack.all;

entity t_res is
end t_res;

architecture stim of t_res is

component reseau is
port (
     horloge : in std_logic;
     etat_pr : in std_logic_vector(15 downto 0);
     entree1 :  in std_logic_vector(15 downto 0);
     entree2 :  in std_logic_vector(15 downto 0);
     entree3 :  in std_logic_vector(15 downto 0);
     entree4 :  in std_logic_vector(15 downto 0);
     entree5 :  in std_logic_vector(15 downto 0);
     entree6 :  in std_logic_vector(15 downto 0);
     etat_sv1 : in std_logic_vector(15 downto 0);
     etat_sv2 : in std_logic_vector(15 downto 0);
     etat_sv3 : in std_logic_vector(15 downto 0);
     etat_sv4 : in std_logic_vector(15 downto 0);
     etat_sv5 : in std_logic_vector(15 downto 0);
     etat_sv6 : in std_logic_vector(15 downto 0)
     );
```

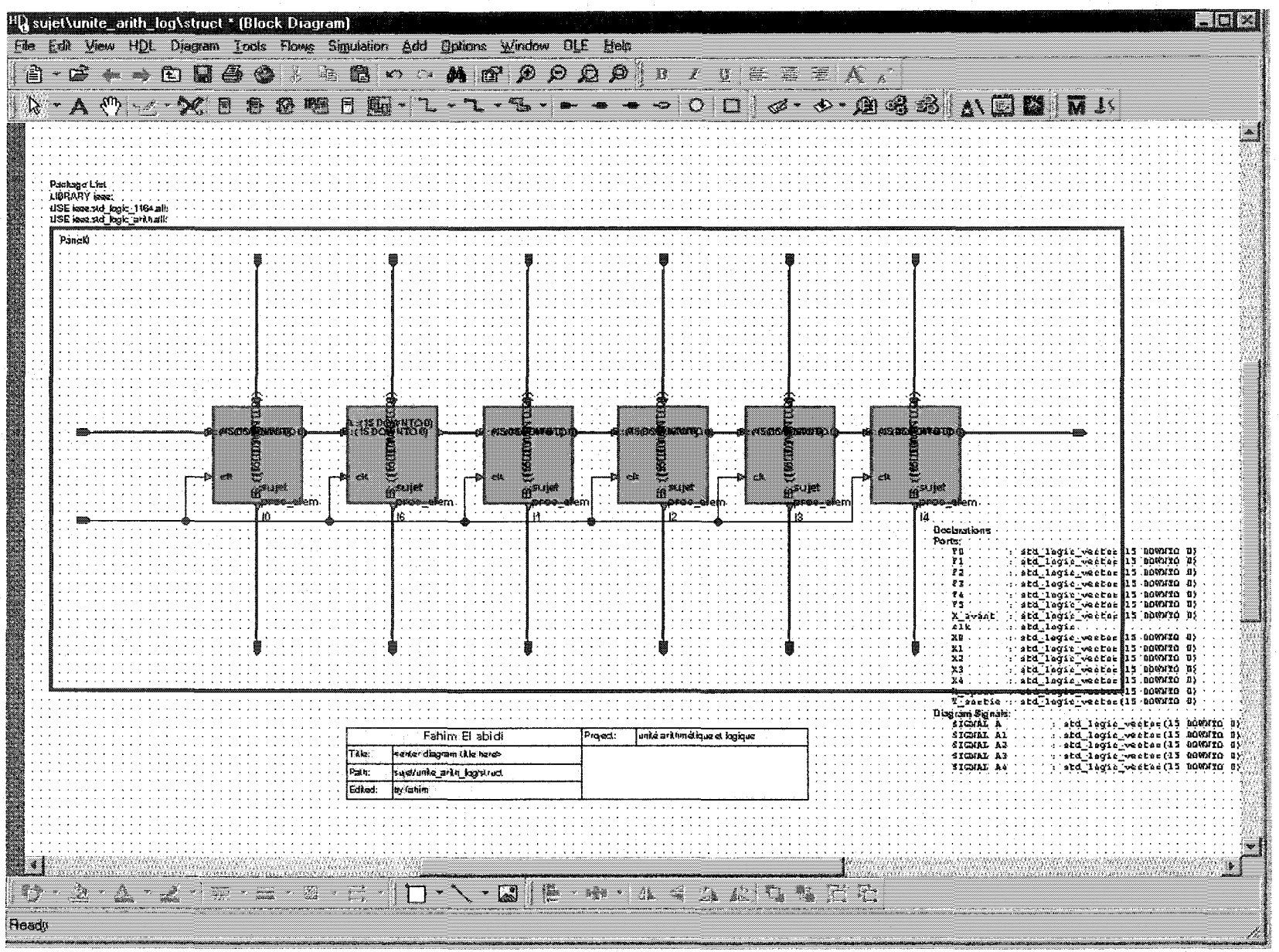
```
end component;

constant PERIOD: time := 4 us;
signal h : std_logic;
signal etat_pr : std_logic_vector(15 downto 0);
signal entree1 :  std_logic_vector(15 downto 0);
signal     entree2 :  std_logic_vector(15 downto 0);
signal     entree3 :  std_logic_vector(15 downto 0);
signal     entree4 :  std_logic_vector(15 downto 0);
signal     entree5 :  std_logic_vector(15 downto 0);
signal     entree6 :  std_logic_vector(15 downto 0);
signal     etat_sv1 : std_logic_vector(15 downto 0);
signal     etat_sv2 : std_logic_vector(15 downto 0);
signal     etat_sv3 : std_logic_vector(15 downto 0);
signal     etat_sv4 : std_logic_vector(15 downto 0);
signal     etat_sv5 : std_logic_vector(15 downto 0);
signal     etat_sv6 : std_logic_vector(15 downto 0);
signal done: boolean := false;

begin
    ABC: reseau port map ( h, etat_pr, entree1, entree2, entree3,
entree4, entree5, entree6, etat_sv1, etat_sv2, etat_sv3, etat_sv4,
etat_sv5, etat_sv6 );

    CLOCK1: process

     variable clktmp: std_ulogic := '1';
    begin
        clktmp := not clktmp;
        h <= clktmp;
           if done = true then
           wait;
        end if;
         wait for PERIOD/2;
    end process CLOCK1;

    STIMULUS1: process
    begin
           etat_pr<="0000000000000001";
           entree1<="0000000000000011"; entree2<="0000000000000000";
entree3<="0000000000000000"; entree4<="0000000000000000";
entree5<="0000000000000000"; entree6<="0000000000000000";
           wait for 2.002 us;

           etat_pr<="0000000000000010";
           entree1<="0000000000000000"; entree2<="0000000000000000";
entree3<="0000000000000000"; entree4<="0000000000000000";
entree5<="0000000000000000"; entree6<="0000000000000000";
           wait for 4 us;

           etat_pr<="0000000000000011";
```

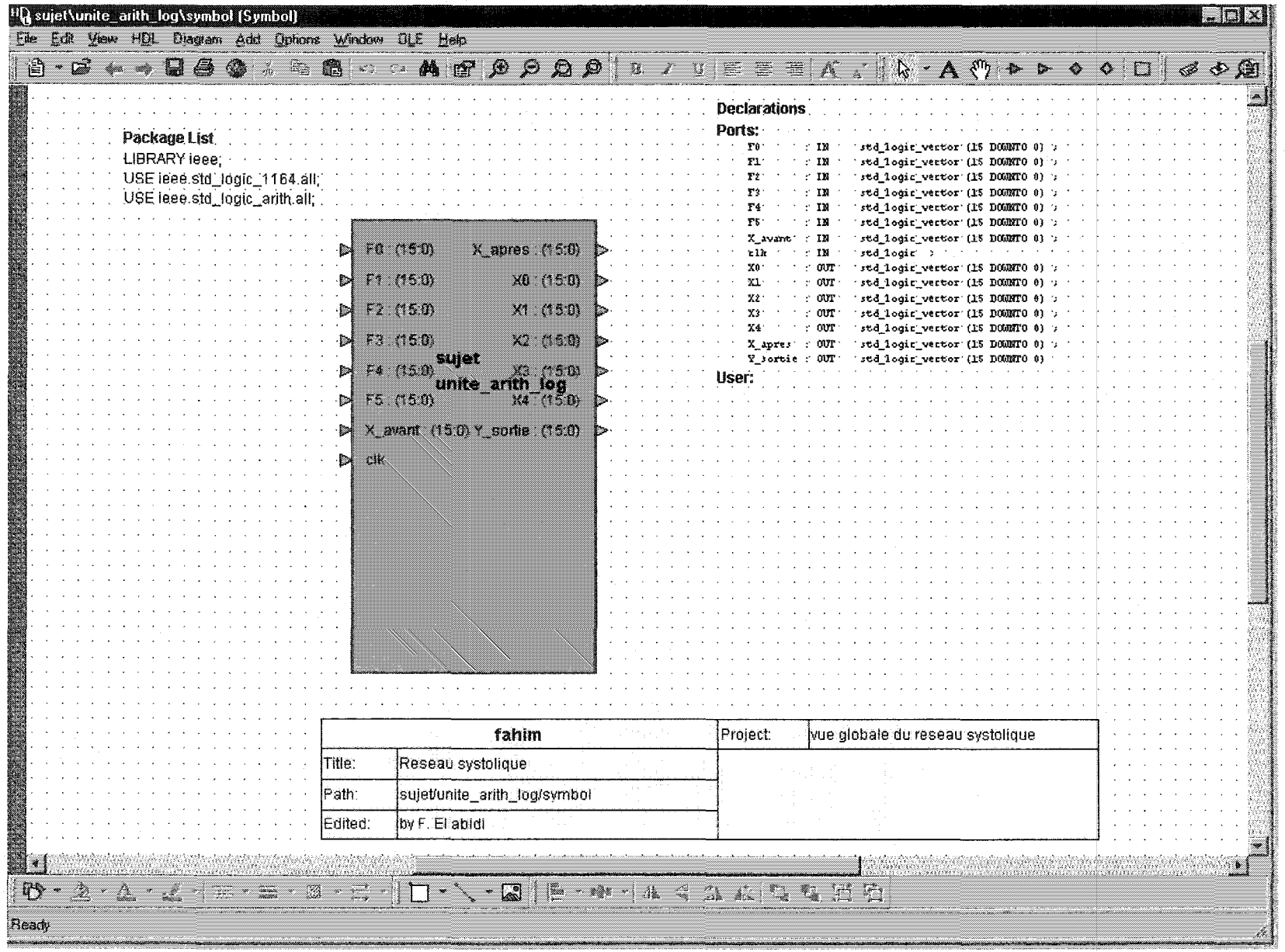
```
            entree1<="0000000000000000"; entree2<="0000000000000001";
entree3<="0000000000000000"; entree4<="0000000000000000";
entree5<="0000000000000000"; entree6<="0000000000000000";
            wait for 4 us;

            etat_pr<="0000000000000100";
            entree1<="0000000000000000"; entree2<="0000000000000000";
entree3<="0000000000000000"; entree4<="0000000000000000";
entree5<="0000000000000000"; entree6<="0000000000000000";
            wait for 4 us;

            etat_pr<="0000000000000101";
            entree1<="0000000000000000"; entree2<="0000000000000000";
entree3<="0000000000000010"; entree4<="0000000000000000";
entree5<="0000000000000000"; entree6<="0000000000000000";
            wait for 4 us;

            etat_pr<="0000000000000110";
            entree1<="0000000000000000"; entree2<="0000000000000000";
entree3<="0000000000000000"; entree4<="0000000000000000";
entree5<="0000000000000000"; entree6<="0000000000000000";
            wait for 4 us;

            etat_pr<="0000000000000000";
            entree1<="0000000000000000"; entree2<="0000000000000000";
entree3<="0000000000000000"; entree4<="0000000000000110";
entree5<="0000000000000000"; entree6<="0000000000000000";
            wait for 4 us;

            etat_pr<="0000000000000000";
            entree1<="0000000000000000"; entree2<="0000000000000000";
entree3<="0000000000000000"; entree4<="0000000000000000";
entree5<="0000000000000000"; entree6<="0000000000000000";
            wait for 4 us;

            etat_pr<="0000000000000000";
            entree1<="0000000000000000"; entree2<="0000000000000000";
entree3<="0000000000000000"; entree4<="0000000000000000";
entree5<="0000000000000100"; entree6<="0000000000000000";
            wait for 4 us;

            etat_pr<="0000000000000000";
            entree1<="0000000000000000"; entree2<="0000000000000000";
entree3<="0000000000000000"; entree4<="0000000000000000";
entree5<="0000000000000000"; entree6<="0000000000000000";
            wait for 4 us;

            etat_pr<="0000000000000000";
            entree1<="0000000000000000"; entree2<="0000000000000000";
entree3<="0000000000000000"; entree4<="0000000000000000";
entree5<="0000000000000000"; entree6<="0000000000000101";
            wait for 4 us;
```

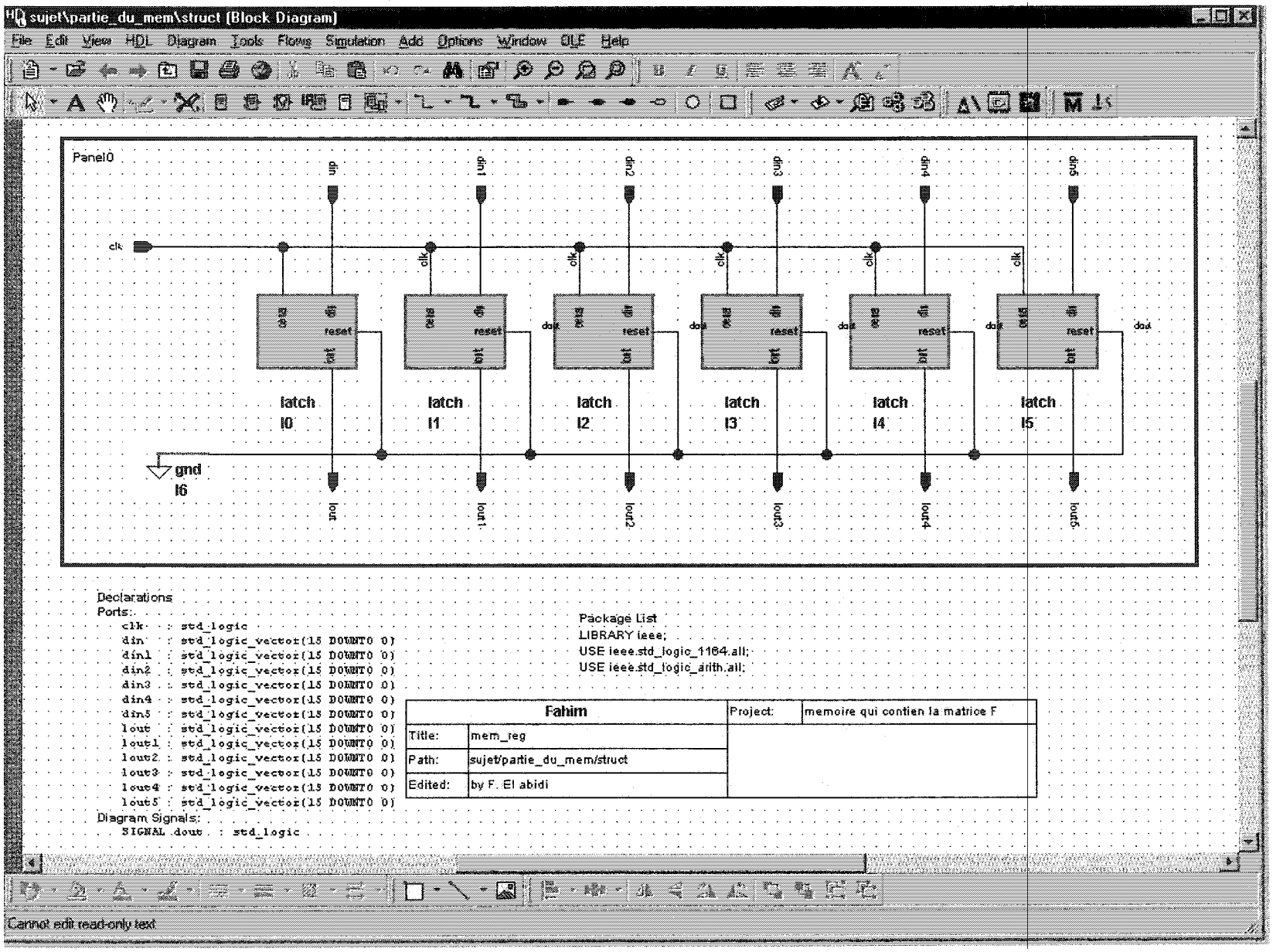
```
        done <= true;     -- Turn off the clock
        wait;             -- Suspend simulation
    end process STIMULUS1;

end stim;


library ieee;
use ieee.std_logic_1164.all;
-- use ieee.numeric_std.all;      -- Note: uncomment this if you use
                                  -- IEEE standard signed or unsigned
types.
-- use ieee.std_logic_arith.all; -- Note: uncomment/modify this if you
use
                                  -- Synopsys signed or unsigned types.
use std.textio.all;
use work.pq;

entity t_pq is
end t_pq;

architecture stimulus of t_pq is

component pq is
     port (
          e, xi : in std_logic_vector(31 downto 0);
          s : out std_logic_vector(15 downto 0);
          xo : out std_logic_vector(31 downto 0)
     );
end component;

signal e, xi : std_logic_vector(31 downto 0);
signal xo : std_logic_vector(31 downto 0);
signal s : std_logic_vector(15 downto 0);


begin
    ABC: pq port map (e, xi, s, xo );

    STIMULUS1: process
    begin
          e<="00000000000000000000000000000010";
xi<="00000000000000000000000000000001"; wait for 1.8 us;
          xi<="00000000000000000000000000000010"; wait for 4 us;
          xi<="00000000000000000000000000000011"; wait for 4 us;
          xi<="00000000000000000000000000000100"; wait for 4 us;

        wait;             -- Suspend simulation
    end process STIMULUS1;
end stimulus;
```


sujet\PE\struct * (Block Diagram)
File Edit View HDL Diagram Tools Flows Simulation Add Options Window OLE Help
Package List
LIBRARY ieee:
USE ieee.std_logic_1164.all:
USE ieee.std_logic_arith.all:
Declarations
Ports:
A : std_logic_vector(15 DOWNTO 0)
B : std_logic_vector(15 DOWNTO 0)
clk : std_logic
E : std_logic_vector(15 DOWNTO 0)
S : std_logic_vector(15 DOWNTO 0)
Diagram Signals:
SIGNAL R : std_logic_vector(31 DOWNTO 0)
SIGNAL dout : std_logic
SIGNAL dout1 : std_logic
SIGNAL lout : std_logic_vector(31 DOWNTO 0)
SIGNAL m : std_logic_vector(31 DOWNTO 0)
SIGNAL memo : std_logic_vector(31 DOWNTO 0)
Panel0
gnd
I6
latch
I1
gnd
I4
mult
I3
add
I2
latch
I0
eb1
1
eb1 1
S <= lout(31 downto 16);
latch
I5
gnd
fahim
Project:
PE
Title:
processeur elementaire
Path:
sujet/PE/struct
Edited:
by Fahim
Ready


sujet\proc_elem\struct * (Block Diagram)
File Edit View HDL Diagram Tools Flows Simulation Add Options Window OLE Help
Package List
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_arith.all;
Declarations
Ports:
A : std_logic_vector(15 DOWNTO 0)
E : std_logic_vector(15 DOWNTO 0)
clk : std_logic
B : std_logic_vector(15 DOWNTO 0)
S : std_logic_vector(15 DOWNTO 0)
Diagram Signals:
Panel0
A
E
clk
sujet
PE
i0
B (15:0)
B
S (15:0)
S
fahim
Project:
PE
Title:
processeur élémentare
Path:
sujet/proc_elem/struct
Edited:
by f. El abidi
Ready


sujet\unite_arith_log\struct * (Block Diagram)
File Edit View HDL Diagram Tools Flows Simulation Add Options Window OLE Help
Package List
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_arith.all;
Panel0
sujet
proc_elem
I0
I6
I1
I2
I3
I4
Declarations
Ports:
Diagram Signals:
Fahim El abidi
Project:
unité arithmétique et logique
Title:
<enter diagram title here>
Path:
sujet/unite_arith_log/struct
Edited:
by fahim
Ready


sujet\unite_arith_log\symbol (Symbol)
File Edit View HDL Diagram Add Options Window OLE Help
Package List
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_arith.all;
F0 (15:0)
F1 (15:0)
F2 (15:0)
F3 (15:0)
F4 (15:0)
F5 (15:0)
X_avant (15:0)
clk
X_apres (15:0)
X0 (15:0)
X1 (15:0)
X2 (15:0)
X3 (15:0)
X4 (15:0)
Y_sortie (15:0)
sujet
unite_arith_log
Declarations
Ports:
F0 : IN std_logic_vector (15 DOWNTO 0) ;
F1 : IN std_logic_vector (15 DOWNTO 0) ;
F2 : IN std_logic_vector (15 DOWNTO 0) ;
F3 : IN std_logic_vector (15 DOWNTO 0) ;
F4 : IN std_logic_vector (15 DOWNTO 0) ;
F5 : IN std_logic_vector (15 DOWNTO 0) ;
X_avant : IN std_logic_vector (15 DOWNTO 0) ;
clk : IN std_logic ;
X0 : OUT std_logic_vector (15 DOWNTO 0) ;
X1 : OUT std_logic_vector (15 DOWNTO 0) ;
X2 : OUT std_logic_vector (15 DOWNTO 0) ;
X3 : OUT std_logic_vector (15 DOWNTO 0) ;
X4 : OUT std_logic_vector (15 DOWNTO 0) ;
X_apres : OUT std_logic_vector (15 DOWNTO 0) ;
Y_sortie : OUT std_logic_vector (15 DOWNTO 0)
User:
fahim
Project: vue globale du reseau systolique
Title: Reseau systolique
Path: sujet/unite_arith_log/symbol
Edited: by F. El abidi
Ready

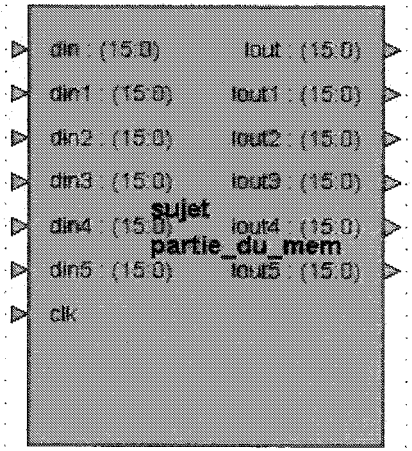

sujet\partie_du_mem\struct (Block Diagram)
File Edit View HDL Diagram Tools Flows Simulation Add Options Window OLE Help
Panel0
din
din1
din2
din3
din4
din5
clk
reset
latch
l0
l1
l2
l3
l4
l5
gnd
l6
dout
lout
lout1
lout2
lout3
lout4
lout5
Declarations
Ports:
clk : std_logic
din : std_logic_vector(15 DOWNTO 0)
din1 : std_logic_vector(15 DOWNTO 0)
din2 : std_logic_vector(15 DOWNTO 0)
din3 : std_logic_vector(15 DOWNTO 0)
din4 : std_logic_vector(15 DOWNTO 0)
din5 : std_logic_vector(15 DOWNTO 0)
lout : std_logic_vector(15 DOWNTO 0)
lout1 : std_logic_vector(15 DOWNTO 0)
lout2 : std_logic_vector(15 DOWNTO 0)
lout3 : std_logic_vector(15 DOWNTO 0)
lout4 : std_logic_vector(15 DOWNTO 0)
lout5 : std_logic_vector(15 DOWNTO 0)
Diagram Signals:
SIGNAL dout : std_logic
Package List
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_arith.all;
Fahim
Project:
memoire qui contien la matrice F
Title:
mem_reg
Path:
sujet/partie_du_mem/struct
Edited:
by F. El abidi
Cannot edit read-only text

sujet\partie_du_mem\symbol * (Symbol)

File Edit View HDL Diagram Add Options Window OLE Help

Package List

LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_arith.all;



Declarations

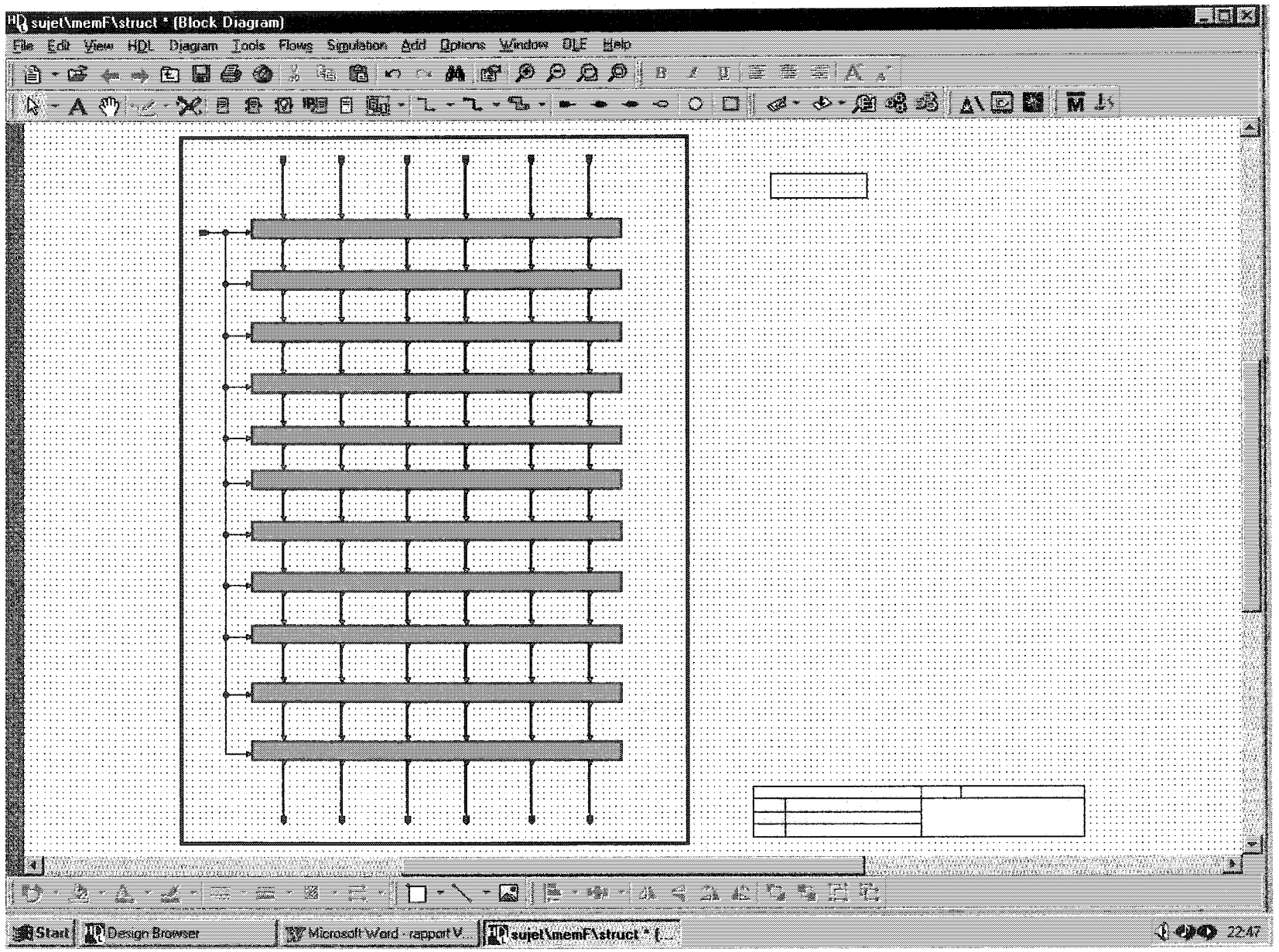
Ports:

```
clk   : IN    std_logic ;
din   : IN    std_logic_vector (15 DOWNTO 0) ;
din1  : IN    std_logic_vector (15 DOWNTO 0) ;
din2  : IN    std_logic_vector (15 DOWNTO 0) ;
din3  : IN    std_logic_vector (15 DOWNTO 0) ;
din4  : IN    std_logic_vector (15 DOWNTO 0) ;
din5  : IN    std_logic_vector (15 DOWNTO 0) ;
dout  : OUT   std_logic_vector (15 DOWNTO 0) ;
dout1 : OUT   std_logic_vector (15 DOWNTO 0) ;
dout2 : OUT   std_logic_vector (15 DOWNTO 0) ;
dout3 : OUT   std_logic_vector (15 DOWNTO 0) ;
dout4 : OUT   std_logic_vector (15 DOWNTO 0) ;
dout5 : OUT   std_logic_vector (15 DOWNTO 0)
```

User:

fahim		Project:	partie du memoire
Title:	1/11 du mem		
Path:	sujet/partie_du_mem/symbol		
Edited:	by F. El abidi		

Ready

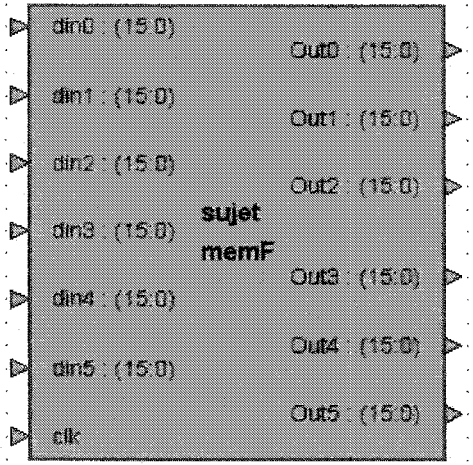

sujet\memF\struct * (Block Diagram)
File Edit View HDL Diagram Tools Flows Simulation Add Options Window OLE Help
Start
Design Browser
Microsoft Word - rapport V...
sujet\memF\struct * (...
22:47

sujet\memF\symbol (Symbol)

File Edit View HDL Diagram Add Options Window DLE Help

Package List

LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_arith.all;



Declarations

Ports:

```
clk  : IN   std_logic  ;
din0 : IN   std_logic_vector (15 DOWNTO 0) ;
din1 : IN   std_logic_vector (15 DOWNTO 0) ;
din2 : IN   std_logic_vector (15 DOWNTO 0) ;
din3 : IN   std_logic_vector (15 DOWNTO 0) ;
din4 : IN   std_logic_vector (15 DOWNTO 0) ;
din5 : IN   std_logic_vector (15 DOWNTO 0) ;
Out0 : OUT  std_logic_vector (15 DOWNTO 0) ;
Out1 : OUT  std_logic_vector (15 DOWNTO 0) ;
Out2 : OUT  std_logic_vector (15 DOWNTO 0) ;
Out3 : OUT  std_logic_vector (15 DOWNTO 0) ;
Out4 : OUT  std_logic_vector (15 DOWNTO 0) ;
Out5 : OUT  std_logic_vector (15 DOWNTO 0)
```

User:

Fahim		Project:	registre contenant F
Title:	reg de matrice F	<enter comments here>	
Path:	sujet/memF/symbol		
Edited:	by Fahim		

Ready