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

Ecole Nationale Polytechnique

Département électronique

Mémoire de Projet de Fin d'Etude

Pour l'Obtention du Diplôme
d'Ingénieur d'état en électronique

SUJET

Techniques d'identification aveugle multicapteurs : Etude algorithmique, réalisation d'une plate forme logicielle et implémentation sur circuit FPGA

Présenté par :

- Aïssa El Bey Abdeldjallil
- Grebici Mimi

Encadré par :

- Dr. A. Belouchrani, ENP
- Dr. K. Abed-Meraim, Telecom Paris
- M. A. Oudjida, CDTA

Jury :

- Dr. L. Hamami, ENP
- M. H. Bousbia-Salah, ENP
- Dr. A. Belouchrani, ENP
- M. A. Oudjida, CDTA

PROMOTION : Juin 2003

En préambule, nous ne pouvons nous empêcher de penser à ce funeste jour du 21 mai 2003 où un séisme de forte magnitude est venu endeuiller l'Algérie toute entière et ému la communauté internationale.

Nos sentiments les plus attristés accompagnent la mémoire de toutes les victimes que ce tremblement de terre a provoqué.

Remerciements

Ce rapport est pour nous l'occasion d'exprimer des remerciements à tous ceux qui nous ont conseillé et aidé à réaliser ce travail dans de bonnes conditions.

On remercie tout d'abord nos encadreurs :

- *Dr. Adel Belouchrani*
- *Dr. Karim Abed-Meraim*
- *Dr. Abdelkrim Oudjida,*

qui ont fait que notre travail soit particulièrement enrichissant. On leur est reconnaissant pour les efforts et le temps qu'ils nous ont consacré.

On remercie aussi les membres du jury pour avoir accepté de lire ce document et d'évaluer notre travail.

On tient finalement à remercier vivement nos parents et tous les membres de nos familles, respectives, pour les encouragements et le soutien moral qu'ils nous ont prodigués, sans lesquels on n'aurait pu terminer ce travail.

Aïssa El Bey Abdeldjallil et Mimi Grebici

ملخص:

في إطار الإتصالات الرقمية ذات الرواج العالي, فإن الإنتشار يؤدي إلى تشتت الدفعات في الزمن خاصة بالنسبة للدفعات التي تسلك طريق أطول في الزمن. منه يجب إدخال علاج ألا وهو التسوية و هدفه هو خفض التداخل بين الدفعات. يتم تعديل جهاز الاستقبال بإستعمال موجة معروفة تسمى بموجة التدريب, في حالة عدم توفر هذه الموجة يسمى العلاج بالتسوية العمياء . يوجد عدة خوارزميات للتسوية العمياء التي تستثمر التنوع الزماني المكاني الذي يسمح بإستعمال الإحصاءات ذات الدرجة الثانية. أحد هذه الخوارزميات هي خوارزمية العلاقات المتقاطعة, التي كانت محل بحث معمق لتحسين مردوديتها. كما كانت محل غرس على جهاز Xilinx VirtexII FPGA باستعمال لغة البر مجة VHDL.

Résume:

Dans le cadre de la transmission numérique à haut débit, la propagation induit une dispersion des impulsions dans le temps en particulier par des trajets de grande durée par rapport au trajet le plus court. Il faut alors introduire un traitement, l'égalisation, pour réduire l'effet des interférences entre symboles. Quand cela est possible le réglage de l'égaliseur est effectué à l'aide d'une séquence de symboles connus d'avance du récepteur dite séquence d'apprentissage. On peut alors vouloir supprimer cette séquence et traiter le problème en aveugle. Il existe plusieurs algorithmes d'égalisation aveugle qui exploitent la diversité spatio-temporelle permettant ainsi l'utilisation des statistiques d'ordre deux. L'un de ces algorithmes est l'algorithme des relations-croisées (CR). Cet algorithme a fait l'objet d'une étude approfondie et d'une amélioration de son coût de calcul, ainsi que d'une implémentation d'une architecture adaptée de cet algorithme sur circuit FPGA Virtex II de Xilinx en utilisant le langage VHDL.

Abstract :

In the context of high-rate communication systems, the multi-path propagation channel results into a time dispersion of the transmitted signal which leads to inter-symbol interferences (ISI). Therefore, an equalization process is required at the receiver side to combat the ISI phenomenon. Standard equalization techniques use a known symbol sequence referred to as training sequence to achieve the channel (or the equalizer) identification. Blind equalization techniques are alternative methods which do not resort to such known sequences and thus achieve the channel equalization using only the observation signal. Many blind equalization techniques exploit the spatio-temporal diversity and consequently can achieve the blind equalization using only the second order statistics of observed signals. Our focus has been on one of these methods known as the cross-relation (CR) method. We have proposed new CR versions of improved computational cost and developed an adapted architecture for its implementation on FPGA Virtex II of Xilinx using VHDL language.

Mots clés : Egalisation aveugle, communications numériques, FPGA, VHDL, Virtex II

Résumé :

Introduction :

Dans le cadre des transmissions numériques à hauts débits, la propagation induit une dispersion des impulsions dans le temps, en particulier par des trajets de grande durée par rapport au trajet le plus court. Dans la plupart des conditions de propagation, les dispersions que subit le signal émis se traduisent par des interférences entre symboles successifs que l'on modélise par un filtre linéaire invariant dans le temps. Lorsque cette condition n'est pas vérifiée, c'est-à-dire lorsque les conditions de propagation varient rapidement pendant la durée symbole, les détecteurs de symboles à seuils classiques deviennent inutilisables. Il faut alors introduire un traitement, l'égalisation, pour réduire l'effet des interférences entre symboles ou pour ajuster le détecteur au canal.

Quand cela est possible, le réglage de l'égaliseur est effectué à l'aide d'une séquence de symboles connus d'avance du récepteur dite séquence d'apprentissage. Mais cette séquence n'est pas toujours disponible et sa présence diminue le débit utile de transmission. On peut alors vouloir supprimer cette séquence et traiter le problème en aveugle.

L'égalisation aveugle nécessite l'utilisation d'informations autres que les statistiques du second ordre du signal reçu. Car celles-ci ne permettent l'égalisation que de canaux à phase minimale, ce qui n'est pas toujours évident à avoir.

Afin d'introduire une information supplémentaire, il existe des algorithmes d'égalisation aveugle qui se basent sur des statistiques d'ordres supérieurs. En exploitant des moments d'ordres supérieurs, ces algorithmes manipulent des fonctions complexes et convergent mal vers la solution désirée. Pour éviter l'utilisation de ces fonctions complexes, une approche parue depuis les travaux de Gardner et Tong 1991, permet aux algorithmes d'égalisation aveugle d'utiliser les statistiques du second ordre du signal reçu en le suréchantillonnant en lui donnant ainsi une nature cyclique ou en utilisant des canaux multiples (i.e récepteur avec plusieurs capteurs). Depuis, de nombreuses méthodes ont été proposées qui reposent sur la diversité temporelle ainsi créée ou bien sur la diversité spatial (utilisation de plusieurs capteurs en réception). Nous évoquons plusieurs de ces méthodes dans le chapitre 2 de la première partie du rapport.

La description d'un algorithme, après sa formulation mathématique, est faite souvent selon une architecture inadaptée à sa mise en œuvre. Des études sur ce thème permettent de passer d'une description abstraite d'un algorithme à sa réalisation pratique sur des systèmes de traitement. C'est pourquoi une étude complète des algorithmes et leur reformulation, s'avère nécessaire pour une implémentation efficace sur des composants de traitement.

Pour cet effet, une deuxième étude a été lancée durant notre travail. Celle-ci, quoi qu'elle reste incomplète, comprend une introduction à la mise en œuvre d'une méthode de conception, ainsi qu'une présentation d'une architecture adaptée pour circuits FPGAs de l'implémentation de la méthode CR (méthode des relations croisées) pour l'identification aveugle multicapteurs.

Au cours des quinze dernières années, les méthodes de conception des fonctions numériques ont subi une évolution importante. D'abord, sont apparus les applications de la logique câblée construites autour de circuits intégrés standard, souvent pris dans la famille

TTL, parallèlement, il y a eu les premiers circuits programmables par l'utilisateur du côté des circuits simples et les circuits intégrés spécifiques (ASICs) pour les fonctions complexes fabriquées en grande série. La complexité de ces derniers a nécessité la création d'outils logiciels de haut niveau qui sont à la description structurelle (schémas au niveau des portes élémentaires) ce que les langages évolués sont au langage machine dans le domaine de la programmation. A l'heure actuelle, l'écart de complexité entre circuits programmables et ASICs s'est restreint : on trouve une gamme continue de circuits qui vont des héritiers des premiers PALs (programmable array logic), équivalents de quelques centaines de portes, à des FPGAs (Field programmable gate array) ou des LCAs (Logic cell array) de quelques dizaines de milliers de portes équivalentes. Les outils d'aide à la conception se sont unifiés ; un même langage, VHDL par exemple, peut être employé quels que soient les circuits utilisés, des PALs aux ASICs.

But du travail :

Le but de ce travail est d'étudier le domaine de l'égalisation aveugle en se focalisant en particulier sur les principes de base et quelques méthodes d'égalisation aveugle pour des systèmes RIF (réponse impultionnelle finie) à entrée unique et multiple sorties (SIMO)

Un but important de notre travail est celui de concevoir une plate forme logicielle qui intégrera certaines méthodes SIMO d'égalisation aveugle et permettra ainsi leur évaluation et leur études comparatives dans un environnement ergonomique agréable. La plate forme devra être conçue de façon à être facilement extensible afin de pouvoir l'enrichir et y intégrer d'autres fonctions et méthodes dans le futur.

Un autre objectif est celui d'étudier plus en détailles l'une des méthodes SIMO d'identification aveugle: méthode des relation croisées. Il s'agit ici d'étudier sa mise en œuvre pratique et éventuellement apporter des modifications à cette méthode afin de réduire son coût de calcul.

Une nouvelle partie a été ajoutée à cette étude permettant de juger la capacité de l'algorithme CR à s'adapter au domaine de la microélectronique, entre autres trouver les modifications qu'il faudrait lui apporter, si jamais cela était nécessaire, afin d'adapter son architecture, et voir si les limites physiques (fréquence, tailles des données, flot des données,...) du circuit programmable utilisé pourrait convenir aux applications de cet algorithme. Il est à noter qu'un tel algorithme est destiné à trouver son application au cœur de systèmes de communications numériques par exemple (appareils de communication mobile).

Travail effectué :

Au cours de notre travail, nous avons étudié quelques méthodes SIMO d'égalisation aveugle du second ordre, avec une attention plus particulière sur la méthode des relations croisées, que nous avons modifié afin de minimiser le coût de calcul de son algorithme. Ce qui a donné naissance à deux nouvelles versions de cette méthode que nous présentons au chapitre 4 de la première partie du rapport. Ce travail a fait l'objet d'une publication d'un article dans une conférence internationale IEEE (*7th International Symposium on Signal Processing and its Applications, Paris July 2003*).

Nous avons aussi réalisé une plate forme logicielle permettant l'évaluation des performances des algorithmes étudiés, ainsi que l'études comparative de ces dernières.

Dans la deuxième partie, nous avons consacré les deux premiers chapitres à la description des étapes qu'il faut considérer afin de réussir un projet de conception. Ces étapes se résument comme suit:

- choisir un style de conception selon les besoins du projet;
- connaître les caractéristiques du circuit utilisé;
- respecter une méthodologie de conception.

Le troisième chapitre décrit le travail effectué dont le but était l'implémentation de l'algorithme des relations croisées sur circuit FPGA. Ce travail n'a été fait que partiellement, nous avons réussi à réaliser la partie spécification qui consiste en une validation par simulations fonctionnelles, qui est une première description permettant de corriger ou de compléter le cahier des charges afin d'aboutir au système désiré. La partie synthèse, qui est une étape effectué automatiquement à partir du code VHDL, permet de générer un schéma de câblage. Nous nous sommes arrêté à l'étape placement et routage mais nous n'avons pas atteint l'étape de l'analyse temporelle.

L'étude de cette partie permet d'avoir une idée sur les points auxquels il faut donner de l'importance pour développer une étude d'implémentation sur circuit intégré.

Plan du rapport :

Ce rapport est constitué de deux parties :

La première partie est consacrée à la plus grande part du travail qui est l'étude algorithmique de méthodes d'égalisation aveugle du second ordre, et la réalisation d'un outil logiciel pour l'évaluation des algorithmes présentés. La présentation dans cette partie est faite en quatre chapitres :

Chapitre 1 : Généralités sur l'égalisation.
Chapitre 2 : Présentation de quelques méthodes SIMO d'égalisation aveugle.
Chapitre 3 : Présentation de la plate forme logicielle.
Chapitre 4 : Etude de la méthode des relations croisées et de ses nouvelles versions.

La deuxième partie présente une introduction à la conception de circuits intégrés spécialisés, et une introduction à un outil de description hardware, le VHDL, pour le développement d'une telle conception, ce qui permet d'avoir une vision globale sur ses étapes. Elle inclue aussi une présentation détaillée de l'architecture réalisée pour l'implémentation de la méthode CR sur FPGA. La présentation est faite en trois chapitres:

Chapitre 1 : Description des styles de conception de circuits intégrés et de certaines caractéristiques de circuit utilisé.
Chapitre 2 : Description du cycle de conception d'un circuit logique programmable.
Chapitre 3 : Description du travail effectué et de l'architecture implémentée.

Sommaire :

Résumé 1

Introduction 1
But du travail 2
Travail effectué 2
Plan du rapport 3

Partie 1 : Identification/Egalisation Aveugle Multicapteur : Plate forme Logicielle.

Chapitre I : Introduction 8

I-1 Généralités 8

I-2 Egalisation : Objectif et définitions 9

Chapitre II : Egalisation aveugle multicapteurs 14

II-1 Formulation du problème 14
II-2 Algorithmes d'identification du canal 15

II-2-1 Algorithme de la méthode du maximum de vraisemblance 15
II-2-2 Algorithme de la méthode des relations croisées 18
II-2-3 Algorithme de la méthode du sous-espace 18

II-3 Algorithmes d'égalisation directe 20

II-3-1 Algorithme de la méthode du sous-espace signal 20
II-3-2 Algorithme de la méthode égaliseurs mutuellement référencés 22
II-3-3 Algorithme de la méthode de prédiction linéaire 23

II-4 Conclusion 25

Chapitre III : Interface d'évaluation d'algorithmes d'égalisation aveugle 27

Introduction 27

III-1 Description de la plate forme 27

III-2 Simulations 30

III-3 Conclusion 34

Chapitre IV : Algorithme CR 36

IV-1 Algorithme de la méthode CR (Cross Relation) 36
IV-2 Algorithme de la méthode MCR (Minimum Cross Relation) 38
IV-3 Algorithme de la méthode UMCR (Unbiased MCR).....39
IV-4 Cas entrées multiples 40
IV-5 Simulations 41
IV-6 Conclusions 43

Partie 2 : Implémentation sur circuit FPGA de Xilinx famille Virtex II

Introduction 44

Chapitre I : Styles de conception de circuits intégrés.....46

I-1 Circuits ASICs 46
I-2 Circuits PLDs 47
- PALs 47
- CPLDs.....47
- FPGAs.....47
  - Circuits FPGA Virtex-II.....48
    - Description générale de l'architecture d'un circuit FPGA Virtex-II...48
    - CLBs / Slices 49
    - IOBs 50
    - DCM 51
    - Blocs Multiplieurs 51
    - Blocs Mémoire.....52

I-3 Conclusion 53

Chapitre II : Cycle de conception des circuits FPGAs :.....55

II-1 Langage de description VHDL 55
II-2 Approche de conception 56
II-3 Outil de conception ISE Foundation 57
II-4 Conclusion.....59

Chapitre III : Implémentation de l'algorithme CR sur circuits FPGA de Xilinx famille Virtex II 61

III-1 : Présentation générale de l'algorithme implémenté 61

III-2 Architecture proposée.....62

III-2-1 Dimensionnement des données 62
III-2-2 Choix des fonctions 64
III-2-3 Simulations 74
III-2-4 Synthèse 80
III-2-5 Implémentation 80

III-3 Conclusions 81

Conclusion Générale.....82

Annexes

Annexe 1 : Etat de l'art des FPGAs de Xilinx 83

1 - Introduction (propriétés) 83
2 - La famille Virtex-II 83
2-1 Généralités sur les produits Virtex-II 83
2-2 Les circuits Virtex-II 85
- Résumé des principales caractéristiques des FPGAs Virtex-II 85
3 - Derniers produits Xilinx 87

Annexe 2 Définitions 89

Annexe 3 Programmes VHDL 91

Annexe 4 Article 123

Annexe 5 Rapport de l'analyse temporelle 127

Bibliographie.....131

Partie 1

Chapitre I

Introduction

I-1 Généralités 8

I-2 Egalisation : Objectif et définition 9

Chapitre I :

Introduction

I-1 Généralités : Description d'une chaîne de communication

Le but d'un système de communication numérique est de transmettre une suite de symboles porteurs d'information, d'un point à un autre (émetteur/récepteur).
Les retards de propagation dûs aux trajets multiples que peut rencontrer le signal transmis, provoquent la superposition de plusieurs versions décalées d'une même séquence de symboles. Cette superposition est appelée interférence entre symboles. En d'autres termes, la multitude de trajets -autres que le trajet direct- que trouve l'acheminement de cette suite de symboles, provoque chez le récepteur des interférences entre symboles successifs.
Cette multitude de trajets est due, dans le cas des communications mobiles, aux réflexions que subit le signal émis sur les objets qui constituent l'environnement de sa transmission. C'est le cas, par exemple, lors de la transmission dans un milieu urbain d'un message d'un émetteur mobile à une station de base. L'onde transmise est réfléchie sur des immeubles, des voitures et autres, et la station recevra donc le signal et plusieurs de ses versions retardées et atténuées.

Il existe un traitement appliqué au signal reçu dont le but est de permettre la restitution du message émis en éliminant les interférences entre symboles; ce traitement est appelé **égalisation**.

Cet égaliseur fait partie d'une chaîne dite de communication, dont le rôle de chacun des éléments qui la constituent est bien défini; préparer le signal à transmettre, adapter le signal au moyen de transmission, véhiculer le signal, recevoir le signal et le remettre à sa bande de base, "traiter le signal" et enfin restituer l'information du signal. Ces différentes fonctions sont attribuées respectivement aux dispositifs suivants: Un dispositif de numérisation de compression (codage source) et de codage canal; Un dispositif de modulation ; Un support de transmission; Un dispositif de démodulation ; Un dispositif d'égalisation ; Un dispositif de décodage et de décompression.

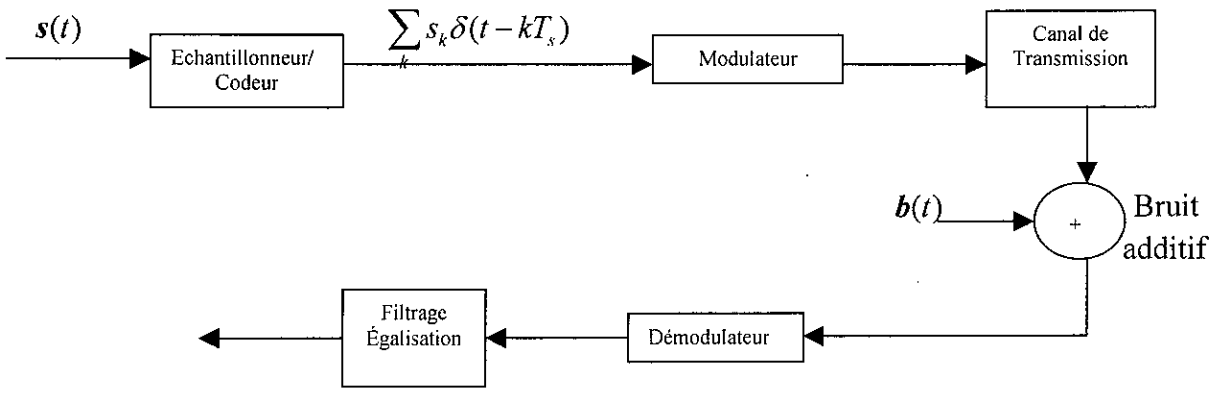


Fig.1.1 : Modèle de chaîne de transmission.

Après avoir situé l'égalisation dans un système de communication, le paragraphe suivant présente un descriptif de ce traitement.

I-2 Egalisation : Objectif et définition

En communication numérique, l'égalisation désigne un traitement appliqué à un signal afin de réduire au mieux l'effet du bruit et les distorsions que ce signal aurait subit en traversant le canal de propagation (support de transmission, dispositifs électroniques d'émission et de réception).

De même, un égaliseur peut être considéré aussi comme un filtre linéaire (la plupart du temps considéré comme tel) qui permet d'adapter le récepteur au canal de transmission.

Dans le domaine fréquentiel, et en l'absence de bruit, un égaliseur serait l'inverse de la réponse fréquentielle de l'ensemble des dispositifs de communication (méthode forçage à zéro). Cela rendrait la réponse fréquentielle globale plate et permettrait de retrouver le message envoyé directement.

Dans la plupart des cas, l'algorithme d'égalisation calcule les coefficients du filtre égaliseur à l'aide d'une séquence dite d'apprentissage au cours de laquelle l'émetteur envoie une séquence connue à l'avance par le récepteur. Ces coefficients restent constants jusqu'à la nouvelle séquence d'apprentissage envoyée par l'émetteur.

Il n'est pas toujours possible d'avoir une telle séquence, surtout dans le cas des communications de type point à multipoint où la mise en place d'un nouveau poste nécessiterait l'envoi d'une nouvelle séquence d'apprentissage, ou dans un cadre non coopératif. De plus, par son existence, cette séquence réduit le débit utile de transmission car elle occupe une part de la trame de données transmise (exp. pour le standard GSM en communications mobiles, cette séquence est de 26 bits sur 148 bits par trame ce qui correspond à une perte en débit utile de plus de 17% de perte).

Une solution alternative serait de traiter le problème en aveugle sans utiliser la séquence d'apprentissage.

Pour faciliter le traitement du signal reçu, la structure d'un égaliseur commence toujours par un échantillonnage afin de rendre le signal -traité par l'égaliseur- stationnaire. Cette stationnarité exige l'utilisation des statistiques d'ordre supérieur (SOS), car les statistiques du second ordre ne donnent aucune information sur la phase du signal et ne permettent l'égalisation que pour les cas où l'information sur la phase est connue ou peut être déduite (canal à phase minimale par exemple).

Beaucoup de travaux sont apparus utilisant de manière explicite ou implicite les statistiques d'ordres supérieurs (SOS) du signal reçu.

Technique SOS implicite : L'adaptation se l'égaliseur de fait en minimisant une fonction de contraste quelconque, autre que l'erreur quadratique moyenne, qui exploite implicitement les statistiques d'ordres supérieurs des signaux observés.

Pour ce type de techniques, l'égaliseur est défini par la donnée de la fonction de contraste, et ainsi tout le problème réside dans le bon choix de cette fonction.

Plusieurs méthodes utilisant ces techniques sont présentées dans [23,24 ,25 , 26, 27]

Techniques SOS explicites : L'estimation de fait d'abord sur l'amplitude de la fonction de transfert du canal en utilisant de façon explicite les statistiques du second ordre, ensuite sur la phase en utilisant les statistiques d'ordres supérieurs. Les SOS utilisées en général sont ceux du troisième et quatrième ordre. Mais le plus souvent celles du troisième ordre ne sont pas utilisées à cause de la symétrie que présente la distribution du signal reçu.

Quelques méthodes développés pour ces techniques sont présentées dans **[28, 29, 30, 31].**

Mais le fait que les techniques qui se basent sur les SOS demandent un grand nombre d'échantillons pour donner de bons résultats, rend ces techniques incompatibles avec les exigences de traitement en temps réel des systèmes de communication, car il y aurait une perte considérable d'informations utiles. Il y a encore d'autres inconvénients à leur utilisation : ces critères n'étant pas quadratiques, des minima indésirables peuvent être détectés et induire en erreur; dans certains cas la valeur de ces statistiques est proche du zéro ce qui gâcherait leur intérêt; le signal à l'entrée de l'égaliseur ne doit pas être gaussien auquel cas ses SOS seraient nuls.

Depuis les travaux de Tong **[1]** et de Gardner **[2]**, l'exploitation de la diversité spatio-temporelle a permis l'apparition de nouvelles méthodes utilisant des statistiques d'ordre 2 **[7].**

Le principe de la diversité spatio-temporelle est illustré dans le paragraphe suivant, et quelques uns des algorithmes qui se basent sur ce principe seront présentés au chapitre II et chapitre IV de cette partie.

La diversité spatiale et la diversité temporelle (ou même la combinaison des deux), représentent le modèle multi-canaux.

La diversité spatiale est obtenue en disposant plusieurs capteurs à des endroits différents (Fig.1.2). Ces endroits doivent être assez différents pour que les signaux qu'ils récoltent donnent le plus d'informations possible.

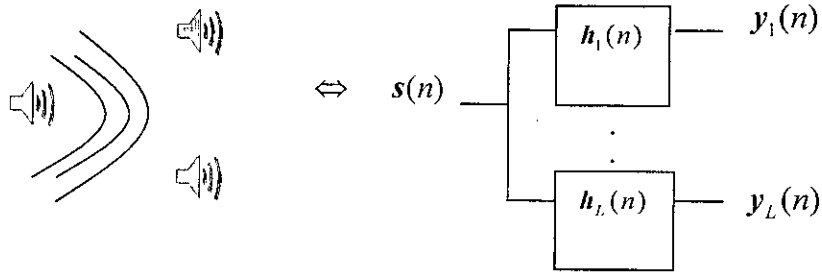


Fig.1.2 : Diversité spatiale.

$s(n)$: Signal source.

$h_l(n)$: Fonction de transfert du filtre séparant le $l^{éme}$ capteur de la source.

$y_l(n)$: Signal de sortie du $l^{éme}$ capteur.

La diversité temporelle est obtenue en échantillonnant le signal reçu à une fréquence $1/T'$, p fois plus grande que celle correspondant à la durée symbole (Fig.1.3). i.e : $T' = T/p$

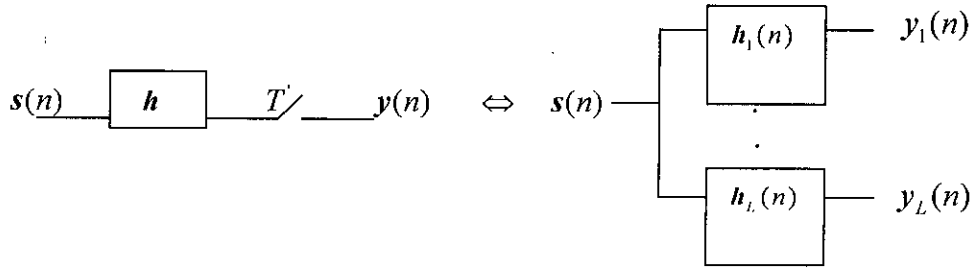


Fig.1.3 : Diversité temporelle.

La séquence $y(n)$ obtenue est cyclostationnaire de période p . Elle peut être décrite en notations vectorielles par l'expression suivante:

$$y(n) = \begin{pmatrix} y_1(n) \\ y_2(n) \\ \text{M} \\ y_L(n) \end{pmatrix} \qquad (1.1)$$

tel que: $y_l(n) = y(nT - (l-1)T/p)$
$= \boldsymbol{s}(n) * \boldsymbol{h}_l(n) + \boldsymbol{b}_l(n)$ (1.2)

où * représente l'opérateur convolution.

$\boldsymbol{h}_l(n)$: canal virtuel défini par $\boldsymbol{h}_l(n) = \boldsymbol{h}(nT - (l-1)T/p)$.
$\boldsymbol{b}_l(n)$: bruit additif correspondant au canal virtuel $\boldsymbol{h}_l(n)$.

Le système s'écrit alors comme suit:

$$y(n) = \sum_{k=0}^{M} \boldsymbol{h}(k)\boldsymbol{s}(n-k) + \boldsymbol{b}(n) \qquad (1.3)$$

Cette cyclostationarité du signal obtenu en exploitant la diversité spatio-temporelle, permet de retrouver l'information sur la phase dans l'expression des statistiques d'ordre 2. Ce résultat est démontré dans **[3]**.

Par la diversité spatiale -comme la diversité temporelle- on récolte plus d'informations que dans le cas monocapteur, et ceci permet, comme on va le voir, de déterminer la phase en utilisant seulement les statistiques du second ordre.

Toute fois, l'égalisation aveugle requière certaines conditions pour que le système considéré soit identifié de façon unique (à un scalaire près). L'identifiabilité est étudiée en détails dans **[4]**. Ces conditions traduisent d'une manière rigoureuse les exigences intuitives suivantes:

Les canaux doivent être suffisamment distincts les uns des autres (ne partagent pas de zéros communs).

La séquence d'entrée doit être suffisamment complexe pour donner le plus d'informations possible. Par exemple, elle ne peut pas être ni nulle, ni constante, ni une sinusoïde.

Il doit y avoir un nombre suffisant d'observations pour identifier le canal. Par exemple le nombre d'échantillons observé ne peut pas être inférieur au nombre de paramètres à estimer.

Pour un système identifiable, il existe dans la littérature plusieurs méthodes pour effectuer l'identification ou bien l'égalisation, uniquement à partir de l'observation recueillie, avec la diversité spatio-temporelle. Dans le prochain chapitre, nous allons présenter des méthodes qui se basent sur le principe de la diversité spatio-temporelle et sur différentes techniques d'estimation pour l'identification du canal.

Chapitre II

Egalisation aveugle multicapteurs

II-1 Formulation du problème 14

II-2 Algorithmes d'identification directe du canal 15

II-2-1 Algorithme de la méthode du maximum de vraisemblance 15

II-2-2 Algorithme de la méthode des relations croisées 18

II-2-3 Algorithme de la méthode du sous-espace canal 18

II-3 Algorithmes d'égalisation directe 20

II-3-1 Algorithme de la méthode du sous-espace signal.....20

II-3-2 Algorithme de la méthode des égaliseurs mutuellement référencés ...22

II-3-3 Algorithme de la méthode de prédiction linéaire 23

II-4 Conclusion 25

Chapitre II :

Egalisation aveugle multicapteurs

Afin d'introduire l'égalisation aveugle basée sur des statistiques du second ordre pour des systèmes type SIMO (Single Input Multiple Output), nous allons dans le présent chapitre aborder l'identification du canal et l'égalisation directe en étudiant plusieurs algorithmes exprimant chacun une technique différente :

la technique du maximum de vraisemblance;
la technique des relations croisées ;
la technique du sous-espace canal ;
la technique du sous-espace signal ;
la technique de prédiction linéaire ;
la technique des égaliseurs mutuellement référencés.

II-1 Formulation du problème :

On considère un système SIMO à L sorties ayant comme expression :

$$\boldsymbol{y}(n)=\sum_{k=0}^{M}\boldsymbol{h}(k)\boldsymbol{s}(n-k)+\boldsymbol{b}(n) \qquad n=1,\ldots,N \qquad (2.1)$$

avec :

N : taille de la trame.

M : mémoire du canal

$\boldsymbol{h}(z)=\sum_{k=0}^{M}\boldsymbol{h}(k)z^{-1}$ la fonction de transfert causale à réponse impulsionnelle finie (FIR) du canal inconnu de taille $L\text{x}1$, et avec la condition : $\boldsymbol{h}(z)\neq 0 \quad ,\forall z$

$\boldsymbol{s}(n)$: signal transmis.

$\boldsymbol{b}(n)$: Bruit additif blanc gaussien de taille $L\text{x}1$, i.e : $E\,[\boldsymbol{b}(n)\boldsymbol{b}^{\mathrm{H}}(n)] = \sigma^2 \boldsymbol{I}_L$.

L'objectif est de retrouver les coefficients du vecteur canal de transmission $\boldsymbol{h}=\left[\boldsymbol{h}(0)^T, \cdots, \boldsymbol{h}(M)^T\right]^T$ à un scalaire près en utilisant différents algorithmes d'égalisation aveugle basés sur la diversité spatio-temporelle et utilisant des statistiques du second ordre.

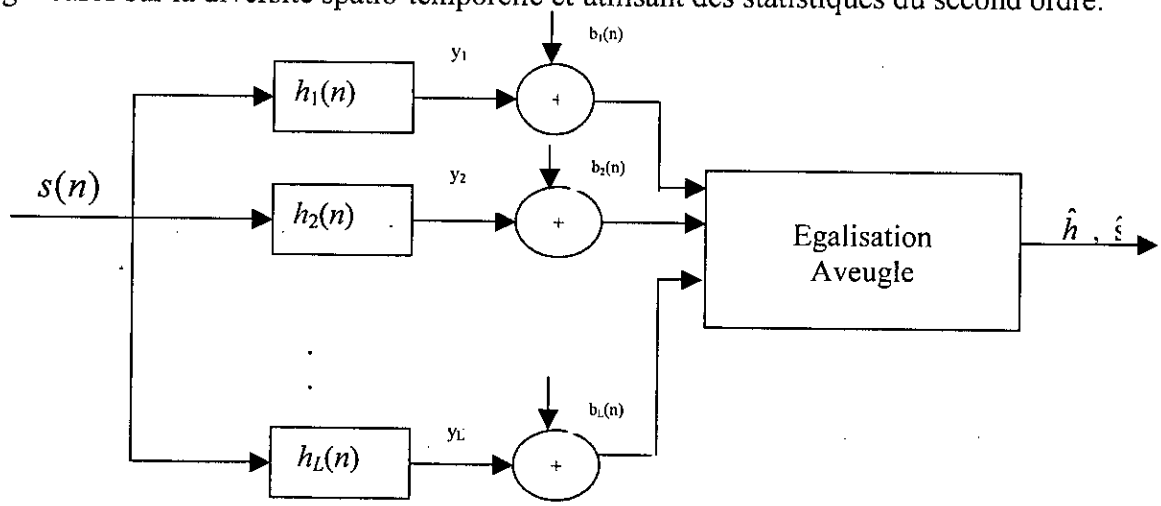


Fig.2.1: Egalisation aveugle multicapteurs.

II-2 Algorithmes d'identification directe du canal :

II-2-1 Algorithme de la méthode du maximum de vraisemblance (MV): [5]

La méthode MV s'applique à tout problème d'estimation paramétrique où la densité de probabilité des données est connue.

En supposant que le bruit à la sortie du système (1.3) est un bruit gaussien circulaire temporellement et spatialement blanc, on peut alors écrire le vecteur d'observation comme suit:

$$\boldsymbol{y} = \boldsymbol{T}_N(h)\boldsymbol{s} + \boldsymbol{b} \qquad (2.2)$$

où :

$$\boldsymbol{y} = [y^T(0)\ ,\ \cdots\ ,\ y^T(N\text{-}1)\ ]^T \qquad (2.3)$$

$$\boldsymbol{b} = [b^T(0)\ ,\ \cdots\ ,\ b^T(N\text{-}1)\ ]^T \qquad (2.4)$$

$$\boldsymbol{s} = [s^T(-L)\ ,\ \cdots\ ,\ s^T(N\text{-}1)\ ]^T \qquad (2.5)$$

et :

$$\boldsymbol{T}_N(h) = \begin{bmatrix} \boldsymbol{h}(M) & \cdots & \boldsymbol{h}(0) & & 0 \\ & \ddots & & \ddots & \\ 0 & & \boldsymbol{h}(M) & \cdots & \boldsymbol{h}(0) \end{bmatrix} \qquad (2.6)$$

$\boldsymbol{T}_N\ (h)$ est une matrice bloc-Toeplitz de N blocs (lignes) dépendant du vecteur $\boldsymbol{h}$ des coefficients du canal à estimer.

L'expression de la densité de probabilité de l'observation $\boldsymbol{y}$ s'écrit comme suit:

$$P(y) = \frac{1}{\pi^N \sigma^{2N}} \exp\left(\frac{1}{\sigma^2} \left\| \boldsymbol{y} - \boldsymbol{T}_N(h)\, \boldsymbol{s}(n) \right\|^2 \right) \qquad (2.7)$$

car le bruit $\boldsymbol{b}$ dans l'expression (2.2) est blanc gaussien.

Les estimateurs de $\boldsymbol{h}$ et $\boldsymbol{s}$, au sens du maximum de vraisemblance, sont ceux qui maximisent la densité de probabilité (2.7), tel que:

$$(\boldsymbol{h},\boldsymbol{s})_{MV} = \arg\max_{(h,s)} P(y) = \arg\max_{(h,s)} \left\| \ \boldsymbol{y} - \boldsymbol{T}_N(h)\, \boldsymbol{s} \ \right\|^2 \qquad (2.8)$$

Pour un $\boldsymbol{h}$ fixé, l'expression de l'estimée de $\boldsymbol{s}$ s'écrit:

$$\boldsymbol{s}_{MV} = [\boldsymbol{T}_N(h)^H \boldsymbol{T}_N(h)]^{-1} \boldsymbol{T}_N(h)^{\mathrm{H}} \boldsymbol{y} \qquad (2.9)$$

où l'inverse de la matrice Toeplitz existe sous les conditions d'identifiabilités donnés dans **[6]**. En remplaçant l'expression de $\boldsymbol{s}$ dans (2.8) par celle de (2.9) on obtient l'expression de l'estimé de $\boldsymbol{h}$ suivante:

$$\boldsymbol{h}_{MV} = \arg\min_{h} \left\| \left(\boldsymbol{I} - \boldsymbol{P}_h\right) \boldsymbol{y} \right\|^2 \qquad (2.10)$$

tel que $\boldsymbol{P}_h$ est la matrice de projection orthogonale sur l'espace image de $\boldsymbol{T}_N(\boldsymbol{h})$, donnée par :

$$\boldsymbol{P}_h = \boldsymbol{T}_N(h)\,[\boldsymbol{T}_N(h)^H \boldsymbol{T}_N(h)]^{-1}\boldsymbol{T}_N(h)^H \qquad (2.11)$$

De l'expression (2.11), on voit bien que le calcul de $\boldsymbol{h}_{MV}$ rencontre un problème fortement non-linéaire. Pour simplifier ce calcul, on peut procéder comme suit:

On définit le vecteur polynomial $h_{i,j}$ de taille $1\times L$, dont le $i^{ème}$ et le $j^{éme}$ élément sont donnés par $-h_j$ et h_i respectivement.

$$\boldsymbol{h}_{i,j}(z) = [\,0\,,\cdots,\,0\,,-h_j(z)\,,\,0\,,\,\cdots\,,0\,,h_i(z)\,,0\,,\,\cdots\,,0\,] \qquad (2.12)$$

Il est clair que la relation suivante est toujours satisfaite :

$$\boldsymbol{h}_{i,j}(z)\,\boldsymbol{h}(z) = -h_j(z)\,h_i(z) + h_i(z)\,h_j(z) = 0 \qquad (2.13)$$

Soit la matrice polynomiale $\boldsymbol{G}(z)$ construite comme suit :

$$\boldsymbol{G}(z)=\left[\,\boldsymbol{h}_{1,2}^T(z)\,,\,\ldots\,,\,\boldsymbol{h}_{1,L}^T(z),\,\boldsymbol{h}_{2,3}^T(z)\,,\,\ldots\,,\,\boldsymbol{h}_{2,L}^T(z),\,\ldots\,,\,\boldsymbol{h}_{L\text{-}1,L}^T(z)\right]^T \qquad (2.14)$$

cette matrice satisfait donc :

$$\boldsymbol{G}(z)\,\boldsymbol{h}(z) = 0 \Leftrightarrow \boldsymbol{G}^H\,\boldsymbol{T}_N(h) = 0 \qquad (2.15)$$

tel que $\boldsymbol{G}^H$ est la matrice du complément orthogonal de taille $L(L\text{-}1)(N\text{-}M)/\,2\times LN$ donnée par:

$$\boldsymbol{G}^H = \boldsymbol{T}_{N-M}(G) = \begin{bmatrix} \boldsymbol{G}(M) & \cdots & \boldsymbol{G}(0) & & 0 \\ & \ddots & & \ddots & \\ 0 & & \boldsymbol{G}(M) & \cdots & \boldsymbol{G}(0) \end{bmatrix} \qquad (2.16)$$

où $\boldsymbol{G}_k$ est le $k^{ième}$ coefficient de $\boldsymbol{G}(z)$.

avec cette définition nous avons le théorème suivant **[5]**:

Théorème : si les canaux h_i n'ont pas de zéros communs et si $N \geq 2M$, alors nous avons :

$$\boldsymbol{P}_h + \boldsymbol{P}_G = \boldsymbol{I}$$

où $\boldsymbol{I}$ matrice identité et $\boldsymbol{P}_h$, $\boldsymbol{P}_G$ sont les matrices de projection orthogonale sur l'espace image de $\boldsymbol{T}_N(h)$ et $\boldsymbol{G}$ respectivement.

Basé sur ce théorème, la minimisation de (2.10) devient:

$$\boldsymbol{h}_{MV} = \arg\min_{h} \left\| \boldsymbol{P}_G\,\boldsymbol{y} \right\|^2$$

$$\boldsymbol{h}_{MV} = \arg\min_{h} \boldsymbol{y}^H \boldsymbol{P}_G\,\boldsymbol{y}$$

$$\boldsymbol{h}_{MV} = \arg\min_{h} \boldsymbol{y}^H\,\boldsymbol{G}\,(\boldsymbol{G}^H\boldsymbol{G})^{\#}\,\boldsymbol{G}^H\boldsymbol{y} \qquad (2.17)$$

où # désigne la pseudo inverse.

On peut encore reformuler l'expression (2.17) en utilisant la commutativité de la convolution, on peut écrire alors:

$$\boldsymbol{G}^H \boldsymbol{y} = \boldsymbol{Y}\boldsymbol{h} \qquad (2.18)$$

où $\boldsymbol{Y}$ est définie par:

$$\boldsymbol{Y}=\begin{bmatrix} \boldsymbol{Y}(M) & \cdots & \boldsymbol{Y}(0) \\ \vdots & & \vdots \\ \boldsymbol{Y}(N-1) & \cdots & \boldsymbol{Y}(N-M-1) \end{bmatrix}$$

$$\boldsymbol{Y}(k)=[\ y^T_{(1,2)}(k), \cdots, y^T_{(1,L)}(k), y^T_{(2,3)}(k), \cdots, y^T_{(L-1,L)}(k)\ ]^T$$

$$y_{(i,j)}(k)=[\ 0, \cdots, 0, -y_j(k), 0, \cdots, y_i(k), 0, \cdots, 0\ ] \qquad (2.19)$$

en combinant (2.18) et (2.17) on obtient

$$\boldsymbol{h}_{MV} = \arg\min_{\|h\|=1} \boldsymbol{h}^H \boldsymbol{Y}^H (\boldsymbol{G}^H \boldsymbol{G})^{\#}\ \boldsymbol{Y}\boldsymbol{h} \qquad (2.20)$$

cette expression suggère une minimisation à deux étapes :

étape 1 : $\boldsymbol{h}_c = \arg\min_{\|h\|=1}\ \boldsymbol{h}^H \boldsymbol{Y}^H \boldsymbol{Y}\boldsymbol{h}$

étape 2 : $\boldsymbol{h}_{MV} = \arg\min_{\|h\|=1}\ \boldsymbol{h}^H \boldsymbol{Y}^H (\boldsymbol{G}_c{}^H \boldsymbol{G}_c)^{\#}\ \boldsymbol{Y}\boldsymbol{h}$, où $\boldsymbol{G}_c$ correspond à la matrice $\boldsymbol{G}$ construite à partir de $\boldsymbol{h}_c$.

La première étape vient de l'équation (2.20) où l'on remplace la matrice de pondération $(\boldsymbol{G}^H\boldsymbol{G})^{\#}$ par la matrice identité. Dans **[5]** il est démontré que cette première étape de l'algorithme fournit une estimation exacte de $\boldsymbol{h}$ en absence de bruit ou dans le cas d'une observation infinie et un bruit additif blanc.
La deuxième étape fournit à fort rapport signal sur bruit l'estimée optimale de $\boldsymbol{h}$.

Le coût numérique est relativement élevé à cause de la pseudo-inversion de la matrice$(\boldsymbol{G}^H\boldsymbol{G})^{\#}$. Il existe des versions plus rapides, mais que nous ne traiterons pas dans ce rapport, qui utilisent la structure bloc-Toeplitz ou bande-limité de $(\boldsymbol{G}^H\boldsymbol{G})^{\#}$ pour le calcul rapide de la pseudo-inverse **[8,9]**.

II-2-2 Algorithme de la méthode des relations croisées:

La première étape de l'algorithme du maximum de vraisemblance décrite précédemment coïncide avec la méthode des relations croisées **[15]**. Ces relations s'expriment par :

$$h_j(k) * y_i(k) = h_i(k) * y_j(k) \qquad (2.21)$$

et résultent en un système d'équations linéaires en groupant toutes les relations croisées pour toutes les paires (i,j) qui s'écrit :

$$\boldsymbol{Yh} = 0 \qquad (2.22)$$

où $\boldsymbol{Y}$ est la matrice définie par les vecteurs de la relation (2.19). En présence du bruit, la solution de (2.22) est remplacée par la solution au sens des moindres carrés :

$$\boldsymbol{h}_{CR} = \arg\min_{\|\boldsymbol{h}\|=1} \boldsymbol{h}^H \boldsymbol{Y}^H \boldsymbol{Y}\boldsymbol{h} \qquad (2.23)$$

qui n'est autre que l'estimée de $\boldsymbol{h}$ donnée par la première étape de l'algorithme MV.

Remarque :

La méthode des relations croisées étant l'objet d'une grande partie du travail effectué, un chapitre lui est consacré où plus de détails sur cette méthode et ses versions améliorées y sont donnés.

II-2-3 Algorithme de la méthode du sous-espace:

La méthode sous-espace introduite dans **[6]** utilise la notion de variables spatio-temporelles définies par:

$$\begin{aligned} \boldsymbol{y}_n &= [\, \boldsymbol{y}^T(n) \;\; , \;\; \cdots \;\; , \;\; \boldsymbol{y}^T(n+W-1) \,]^T = \boldsymbol{T}_W(h)\, \boldsymbol{s}_n + \boldsymbol{b}_n \\ \boldsymbol{s}_n &= [\, \boldsymbol{s}^T(n-M) \;\; , \;\; \cdots \;\; , \;\; \boldsymbol{s}^T(n+W-1) \,]^T \end{aligned} \qquad (2.24)$$

où W (taille de la fenêtre temporelle dont on verra l'utilité ultérieurement) est un paramètre entier que l'on choisit.

Principe de la méthode:

Si l'on arrive à trouver un vecteur qui appartient à un sous-espace de l'espace image de $\boldsymbol{T}_W(h)$, alors forcément ce vecteur définit, à un scalaire près, la réponse impulsionnelle du système, et ce grâce au théorème suivant:

Théorème: Soit $\boldsymbol{T}_W(h')$ une autre matrice bloc-Toeplitz de taille $LW\times(M+W)$ correspondant à une réponse impulsionnelle $\boldsymbol{h}'(z)$.

Si $W \geq M+1$ et si les L canaux $h_i(z)$, $i=1,...,L$ ne partagent pas de zéros communs, alors $\boldsymbol{T}_W(h)$ est de rang colonne plein et la relation

$$\text{Image}(\mathrm{T}_N(h')) \subset \text{Image}(\mathrm{T}_N(h)) \qquad (2.25)$$

N'est vérifiée que si les vecteurs $\boldsymbol{h}'$ et $\boldsymbol{h}$ sont linéairement dépendants, i.e:

$$\exists\, \alpha \in \Re \text{ , tel que } \boldsymbol{h}'(z) = \alpha\, \boldsymbol{h}(z) \qquad (2.26)$$

pour construire le sous-espace orthogonal à l'image de $\boldsymbol{T}_W(h)$ (dit sous-espace bruit), l'algorithme sous-espace construit la matrice de covariance de $y(n)$:

$$\boldsymbol{R}_y = \frac{1}{N-W+1} \sum_{n=0}^{N-W} \boldsymbol{y}(n)\, \boldsymbol{y}^H(n) \qquad (2.27)$$

En considérant le modèle non bruité, cette matrice possède la structure suivante:

$$\boldsymbol{R}_y = \boldsymbol{T}_W(h)\, \boldsymbol{R}_s\, \boldsymbol{T}_W(h)^H \qquad (2.28)$$

où

$$\boldsymbol{R}_s = \frac{1}{N-W+1} \sum_{n=0}^{N-W} \boldsymbol{s}(n)\, \boldsymbol{s}^H(n) \qquad (2.29)$$

si $\boldsymbol{R}_s$ est de rang plein, on peut démontrer que l'espace image de $\boldsymbol{R}_y$ coïncide avec l'espace image de $\boldsymbol{T}_W(h)$, donc trouver un espace orthogonale à l'espace image de $\boldsymbol{T}_W(h)$ revient à calculer l'espace orthogonale à l'espace image de $\boldsymbol{R}_y$. Par conséquent le sous-espace signal (espace image de $\boldsymbol{T}_W(h)$) et le sous-espace bruit peuvent être calculés par la décomposition propre de $\boldsymbol{R}_y$:

$$\boldsymbol{R}_y = \boldsymbol{E}\, \Lambda \boldsymbol{E}^H \qquad (2.30)$$

où $\boldsymbol{E}$ et Λ sont les matrices vecteurs propres et valeurs propres,

$$\boldsymbol{E} = [\, \boldsymbol{e}_1 , \ \cdots \ , \boldsymbol{e}_{LW} \,] \qquad (2.31)$$

$$\Lambda = diag\, (\, \lambda_1, \ \cdots \ , \lambda_{LW}) \qquad (2.32)$$

où les valeurs propres sont rangées en ordre descendant:

$$\lambda_1 \geq \cdots \geq \lambda_{M+W} \geq \lambda_{M+W+1} = \cdots = \lambda_{LW} = 0 \qquad (2.33)$$

il en résulte que

$$\text{Image}\, ([\boldsymbol{e}_1 , \ \cdots \ , \boldsymbol{e}_{M+W}\,]) = \text{Image}\, (\boldsymbol{T}_W(h)) \qquad (2.34)$$

et

$$\text{Image}\, (\boldsymbol{T}_W(h)) \perp \text{Image}\, ([\, \boldsymbol{e}_{M+W+1} , \ \cdots \ , \boldsymbol{e}_{LW}\,])$$

car les valeurs propres (λ_{M+W+1}, $\cdots$, λ_{LW}) étant nulles, les vecteurs propres qui leurs sont associés génèrent un espace orthogonal à l'espace de $\boldsymbol{R}_y$, et donc à l'espace de $\boldsymbol{T}_W(h)$.

Cette relation nous permet d'identifier $\boldsymbol{h}$ à un scalaire près comme le vecteur satisfaisant :

$$E_n^H \, T_W(h) = 0 \qquad (2.35)$$

où

$$E_n = ([\, e_{M+W+1} \,, \; \cdots \;, e_{LW} \,]) \qquad (2.36)$$

En prenant en considération le bruit additif, l'algorithme sous-espace résout (2.35) au sens des moindres carrés, i.e:

$$\boldsymbol{h} = \arg\min_{\|\boldsymbol{h}\|=1} \left\| E_n^H \, T_W(h) \right\|^2 \qquad (2.37)$$

le résultat obtenu est égale à $\boldsymbol{h}$ exact si le bruit additif est spatialement blanc et si le nombre d'échantillons N tend vers l'infini **[10].**

II-3 Algorithmes d'égalisation directe :

II-3-1 Algorithme de la méthode du sous-espace source :

Considérons la matrice des observations (non bruitées) donnée par :

$$[\, y_0 \,, \; \cdots \;, y_{N-W} \,] = T_W(h) \, S_{W+M} \qquad (2.38)$$

où :

$$S_{W+M} = \begin{bmatrix} s_{-M} & s_{-M+1} & \cdots & s_{N-W-M} \\ s_{-M+1} & s_{-M+2} & \cdots & s_{N-W-M+1} \\ \vdots & \vdots & & \vdots \\ s_{W-1} & s_W & \cdots & s_{N-1} \end{bmatrix} \qquad (2.39)$$

S_{W+M} est une matrice de Hankel de dimension $(W+M)\text{x}(N-W+1)$.

Le sous-espace défini par les vecteurs lignes de S_{W+M} est appelé sous-espace source.

Lorsque la matrice du système $T_W(h)$ est de rang colonne plein, la matrice S_{W+M} et celle des observations génèrent le même sous-espace ligne (source). Soit V_0 la matrice générant le sous-espace orthogonal au sous-espace source, i.e.

$$S_{W+M} \, V_0 = 0 \qquad (2.40)$$

la méthode sous-espace **[12,13,14]** estime la séquence d'entrée $\boldsymbol{s}$ en utilisant la relation d'orthogonalité (2.40). Ceci est possible grâce au théorème suivant :

Théorème: Supposons qu'il existe W tel que $\boldsymbol{T}_W(h)$ est de rang colonne plein et que la séquence $\{s_n\}_{-M \le n \le N-1}$ possède plus de $W+M+1$ modes[1]. Alors, l'unique solution (à un scalaire près) à la relation d'orthogonalité (2.40) est le vecteur $\boldsymbol{s} = [\, s_{-M}, \cdots, s_{N-1} \,]^T$.

Pour résoudre (2.40), on utilise la structure de Hankel en (2.39) où l'on construit le noyau de $\boldsymbol{S}_{W+M-1}$ à partir de celui de $\boldsymbol{S}_{W+M}$ comme suit **[12,13]**

$$\begin{bmatrix} \boldsymbol{V}_0 & 0 \\ 0 & \boldsymbol{V}_0 \end{bmatrix}$$

où 0 est un bloc de zéros de taille $1 \times (N-2W-M+1)$. Plus généralement, le noyau de $\boldsymbol{S}_k$, $k=W+M-j+1$, $j=1, \cdots, W+M$, désigné par $\boldsymbol{V}_k$ a la forme suivante :

$$\boldsymbol{V}_k = \begin{bmatrix} \boldsymbol{V}_0 & & 0 \\ & \ddots & \\ 0 & & \boldsymbol{V}_0 \end{bmatrix} \qquad (2.41)$$

Ceci conduit en particulier à l'équation linéaire suivante :

$$\boldsymbol{s}^T \boldsymbol{V}_1 = 0 \quad \text{où} \quad \boldsymbol{S}_1 = [\, s_{-M}, \cdots, s_{N-1} \,] = \boldsymbol{s}^T \qquad (2.42)$$

que l'on résout au sens des moindres carrés pour prendre en compte le bruit additif.

En résumé, l'algorithme sous-espace source est mis en œuvre comme suit (pour plus de détails voir **[12,13]** ou **[14]**) :

- Calculer la matrice $\boldsymbol{V}_0$ du noyau de $\boldsymbol{S}_{W+M}$ à partir d'une décomposition en valeurs singulières de la matrice des observations.
- Construire $\boldsymbol{V}_1$ comme indiqué par (2.41).
- Résoudre au sens des moindres carrés l'équation (2.42) pour obtenir une estimée (à un scalaire près du vecteur source $\boldsymbol{s}$.

[1] : Un mode est une séquence de la forme $m_k = k^n z^k$ où n est un entier (ordre du mode) et z un nombre complexe (racine du mode).

II-3-2 algorithme de la méthode des égaliseurs mutuellement référencés (EMR) :

Cette méthode, introduite dans **[16,17]**, estime directement un ensemble d'égaliseurs 'forçage à zéro' dit mutuellement référencés. L'existence de tels égaliseurs relève de celle de l'inverse à gauche (pseudo-inverse) de la matrice du système $\boldsymbol{T}_W(h)$ lorsque celle-ci est de rang plein, i.e. :

$$\boldsymbol{T}_W^{\#}(h)\, \boldsymbol{y}_n = \boldsymbol{s}_n \qquad (2.43)$$

il est clair que le $i^{éme}$ vecteur ligne de $\boldsymbol{T}_W^{\#}(h)$ est un égaliseur 'forçage à zéro' associé au retard $i-1$. L'idée de la méthode EMR est alors la suivante : considérons les égaliseurs, de taille $LW\times 1$, $\boldsymbol{g}_i$ et $\boldsymbol{g}_{i+1}$ satisfaisant :

$$\begin{aligned} \boldsymbol{g}_i^H \boldsymbol{y}_n &= \boldsymbol{s}_{n-i} \\ \boldsymbol{g}_{i+1}^H \boldsymbol{y}_n &= \boldsymbol{s}_{n-i-1} \end{aligned} \qquad (2.44)$$

il en découle que

$$\boldsymbol{g}_i^H \boldsymbol{y}_n = \boldsymbol{g}_{i+1}^H \boldsymbol{y}_{n+1} \quad , i = 0, \ldots, W+M-2 \qquad (2.45)$$

où les sorties des 2 égaliseurs sont dites référencées l'une à l'autre. Dans **[16]**, il est montré que les $W+M-1$ relations (2.45) suffisent pour que $\boldsymbol{g}_0, \ldots, \boldsymbol{g}_{W+M-1}$ soient des égaliseurs 'forçage à zéro'. Nous avons le théorème suivant **[16]**

Théorème : Soit W un entier positif tel que $\boldsymbol{T}_W(h)$ et $\boldsymbol{S}_{W+M}$ soient de rang colonne et de rang ligne plein respectivement. Soient $\boldsymbol{g}_0, \ldots, \boldsymbol{g}_{W+M-1}$, $W+M$ vecteurs de tailles $WL\times 1$ et vérifiant $\boldsymbol{g}_i^H \boldsymbol{y}_n = \boldsymbol{g}_{i+1}^H \boldsymbol{y}_{n+1}$ pour tout i et tout n. On a alors

$$\boldsymbol{G}^H \boldsymbol{T}_W(h) = \alpha \boldsymbol{I} \qquad (2.46)$$

où $\boldsymbol{G} \overset{def}{=} [\boldsymbol{g}_0, \ldots, \boldsymbol{g}_{W+M-1}]$ et α est une constante scalaire.

En pratique, on estime $\boldsymbol{G}$ de façon à minimiser le critère quadratique suivant

$$J(G) = \sum_{n=0}^{N-W} \left\| \boldsymbol{E}(n) \right\|^2$$

où

$$\boldsymbol{E}(n) = [\boldsymbol{I}, \boldsymbol{0}]\, \boldsymbol{G}^H \boldsymbol{y}_n - [\boldsymbol{0}, \boldsymbol{I}]\, \boldsymbol{G}^H \boldsymbol{y}_{n+1}$$

($\boldsymbol{0}$ etant le vecteur nul de dimension $(M+W-1)\times 1$)
sous une contrainte appropriée, e.g. $Trace(\boldsymbol{G}^H\boldsymbol{G})=1$. Le rôle de la contrainte n'est pas seulement d'éviter la solution triviale $\boldsymbol{G}=0$ mais aussi d'éviter que $\boldsymbol{G}$ ne soit une matrice dite de blocage,i.e. satisfaisant $\boldsymbol{G}^H\boldsymbol{T}_W(h)=0$, qui correspond à $\alpha = 0$. Dans **[18]**, des contraintes linéaires ou quadratiques sont considérées pour garder la procédure de minimisation aussi

simple que possible. Les différents choix de contraintes correspondent à différentes implémentations et différentes performances de l'algorithme EMR **[18]**.

II-3-3 Algorithme de la méthode de prédiction linéaire (PL) :

Pour cette méthode nous supposerons que les systèmes d'entrée sont des variables aléatoires à moyenne nulle et temporellement décorrélés. Nous supposerons aussi, comme pour les méthodes précédentes, que les canaux $h_i(z)$, $i = 1,\ldots,L$ ne partagent pas de zéro en commun. Le signal observé (sans bruit) suit alors un modèle MA (moyenne ajustée) donné par

$$y(n) = \sum_{k=0}^{M} \boldsymbol{h}(k)\, s(n-k) \qquad (2.47)$$

L'idée de la méthode PL dans le contexte de la diversité spatio-temporelle a été présentée pour la première fois par Slock **[19]**. Elle consiste à observer que le signal observé $y(n)$ est aussi un signal auto-regressif (AR).
Cette propriété découle de l'identité de Bezout qui stipule que, du fait que les $h_i(z)$ sont globalement premiers entre eux (pas de zéros communs à tous), il existe un vecteur polynomial $\boldsymbol{g}(z)$ de taille $1\times L$ et de degré M tel que

$$\boldsymbol{g}(z)\, \boldsymbol{h}(z) = 1 \qquad (2.48)$$

En appliquant donc $\boldsymbol{g}$ à $y(n)$ on obtient

$$\sum_{k=0}^{M} \boldsymbol{g}(k)\, y(n-k) = \boldsymbol{s}(n) \qquad (2.49)$$

autrement dit, $\boldsymbol{h}(z)$ peut être exactement inversé par un filtre causal RIF.

Plus rigoureusement, on peut traduire la relation (2.49) de la façon suivante :

$$H_{n-1}(y) = span\{\, y_i(n-l) \,/\, i = 1, \ldots, L \,;\, l \geq 1\} \qquad (2.50)$$

Ici, $\text{span}\{x_i \in I\}$ représente l'espace de Hilbert généré par $\{x_i \in I\}$. Similairement, soit $\mathrm{H}_{n-1,P}(y)$ l'espace généré par les observations passées de retard inférieur à P, i.e.,

$$H_{n-1,P}(y) = span\{\, y_i(n-l) \,/\, i = 1, \ldots, L \,;\, 1 \leq l \leq P\} \qquad (2.51)$$

Le processus d'innovation $\boldsymbol{i}(n)_{n\in Z}$ de $y(n)$ est le bruit blanc L-dimensionnel défini par

$$\boldsymbol{i}(n) = y(n) - y(n)|_{\,Hn-1(y)} \qquad (2.52)$$

où | désigne la projection orthogonale sur le sous espace H. Le terme $y(n)|_{\,Hn-1(y)}$. est en fait la prédiction linéaire de $y(n)$. Le processus $y(n)$ est dit être autorégressif d'ordre P si $\boldsymbol{i}(n)$ coïncide avec l'innovation d'ordre fini $\boldsymbol{i}_P(n) = y(n) - y(n)|_{\,Hn-1,P(y)}$. Nous avons le théorème suivant **[20,21]** :

Théorème : le signal (sans bruit) $y(n)$ est un processus autorégressif d'ordre M (M étant le degré de $\boldsymbol{h}(z)$). Son processus d'innovation est donné par $i(n) = \boldsymbol{h}(0)\, s(n)$.

L'innovation $\boldsymbol{i}(n)$ est calculée par projection de $\boldsymbol{y}(n)$ dans le sous-espace généré par $\{ y_i(n-l) / i = 1, \ldots, L ; l = 1, \ldots, P \}$ où $P \geq M$. Soit $[\boldsymbol{A}_1, \ldots, \boldsymbol{A}_P]$ les matrices coefficient de prédiction linéaire de taille LxL tels que $y(n)+\sum_{k=1}^{P} \boldsymbol{A}_k\, y(n-k) = \boldsymbol{i}(n)$. Il est alors facile de montrer que $\boldsymbol{A}_1, \ldots, \boldsymbol{A}_P$, satisfont (équation de Yule et Walker)

$$[\boldsymbol{A}_1, \ldots, \boldsymbol{A}_P]\, \boldsymbol{R} = -[\boldsymbol{R}_1, \ldots, \boldsymbol{R}_P] \qquad (2.53)$$

où $\boldsymbol{R}$ est la matrice LPxLP définie par

$$\boldsymbol{R} = \begin{bmatrix} \boldsymbol{R}_0 & \boldsymbol{R}_1 & \cdots & \boldsymbol{R}_{P-1} \\ \boldsymbol{R}_{-1} & \boldsymbol{R}_0 & \cdots & \boldsymbol{R}_{P-2} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{R}_{1-P} & \boldsymbol{R}_{2-P} & \cdots & \boldsymbol{R}_0 \end{bmatrix}$$

avec $\boldsymbol{R}_k \overset{def}{=} E\{ y(n+k)\, y(n)^H \}$. Puisque $\boldsymbol{R}$ n'est pas de rang plein **[20]**, les coefficients de prédiction linéaire satisfaisant (2.53) ne sont pas uniques. Une solution de (2.53) au sens des moindres carrés est donnée par :

$$[\boldsymbol{A}_1, \ldots, \boldsymbol{A}_P] = -[\boldsymbol{R}_1, \ldots, \boldsymbol{R}_P]\, \boldsymbol{R}^{\#} \qquad (2.54)$$

en se basant sur le théorème précèdent, la covariance de l'innovation $\boldsymbol{D} \overset{def}{=} E\{ \boldsymbol{i}(n)\, \boldsymbol{i}(n)^H \}$ est donnée par $\boldsymbol{D} = \sigma_s^2\, \boldsymbol{h}(0)\boldsymbol{h}(0)^H$, où σ_s^2 désigne la puissance du signal source.
La valeur propre non-nulle de $\boldsymbol{D}$ est égale à $\lambda_d = \sigma_s^2\, \|\boldsymbol{h}(0)\|^2$ et le vecteur propre associée est $\boldsymbol{d} = \boldsymbol{h}(0) / \|\boldsymbol{h}(0)\|$. Soit $\boldsymbol{v} = \boldsymbol{d} / \sqrt{\lambda_d}$. Nous avons alors le résultat suivant **[20]**

Théorème : Le vecteur polynomial de taille 1xL défini par $\boldsymbol{g}(z) = \boldsymbol{v}^H (\boldsymbol{I} + \sum_{k=1}^{P} \boldsymbol{A}_k z^{-k})$ satisfait $[\boldsymbol{g}(z)]\, \boldsymbol{y}(n) = \boldsymbol{s}(n)$.

L'algorithme prédiction linéaire se résume donc comme suit :

- Estimation des matrices de covariance
$$\boldsymbol{R}_k = \frac{1}{N-k} \sum_{i=0}^{N-k-1} y(n+k)\, y^H(n) \;,\; k = 0, \ldots, P \geq M .$$
- Résolution des équations de Yule et Walker données par (2.53) ;
- Estimation de la covariance de l'innovation par $\boldsymbol{D} = \boldsymbol{R}_0 + \sum_{k=1}^{P} \boldsymbol{A}_k \boldsymbol{R}_k^H$

et du vecteur $\boldsymbol{v}$ obtenu à partir de la décomposition propre de $\boldsymbol{D}$;

- Détection des symboles $s(n)$ en utilisant le filtre de prédiction linéaire

$$\boldsymbol{g}(z) = \boldsymbol{v}^H \left(\boldsymbol{I} + \sum_{k=1}^{P} \boldsymbol{A}_k z^{-k} \right)$$

A noter que dans le cas d'un bruit additif blanc, il suffit de remplacer la covariance $\boldsymbol{R}_0$ par sa version débruitée $\boldsymbol{R}_0 - \sigma^2 \boldsymbol{I}$, où σ^2 est la puissance du bruit.

II-4 Conclusion :

Dans ce chapitre, nos avons présenté un ensemble de méthodes et d'algorithmes pour l'identification ou l'égalisation de systèmes RIF (Réponse Impultionnelle Finie) à une entrée et plusieurs sorties (multicapteurs) [1].

Les méthodes étudiées sont basées sur les statistiques du second ordre de l'observation. Elles se présentent sous deux catégories:

- Les méthodes qui identifient directement le canal, et correspondent à la méthode: du maximum de vraisemblance, des relations croisées, du sous espace canal.
- Les méthodes qui identifient directement le signal, celles-ci correspondent à la méthode: de sous espace signal, prédiction linéaire, l'égaliseur mutuellement référencé.

Nous nous sommes contenté d'une description du principe et l'implantation standard de chacune des méthodes considérées sans citer leur différentes versions. Nous invitons le lecteur intéressé par plus de détails à consulter la liste bibliographique de ces méthodes.
Dans le chapitre suivant, nous décrivons la plate forme logicielle que nous avons réalisé et qui permet de comparer et d'étudier toutes les méthodes citées dans un environnement ergonomique agréable.

[1] : ces systèmes sont appelés sustèmes SIMO (single input multiple output).

Chapitre III :

Présentation de la plate forme logicielle

III-1 Description de la plate forme 27

III-2 Simulation 30

III-3 Conclusion 34

Chapitre 3 :

Présentation de la plate forme logicielle

Introduction:

Dans ce qui suit nous allons présenter une plate forme logicielle réalisée sous Matlab pour l'évaluation des algorithmes d'égalisation aveugle multicapteurs, ainsi que les différentes fonctionnalités dont elle dispose.

Cette plate forme offre la possibilité d'évaluer les performances des algorithmes individuellement ou deux à deux et d'ajouter de nouveau algorithmes, ce qui donne un large éventail de choix à l'utilisateur.

III-1 Description de la plate forme :

Cette interface permet d'exécuter des algorithmes d'égalisation aveugle multicapteurs et de déterminer et comparer leurs performances suivant les critères présents sur l'interface.

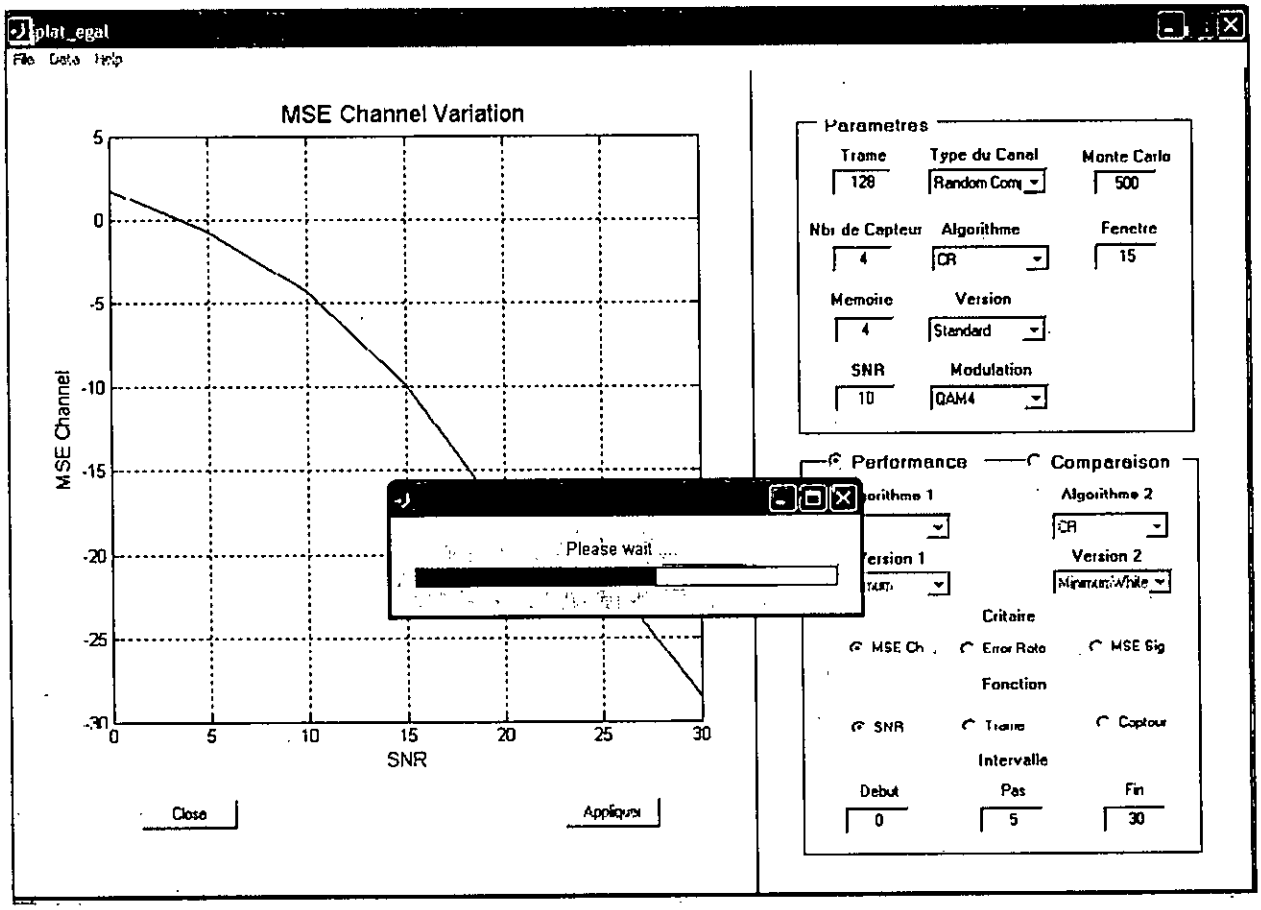


Fig.3. 1 : Plate forme

Pour la sélection et la modification des paramètres de simulation, l'interface dispose des champs et menus suivants:

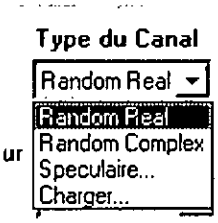
- Champs "Trame" : permet de déterminer la taille de l'observation (trame).

- Champs "Nbr de Capteurs" : permet de définir le nombre de capteurs.

- Champs "Mémoire" : permet de déterminer le degrés du canal .

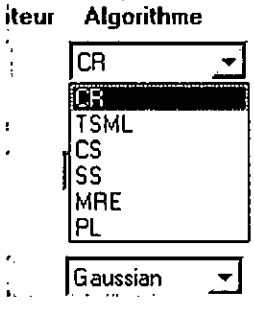
- Champs "SNR" : permet de fixer la valeur du rapport signal sur bruit en dB.

- Menu "Type du Canal" : offre le choix entre 4 possibilités :
  a- Random Real : canaux à coefficients aléatoires réels suivant une loi Gaussienne.
  b- Random complex : canaux à coefficients aléatoires complexes suivant une loi Gaussienne.
  c- Spéculaire : canaux à coefficients aléatoires suivant un modèle spéculaire (eq.(3.2)).
  d- Charger : chargement de nouveaux canaux écrits dans des fichiers **.mat** sous la structure suivante:

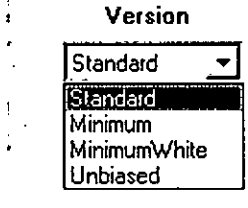


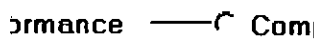
$$\begin{bmatrix} h_1(0) & \cdots & h_1(M) \\ \vdots & & \vdots \\ h_L(0) & \cdots & h_L(M) \end{bmatrix}$$

- Menu "Algorithme" : permet de définir l'algorithme à appliquer.
  Les algorithmes proposés sont:
  a- Algorithmes d'identification du canal :
  Algorithme TSML (Maximum de Vraissemblance) ;
  Algorithme CS (Sous-espace Canal) ;
  Algorithme CR (Relations Croisées).
  b- Algorithmes d'égalisation du signal source :
  Algorithme SS (Sous-espace Signal);
  Algorithme MRE (Egaliseurs Mutuellement référencés) ;
  Algorithme PL (Prédiction linéaire).



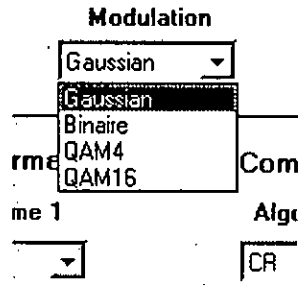
- Menu "Version" : permet de déterminer la version de l'algorithme choisi. Par défaut la version standard est la version présente sur le menu au démarrage.





- Menu "Modulation" : offre le choix entre 4 types de modulation pour le signal source :

a- Gaussien : les symboles suivent une loi Gaussienne complexe circulaire ;
b- Binaire : les symboles appartiennent à l'alphabet {-1,1} ;
c- QAM 4 : les symboles suivent une modulation QAM4 ;
d- QAM 16 : les symboles suivent une modulation QAM16.



- Champs "Monte Carlo" : permet de déterminer le nombre d'expériences à effectuer pour le calcul de l'erreur quadratique moyenne et le nombre d'erreur pour le calcul du taux d'erreur par symbole.

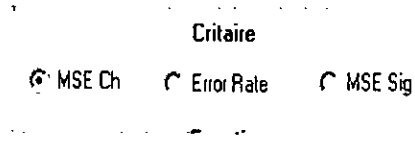
- Champs "Fenêtre" : permet de déterminer la taille de la fenêtre temporelle pour certain algorithmes (ex: Sous-espace Canal)

- Radio-bouton "Performance" : pour le cas de l'évaluation des performances d'un seul algorithme.

- Radio-bouton "Comparaison" : pour le cas d'une comparaison entre deux algorithmes choisis dans les menus "Algorithme1" et "Algorithme2" ainsi que leurs versions respectives choisies dans les menus "Version1" et "Version2".

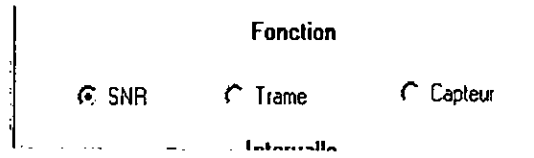
- Radio-bouton "Critère" : permet de déterminer le critère d'évaluation.
Les critères proposés sont:

a- MSE Chan :Erreur Quadratique Moyenne pour l'identification du canal;
b- MSE Sig :Erreur Quadratique Moyenne pour l'égalisation du signal.
c- Error Rate :Taux d'erreur par symbole



- Radio-bouton "Fonction" : permet de déterminer la fonction par rapport à laquelle on évalue les performances. Les fonctions proposées sont:

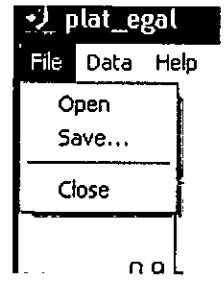
a- SNR (Rapport Signal sur Bruit);
b- Taille de la trame ;
c- Le nombre de capteurs.



- Champs "Intervalle" : permet de fixer l'intervalle de variation du paramètre choisi (SNR,Taille de la trame, Nombre de capteurs) en donnant le Min, Max et le Pas de variation

- Menu File : offre les fonctions suivantes :

Save : Sauvegarder les paramètres et les résultats d'une simulation dans un fichier *.mat;
Open : Ouvrir un fichier contenant des résultats sauvegardés.
Close : Fermer la fenêtre de la plate forme.



III-2 Simulations :

Dans ce paragraphe, nous allons présenter quelques simulations afin de montrer les fonctionnalités qui sont disponibles sur la plate forme .

- **Simulation 1 :**

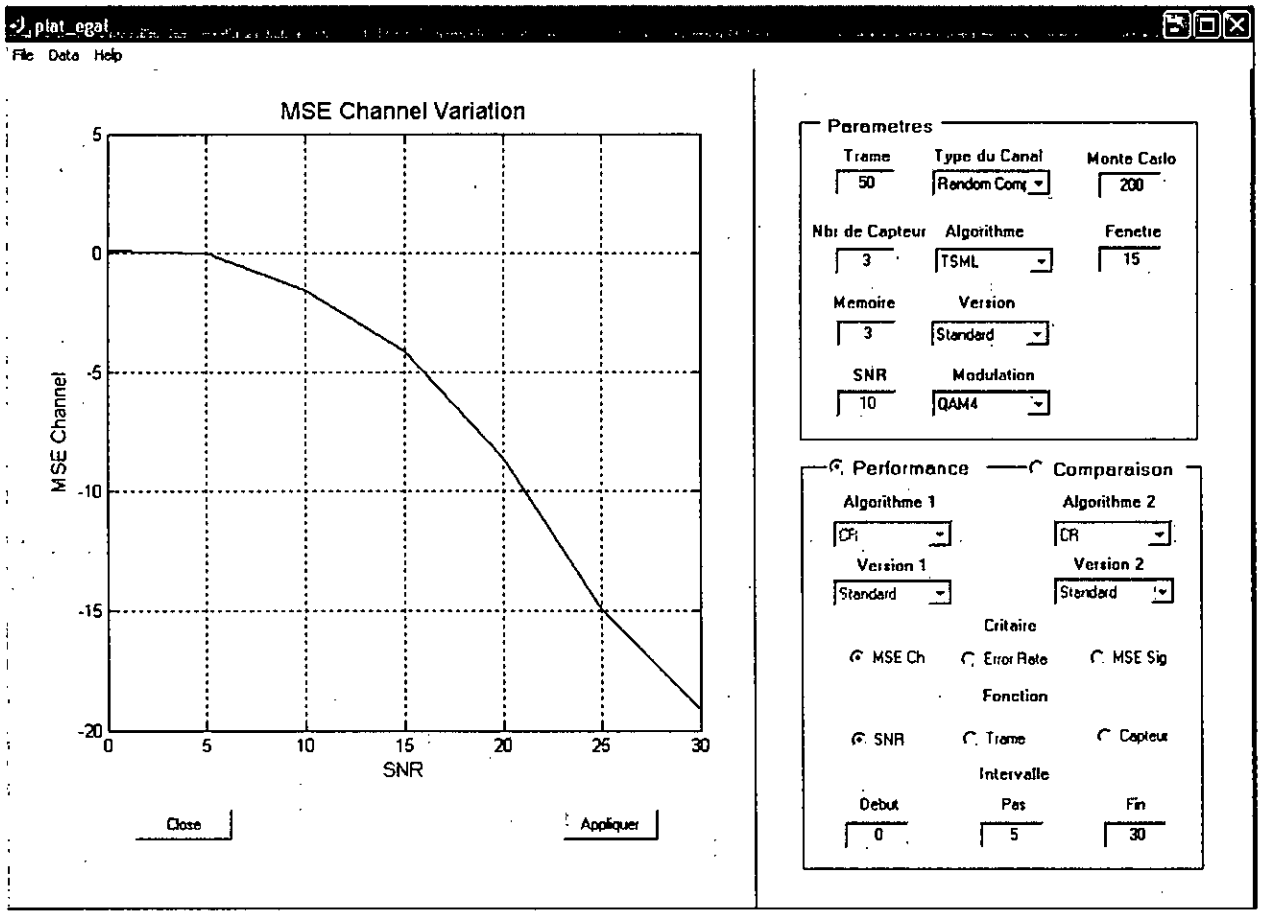


Fig.3. 2 : Simulation 1

Dans cette simulation nous présentons le résultat d'une étude de performance de l'algorithme TSML (Maximum de Vraisemblance). L'erreur quadratique moyenne MSE normalisée de l'estimée du canal définie par :

$$MSE=\frac{1}{N_r}\sum_{i=1}^{N_r}\frac{\left\| h-\hat{h}^{(i)}\right\|^2}{\left\| h\right\|^2} \qquad (3.1)$$

(où $N_r = 200$ est le nombre de Monte-Carlo et $\hat{h}^{(i)}$ est l'estimation de la $i^{éme}$ réalisation)
est représentée en fonction du SNR (rapport signal sur bruit) qui varie entre 0dB et 30dB .

Nous avons choisi pour cette simulation des signaux de modulation QAM4 et un nombre de symboles égal à 50 symboles par trame, ainsi qu'un tirage aléatoire à chaque expérience de L= 3 canaux complexes de mémoire M= 3 (4 coefficients par canal), le code Matlab correspondant et le suivant :

canaux=(randn(*L*,*M*+1)+i*randn(*L*,*M*+1))/sqrt(2)

- **Simulation 2 :**

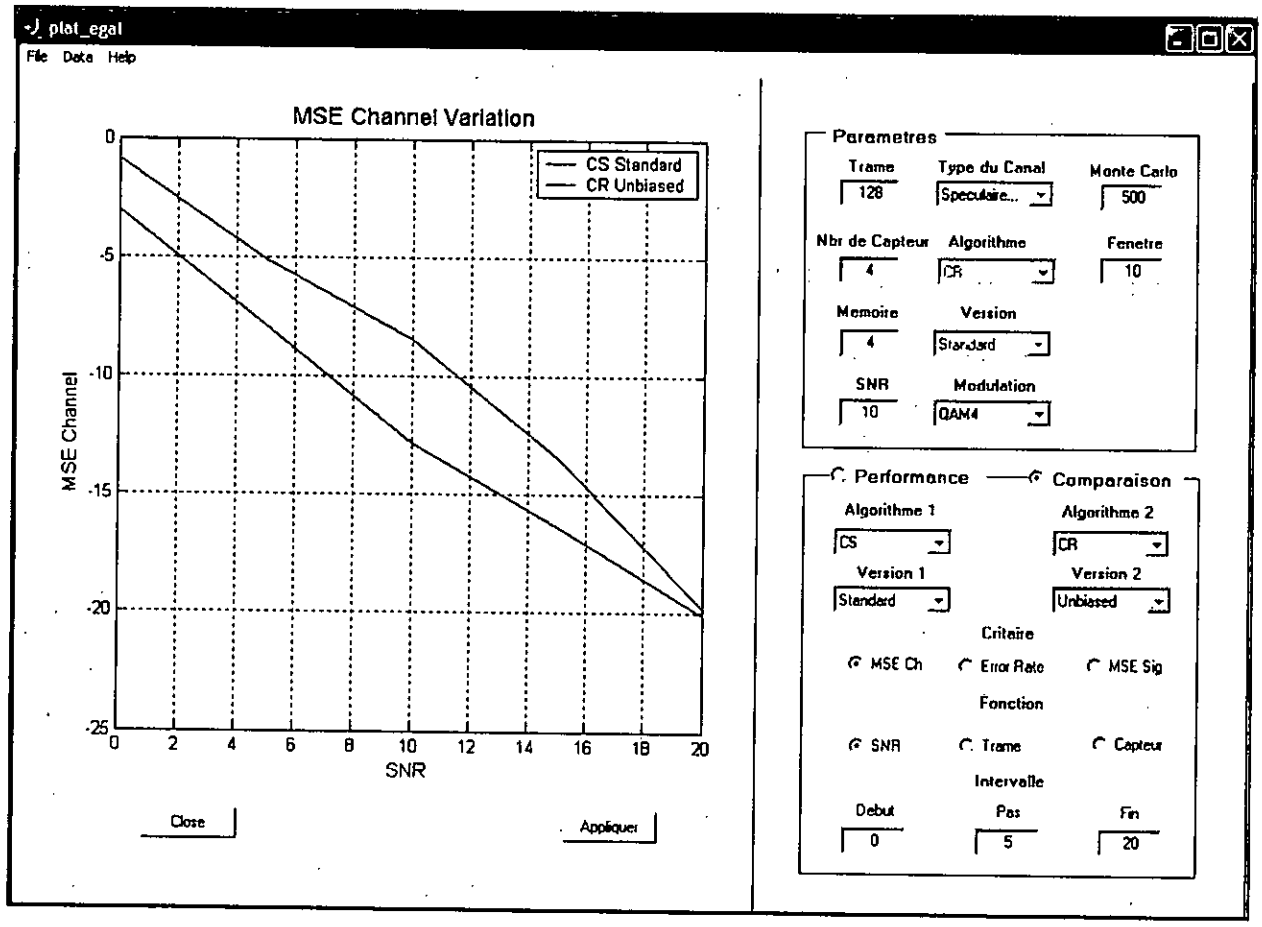


Fig.3. 3 : Simulation 2

Dans cette simulation nous présentons une comparaison des performances des algorithmes CR (Relations Croisées) et CS (Sous-espace Canal). L'erreur quadratique moyenne MSE normalisée de l'estimée du canal (estimé sur 500 Monte-Carlo) est représenté en fonction du SNR variant entre 0dB et 20dB.

Nous avons choisi pour cette simulation des signaux de modulation QAM4 et un nombre de symboles égal à 128 symboles par trame, ainsi qu'un tirage aléatoire à chaque expérience de L= 4 canaux spéculaires de mémoire M= 4 (5 coefficients par canal). Ces derniers sont définis suivant le model :

$$h(t)=\sum_{i=1}^{d}\lambda i\ a(\theta_i)\ g(t-\tau i) \qquad (3.2)$$

où :

d : est le nombre de trajets.

λ_i : est une variable aléatoire complexe gaussienne de puissance σ_i.

$a(\theta_i)$: est le vecteur $\left[1\ ,\ e^{j\pi \sin\theta_i}\ ,\ \cdots\ ,\ e^{j(L-1)\pi \sin\theta_i}\right]^T$.

$g(t)$: est une fonction cosinus surélevé.

Les paramètres $\sigma_i, \theta_i, \tau_i$ représentent respectivement les puissances, les angles d'arrives et les retards des trajets. Ces paramètres sont fournis par un fichier **.mat** correspondant au canal désiré.

- **Simulation 3 :**

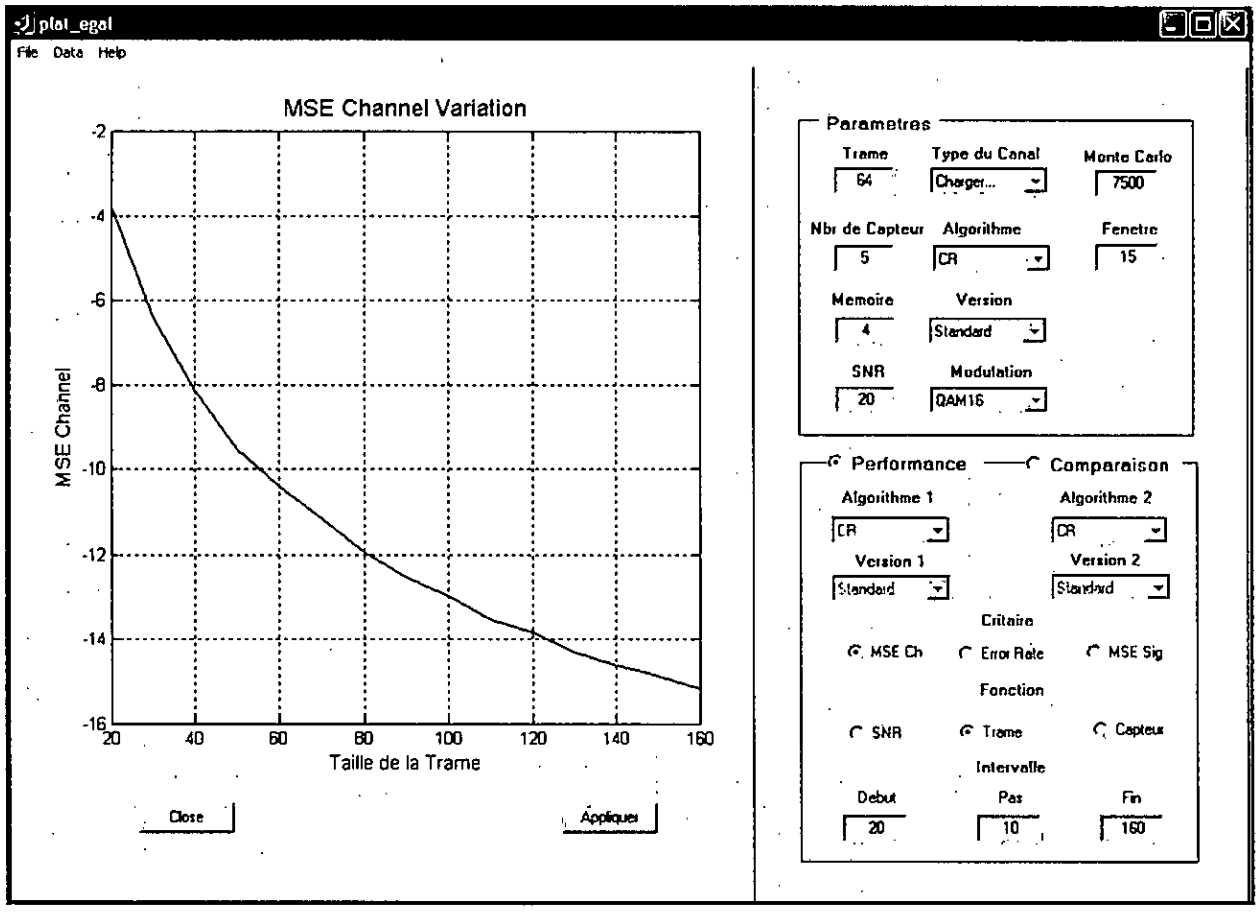


Fig.3. 4 : Simulation 3.

Dans cette simulation nous présentons le résultat d'une étude de performance de l'algorithme CR . L'erreur quadratique moyenne MSE normalisée de l'estimée du canal est représentée en fonction du nombre de symboles par trame qui varie entre 20 symboles et 160 symboles.
Nous avons choisi pour cette simulation des signaux de modulation QAM16 et un SNR égal à 20 dB, ainsi que L= 5 canaux complexes fixes de mémoire M= 4 (5 coefficients) chargés à partir d'un fichier **.mat**.

- **Simulation 4 :**

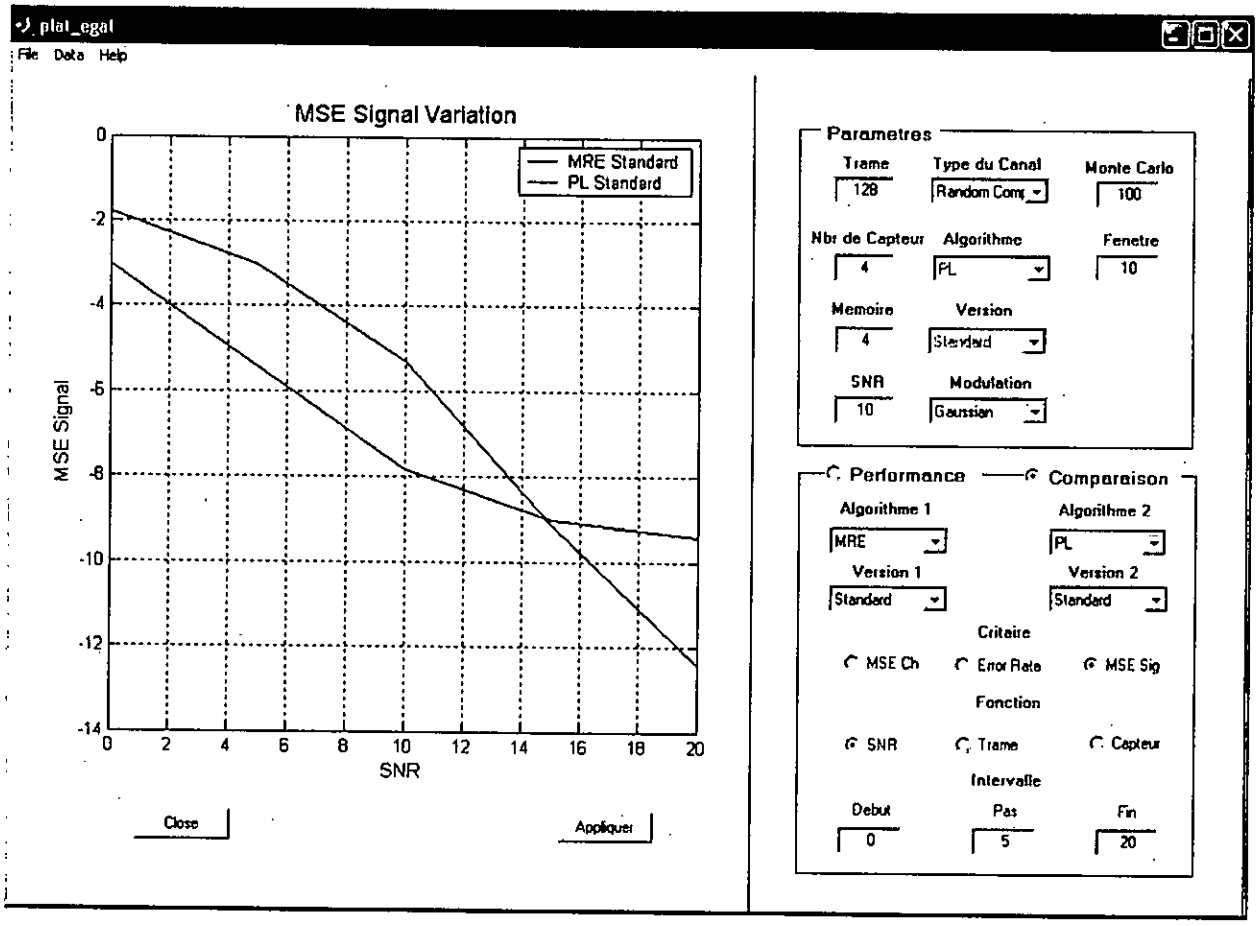


Fig.3. 5 : Simulation 4

Dans cette simulation nous présentons une comparaison de performances des algorithmes MRE (Egaliseurs Mutuellement Référencés) et PL (Prédiction linéaire). L'erreur quadratique moyenne MSE normalisée de l'estimée du signal définie par :

$$MSE = \frac{1}{N_r} \sum_{i=1}^{N_r} \frac{\left\| s - \hat{s}^{(i)} \right\|^2}{\left\| s \right\|^2} \qquad (3.3)$$

(où $N_r = 100$ est le nombre de Monte-Carlo et $\hat{s}^{(i)}$ est l'estimation de la $i^{éme}$ réalisation) est représentée en fonction du SNR (rapport signal sur bruit) qui varie entre 0dB et 20dB .

Nous avons choisi pour cette simulation des signaux aléatoires gaussiens et un nombre de symboles égal à 128 symboles par trame, ainsi qu'un tirage aléatoire à chaque expérience de $L = 4$ canaux complexes de mémoire $M = 4$ (5 coefficients par canal).

- **Simulation 5 :**

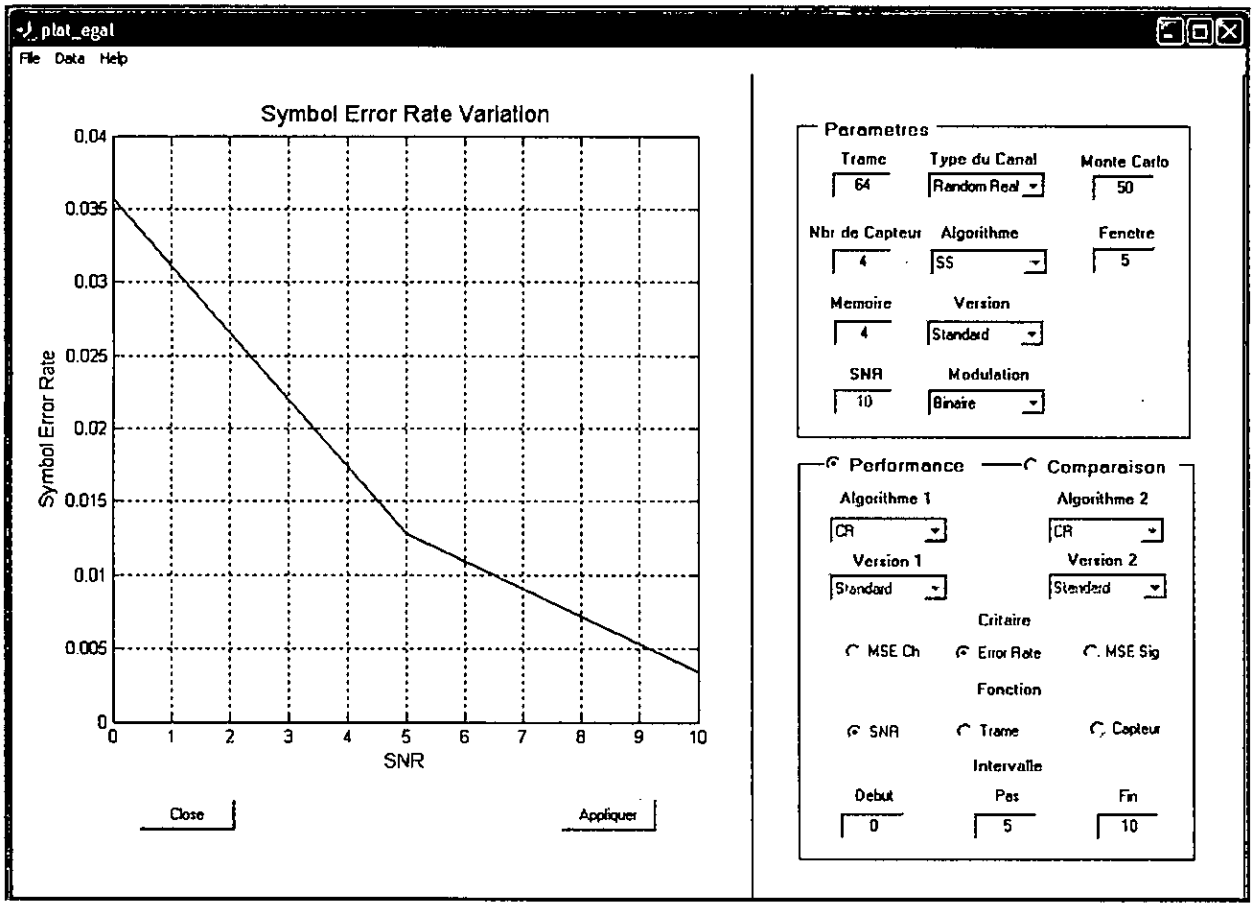


Fig.3. 6 : Simulation 5

Dans cette simulation nous présentons le résultat d'une étude de performance de l'algorithme SS (Signal Subspace) . Le taux d'erreurs par bit est représenté en fonction du SNR qui varie entre 0 dB et 10 dB.

Nous avons choisi pour cette simulation des signaux binaires et un nombre de symboles égal à 64 symboles par trame, ainsi qu'un tirage aléatoire à chaque expérience de L= 4 canaux réels de mémoire M= 4 (5 coefficients par canal), le code Matlab correspondant est le suivant :

canaux=randn(L,M+1).

III-3 Conclusion :

Dans ce chapitre nous avons présenté la plate forme logicielle ainsi que ces différentes fonctionnalités, en donnant d'abord les différents champs et menus qui la composent. Nous avons aussi illustré ces différentes fonctions par divers exemples de simulation.

A noter que cette plate forme comprends à ce jour 6 méthodes d'identification/égalisation aveugle multicapteurs. Néanmoins elle a été conçue pour être facilement extensible et de façon à pouvoir y ajouter d'autres algorithmes/méthodes existant dans la littérature.

Chapitre IV :

Algorithmes CR

IV-1 Algorithme de la méthode CR (Cross Relation) 36

IV-2 Algorithme de la méthode MCR (Minimum Cross Relation) 38

IV-3 Algorithme de la méthode UMCR (Unbiased MCR).....39

IV-4 Cas entrées multiples 40

IV-5 Simulations 41

IV-6 Conclusions 43

Chapitre IV :

Algorithmes CR

L'algorithme CR est l'un des algorithmes d'égalisation aveugle les plus simples et les plus efficaces pour l'identification des systèmes à réponse impulsionnelle finie de type SIMO (Single Input Multiple Output). Cet algorithme exploite la propriété de commutativité de la convolution pour constituer un système d'équations permettant l'identification de la réponse impulsionnelle du canal.

Ce chapitre est consacré à l'étude de l'algorithme CR. Il y sera présenté : d'abord la première version de l'algorithme CR développée dans **[11]** ; ensuite deux nouvelles versions de cet algorithme, des versions qui apportent plus de simplicité au calcul.

La première nouvelle version introduite, appelée Minimum Cross Relation (MCR), exploite le principe de la diversité spatiale en minimisant la redondance que présente la version initiale de l'algorithme CR. Une simulation des deux algorithmes CR et MCR sera présentée afin d'illustrer les performances de cette nouvelle version.

La seconde version, appelée Unbiased Minimum Cross Relation (UMCR), est une version modifiée de la version précédente (MCR), dans laquelle l'estimation du canal est non biaisée et ne requière pas une opération de blanchiment du bruit. De même pour cette version, une simulation la concernent sera présentée.

A la fin de ce chapitre, nous introduirons le cas MIMO (Multiple Input Multiple Output), et nous verrons quelles sont les difficultés rencontrées dans l'application de cet algorithme pour ce type de systèmes, difficultés dues à la paramétrisation non linéaire d'une base du sous-espace bruit en terme des coefficients du canal à estimer.

IV-1 Algorithme de la méthode CR (Cross Relation) :

Cet algorithme repose sur une technique très simple résumée comme suit:

En considérant l'équation (2.1) sans bruit, la sortie $y_i(k)$, $1 \le i \le L$ est donnée par :

$$y_i(k)=h_i(k)*s(k)\,, \qquad 1 \le i \le L \qquad (4.1)$$

en exploitant la commutativité de la convolution, on a :

$$h_j(k) * y_i(k) = h_i(k) * y_j(k) \qquad (4.2)$$

cette expression décrit une équation linéaire vérifiée par toutes les paires de canaux.

Théorème 4.1:

Soit le model considéré dans cette étude. La relation :

$$h'_j(k) * y_i(k) - h'_i(k) * y_j(k) = 0$$

n'est satisfaite que si et seulement si le vecteur polynomial $\boldsymbol{h}'(z)$, de taille $L\times 1$ de degré M, et le vecteur $\boldsymbol{h}(z)$ étaient linéairement dépendants: $\boldsymbol{h}'(z) = \alpha\, \boldsymbol{h}(z)$ où α est une constante scalaire.
Il a été démontré en se basant sur ce théorème que $L(L-1)/2$ couples (h_i , y_i) suffisaient pour une identification unique du canal $\boldsymbol{h}$.

En collectant toutes les paires de canaux possibles sous la forme de l'équation (4.2), on obtient un système d'équations qui peut être décrit sous une forme matricielle en trouvant une structure bâtie à partir des vecteurs observation $y_i(k)$, telle que:

$$\boldsymbol{Y}_L \boldsymbol{h} = 0 \qquad (4.3)$$

la matrice $\boldsymbol{Y}_L$ est sous la forme:

$$\boldsymbol{Y}_2 = [\, \boldsymbol{Y}_{(2)} \; -\boldsymbol{Y}_{(1)}]$$

$$\boldsymbol{Y}_l = \left[\begin{array}{cc|c} \multicolumn{2}{c|}{\boldsymbol{Y}_{l-1}} & 0 \\ \hline \boldsymbol{Y}_{(l)} & & -\boldsymbol{Y}_{(1)} \\ & \ddots & \vdots \\ 0 & \boldsymbol{Y}_{(l)} & -\boldsymbol{Y}_{(l-1)} \end{array}\right] \qquad (4.5)$$

avec l=3,..., L et

$$\boldsymbol{Y}_{(i)} = \begin{bmatrix} y_i(M) & \cdots & y_i(0) \\ \vdots & & \vdots \\ y_i(N-1) & \cdots & y_i(N-M-1) \end{bmatrix} \qquad (4.6)$$

En présence de bruit, la solution de l'équation (4.3) peut être obtenue au sens des moindres carrés :

$$\hat{\boldsymbol{h}}_{CR} = \arg\min_{\|\boldsymbol{h}\|=1} \boldsymbol{h}^H \boldsymbol{Y}_L^H \boldsymbol{Y}_L \boldsymbol{h} \qquad (4.7)$$

$\hat{\boldsymbol{h}}_{CR}$ représente le vecteur canal estimé.

Il est facile à démontrer que le vecteur $\hat{\boldsymbol{h}}_{CR}$ correspond au vecteur propre associé à la plus petite valeur propre de la matrice $\boldsymbol{Y}_L^H \boldsymbol{Y}_L$ **[22].**

IV-2 Algorithme de la méthode MCR (Minimum Cross Relation) :

Le nombre de paires de canaux pris pour l'estimation du canal dans l'algorithme CR (eq.4.2) peut être réduit à un minimum de (L−1) paires grâce au théorème suivant :

Théorème 4.2 :

Soit $\{P_1,\ldots,P_{L-1}\}$, (L-1) couples formant une structure en arbre (Fig.4.1). Alors les (L-1) relations croisées :

$$h_{i_1}(k) * y_{i_2}(k) - y_{i_1}(k) * h_{i_2}(k) = 0, \quad i = 1,\ldots,L-1 \qquad (4.8)$$

où $P_i=(i_1,i_2)$ (i_1 et i_2 représentent les indices des canaux) et $\boldsymbol{h}'_i(z)$ est un vecteur polynomial de degrés M, donne une estimation unique, à un scalaire près, du vecteur canal.

Démonstration : voir article en annexe 4

La relation (4.2) peut s'écrire sous une forme plus compacte, une forme matricielle:

$$\boldsymbol{F}(h) * y(k) = 0 \qquad (4.9)$$

Ou sous la forme:

$$\boldsymbol{G}(y) * \boldsymbol{h}(k) = 0 \qquad (4.10)$$

Tel que $\boldsymbol{F}(h)$ et $\boldsymbol{G}(y)$ sont deux matrices qui dépendent respectivement du vecteur canal et du signal d'observation ayant les structures suivantes:

$$\boldsymbol{F}(h)= \begin{bmatrix} h_2 & -h_1 & 0 & \cdots & 0 \\ h_3 & 0 & -h_1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ h_q & 0 & \cdots & 0 & -h_1 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & 0 & h_q & -h_{q-1} \end{bmatrix} \qquad (4.11)$$

$$\boldsymbol{G}(y)= \begin{bmatrix} y_2 & -y_1 & 0 & \cdots & 0 \\ y_3 & 0 & -y_1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ y_q & 0 & \cdots & 0 & -y_1 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & 0 & y_q & -y_{q-1} \end{bmatrix} \qquad (4.12)$$

Il est à noter que les structures des matrices $\boldsymbol{F}(h)$ et $\boldsymbol{G}(y)$ ne sont pas uniques, il existe bien d'autres structures qui conviennent à l'identification (i.e: une structure est dite convenable pour une identification si celle-ci permet d'identifier le canal de façon unique).

Pour illustrer ce résultat, voici l'exemple suivant :

On considère à la figure (4.1) la structure en arbre décrivant le cas L=5 et $p_1 = (1,2)$, $p_2 = (1,3)$, $p_3 = (3,4)$ et $p_4 = (3,5)$ l'ensemble de paires de canaux.
En présence de bruit, la solution de l'équation de l'équation (4.7):

$$\hat{\boldsymbol{h}}_{MCR} = \arg\min_{\|\boldsymbol{h}\|=1} \boldsymbol{h}^H \overline{\boldsymbol{Y}}_L^H \overline{\boldsymbol{Y}}_L \boldsymbol{h} \qquad (4.13)$$

où $\overline{\boldsymbol{Y}}$ est une matrice bloc construite, selon la structure (4.12), comme suit :

les blocs d'indices (i,l) sont nuls si $l \notin (i_1, i_2)$ avec $p_i = (i_1, i_2)$, et égales à $\boldsymbol{Y}_{(i_2)}$ si $l = i_1$, et égales à $-\boldsymbol{Y}_{(i_1)}$ si $l = i_2$.

Contrairement à l'algorithme CR, dans l'algorithme MCR on a besoin de blanchir le bruit car la matrice de covariance du bruit n'est pas proportionnelle à la matrice identité, ceci peut être expliqué par le fait que les paires de canaux ne soit pas toutes prises de la même manière . Certaines observations sont utilisées plus que d'autres.

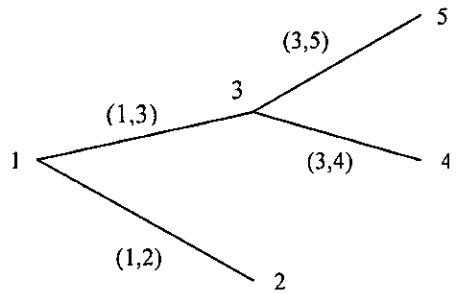


Fig. 4.1 : Exemple d'arbre pour le cas L=5.

Dans le cas de l'exemple précédent:

$$E(\overline{\boldsymbol{W}}^H \overline{\boldsymbol{W}}) \propto diag(2,1,3,1,1) \otimes \boldsymbol{I} \qquad (4.14)$$

tel que $\overline{\boldsymbol{W}}$ est la matrice représentant le bruit ayant la même forme que la matrice $\overline{\boldsymbol{Y}}$ et $\otimes$ représente le produit de Kronecker.

Donc on voit bien que la matrice de covariance du bruit n'est pas proportionnelle à la matrice identité.

Pour ce problème rencontré (concernant le bruit), l'algorithme UMCR propose une solution qui sera expliquée dans le paragraphe suivant.

IV-3 Algorithme de la méthode UMCR (Unbiased MCR) :

Cet algorithme est une version améliorée de l'algorithme MCR ne nécessite pas de blanchiment .

Précisément, au lieu de prendre (L-1) relations croisée comme dans le cas de l'algorithme MCR, c'est à dires (L-1) paires canaux, on prendra L relations correspondant aux paires suivantes:

$$\begin{cases} p_1 = (1,2) \\ p_2 = (2,3) \\ \vdots \\ p_{L-1} = (L-1, L) \\ p_L = (L,1) \end{cases} \qquad (4.15)$$

où les (L-1) premières paires correspondent à ceux de l'algorithme MCR, et la dernière représente la redondance choisie. De cette façon, toutes les sorties des capteurs sont utilisées de la même manière.

Un autre avantage de cet algorithme est le fait que ses performances ne dépendent pas du choix des paires utilisées pour la construction de l'arbre, ce qui conduit à un léger gain en performances par rapport à l'algorithme MCR, chose qui sera illustrée dans le paragraphe IV-4. Cet algorithme ne nécessite pas un blanchiment du bruit, comme le stipule le théorème suivant:

Théorème 4.3:

Selon le modèle considéré, les relations croisées de la relation (4.15) permettent une identification de la réponse du canal asymptotiquement non biaisée.

Démonstration :

En présence de bruit, la matrice $\overline{Y}$ définie en (4.13), construite à partir des paires de la relations (4.15) devient: $\overline{Y} = \overline{X} + \overline{W}$

Tel que $\overline{W}$ représente le bruit additif.

Du fait que le bruit soit blanc gaussien et indépendant du signal source, nous avons la relation suivante:

$$E(\overline{Y}^H \overline{Y}) = E(\overline{X}^H \overline{X}) + E(\overline{X}^H \overline{W}) + E(\overline{W}^H \overline{X}) + E(\overline{W}^H \overline{W})$$

Le second et le troisième terme de cette équation sont nuls, et le dernier terme s'écrit :

$$E(\overline{W}^H \overline{W}) = 2\sigma^2 (T-M)\, \boldsymbol{I}_{L(M+1)}$$

tel que σ^2 représente puissance du bruit, T la taille de l'échantillon et M le degré du canal. De ce fait, le terme bruit qui apparaît lors du calcul du vecteur propre associé à la plus petite valeur propre de la matrice $\overline{Y}^H \overline{Y}$, n'influe pas sur le résultat obtenu.

IV-4 Cas entrées multiples :

Dans le cas entrées multiples, la fonction de transfert devient une matrice de taille $L \times p$, tel que p est le nombre de signaux source et $1 < p < L$.

Les relations écrites indépendamment du bruit sont les suivantes:

$$\begin{aligned} \boldsymbol{y}_{(p)}(k) &= [\boldsymbol{H}_{(p)}(z)]\, \boldsymbol{s}(k) \\ \boldsymbol{y}_{p+1}(k) &= [\boldsymbol{H}_{p+1,:}(z)]\, \boldsymbol{s}(k) \\ &\vdots \\ \boldsymbol{y}_{L}(k) &= [\boldsymbol{H}_{L,:}(z)]\, \boldsymbol{s}(k) \end{aligned} \qquad (4.16)$$

avec $\boldsymbol{y}_{(p)}(k) = [\, y_1(k) , \ldots , y_p(k)]^T$ tel que: $\boldsymbol{H}_{(p)}(z)$ est le bloc supérieure de taille $p \times p$ de la matrice $\boldsymbol{H}(z)$, et $\boldsymbol{H}_{i,:}(z)$ représentent le $i^{ème}$ vecteur ligne de $\boldsymbol{H}(z)$ de taille $1 \times p$.
$[\boldsymbol{H}(z)]\, \boldsymbol{s}(k)$ représente la sortie d'un système de réponse $\boldsymbol{H}(z)$ et de signal source
$\boldsymbol{s}(k) \overset{def}{=} [\, \boldsymbol{s}_1(k) , \ldots , \boldsymbol{s}_p(k)]^T$
sous les conditions d'identifiabilité, la matrice $\boldsymbol{H}(z)$ est de rang p : $\text{rang}[\boldsymbol{H}_{(p)}(z)]=p, \ \forall z$

de ce fait : $\det(\boldsymbol{H}_p(z)) \neq 0$

dans ce cas : $\boldsymbol{s}(k)=[\boldsymbol{H}_{(p)}^{-1}(z)]\boldsymbol{y}_{(p)}(k)=\left[\dfrac{\text{com}(\boldsymbol{H}_{(p)}(z))}{\det(\boldsymbol{H}_{(p)}(z))}\right]\boldsymbol{y}_{(p)}(k)$ (4.17)

En remplaçant cette équation dans (4.16), on obtient :

$$\begin{aligned} [\det(\boldsymbol{H}_p(z))]\, \boldsymbol{y}_{p+1}(k) &= [\boldsymbol{H}_{p+1,:}(z)\ \text{com}(\boldsymbol{H}_{(p)}(z))]\, \boldsymbol{y}_{(p)}(k) \\ &\vdots \\ [\det(\boldsymbol{H}_p(z))]\, \boldsymbol{y}_{L}(k) &= [\boldsymbol{H}_{L,:}(z)\ \text{com}(\boldsymbol{H}_{(p)}(z))]\, \boldsymbol{y}_{(p)}(k) \end{aligned} \qquad (4.18)$$

tel que $\text{com}(\boldsymbol{A})$ et $\det(\boldsymbol{A})$ représentent respectivement la co-matrice et le déterminant de la matrice $\boldsymbol{A}$.
Les équations (4.18) représentent des relations non linéaires des éléments du vecteur $\boldsymbol{h}$, ce qui signifie que l'extension de l'algorithme CR au cas de sources multiples n'est pas évident.

IV-5 Simulations :

Dans ce paragraphe, nous présentons quelques simulations pour évaluer les performances des différentes versions de l'algorithme CR.

Soit un système type SIMO avec L=6 sorties représentées par des fonctions polynomiales de degré M=4. Les coefficients du canal sont générés aléatoirement (à chaque essaie Monte-Carlo) suivant une distribution gaussienne complexe. La taille de trame est de 256 symboles appartenant à l'alphabet QAM4. Un bruit blanc gaussien de variance σ^2 est rajouté au signal observé de telle façon que le rapport signal sur bruit $SNR=\dfrac{\|\boldsymbol{h}\|^2}{\sigma^2}$ vari dans l'intervalle $[0,30]\, dB$.

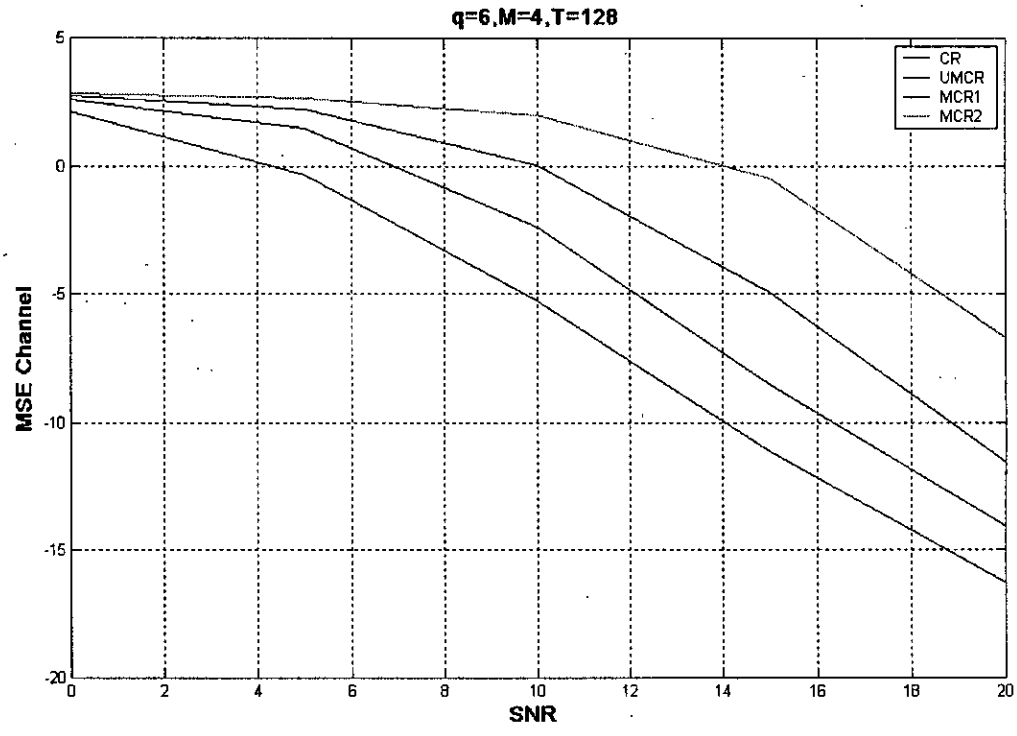


Fig.4.2 : comparaison de performances de l'algorithme CR, MCR et UMCR pour L=6.

Les calculs statistiques sont faits pour $N_r = 1000$ essaies Monte-Carlo, et les performances ont été estimées par le critère de l'erreur quadratique moyenne (MSE) normalisée :

$$MSE = \frac{1}{N_r} \sum_{i=1}^{N_r} \frac{\left\| \boldsymbol{h} - \hat{\boldsymbol{h}}^{(i)} \right\|^2}{\left\| \boldsymbol{h} \right\|^2}$$

où $\hat{\boldsymbol{h}}^{(i)}$ représente le vecteur estimation du canal au $i^{éme}$ essaie.

Les simulations illustrées à la figure 4.2 représentent les performances de :

- l'algorithme CR ;
- l'algorithme MCR pour deux structures différentes d'arbre : MCR1 pour la structure $p_i = (1, i),\ i = 2,...,L$ et MCR2 pour la structure $p_i = (i-1, i),\ i = 2,...,L$;
- l'algorithme UMCR.

Les résultats des simulations montrent bien la dépendance des performances de l'algorithme MCR de la structure de l'arbre choisie, et qu'entre deux structures différentes (comme il est le cas dans les simulations faites) il y a une perte en performances considérable .

Ces simulations montrent aussi que l'algorithme UMCR est beaucoup plus proche de l'algorithme CR en performances que ne l'est l'algorithme MCR.

IV-6 Conclusions :

Nous avons présenté dans ce chapitre la méthode d'identification aveugle multi-canaux basée sur les relations croisées (CR) existantes entre les différentes sorties du système.
Nous avons commencé par rappeler le principe de base et l'algorithme original de la méthode. Nous avons ensuite apporté certaines généralisations ainsi que plusieurs améliorations à cette méthode en vue de réduire son coût de calcul.
Plus précisément, nous avons : (i) proposé une méthode dite des relations croisées minimum (MCR), i.e., utilisant un nombre minimum de relations croisées; (ii) nous avons ensuite proposé une version de celle-ci dite non-biaisée (UMCR) du fait qu'elle ne nécessite pas de blanchiment du terme bruit contrairement à la méthode MCR ; (iii) nous avons finalement généralisé le principe de la méthode CR et discuté sa mise en œuvre dans le cas de systèmes à entrées multiples (MIMO).

Il est à noter que ce travail a fait l'objet d'une publication dans une conférence internationale IEEE (*7th International Symposium on Signal Processing and its Applications, 1-4 Juillet 2003, Paris, France*). Cet article est joint en annexe 4.

Partie 2

Introduction :

La densité croissante des circuits programmables actuels, notamment des FPGA (Field Programmable Gate Array), permet le prototypage rapide des circuits numériques à grande complexité. Ces circuits permettent aussi de tester rapidement la validité de certaines architectures complexes: l'implémentation complète d'un processeur sur des circuits FPGA est aujourd'hui réalisable, entraînant ainsi plus de possibilités d'évaluation que celles offertes par des simulateurs logiciels.

De plus, la reprogrammabilité de ces circuits a ouvert de nouvelles voies de recherche: des méthodologies de conception des systèmes reconfigurables, capables d'évoluer ou de s'adapter à des environnements ou à des contraintes variables.

Le domaine du traitement du signal, nécessite un traitement en temps réel d'une quantité très importante d'informations. De ce fait, les études algorithmiques ne peuvent se faire indépendamment des techniques de mise en œuvre ou d'applications de ces méthodes sur le support physique, car beaucoup d'applications ne peuvent être significatives qu'avec un nombre important de processeurs analogiques ou numériques, ce qui peut, dans certains cas, constituer une limite aux méthodes proposées.

L'étude conjointe des algorithmes de traitement du signal et de leur implémentation sur des architectures devient donc une étape incontournable pour la validation de ces algorithmes.

Dans cette partie du rapport , il n'est pas question d'étudier et de démontrer la capacité d'égalisation de l'algorithme CR (Cross Correlation) , mais d'établir des liens existant entre la capacité d'égalisation de cet algorithme avec les résultats d'implantation sur FPGA. En d'autres termes, nous cherchons par notre travail à vérifier si une implémentation sur circuit FPGA pourrait mettre en valeur toutes les capacités d'égalisation de l'algorithme CR.

Chapitre I :

Styles de conception de circuits intégrés

I-1 Circuits ASICs 46

I-2 Circuits PLDs 47

- PALs 47
- CPLDs.....47
- FPGAs.....47
  - ♦ Circuits FPGA Virtex-II.....48
    - Description générale de l'architecture d'un circuit FPGA Virtex-II.....48
      - CLBs / Slices 49
      - IOBs 50
      - DCM 51
      - Blocs Multiplieurs 51
      - Blocs Mémoire.....52

I-3 Conclusion 53

Chapitre I :

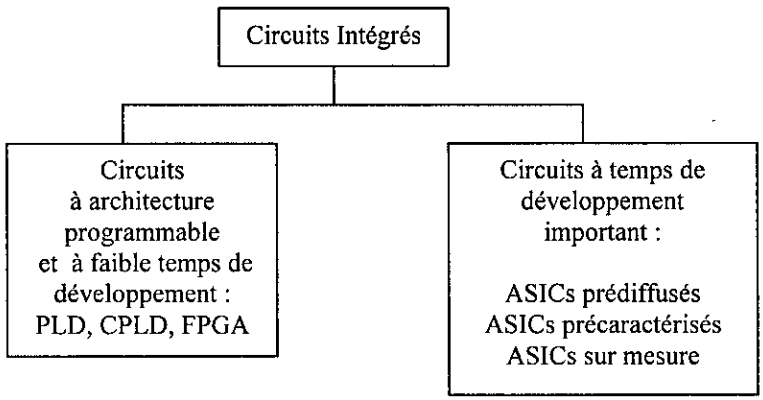
Styles de conception de circuits intégrés

Il existe deux styles de conception pour les circuits intégrés ; conception de circuits type ASICs (Application Specific Integrated Circuit), conception de circuits type logique programmable (PLDs). Selon le domaine d'application et les objectifs visés, on pourra opter pour l'un ou l'autre de ces deux styles de conception.

Les plus importants facteurs pris en considération lors d'un tel choix sont principalement : le 'time-to-market' (temps de développement) ; le coût ; la densité d'intégration ; la rapidité du circuit ; ...

Il est à noter que le choix de l'un de ces deux styles sera suivi de bien d'autres choix et décisions, qui rapprochent le plus possible le produit final des exigences requises, car ces deux grandes familles de circuits intégrés possèdent différents types de circuits, comme il sera présenté dans ce qui suit.

Le schéma suivant résume les principaux types de circuits intégrés appartenant à ces deux familles de circuits :



I-1 Circuits ASICs :[10]

Les ASICs sont des circuits dont la fonction peut être *personnalisée* en vue d'une application spécifique, par opposition aux circuits standards à usage globale dont la fonction serait définie et décrite dans des catalogues.

Les ASICs sont des circuits où il est nécessaire d'avoir une grande maîtrise des conceptions allant d'une description de haut niveau jusqu'aux dessin des masques. Ce sont donc des circuits qui nécessitent un temps de conception très important, qui n'est justifié que pour des grandes séries de fabrication ou pour des conceptions de très grande précision quant aux caractéristiques demandées.

Un ASIC est défini par sa structure de base (réseau programmable, cellule de base, matrice,...) et par les moyens matériels et logiciels qui permettent sa "personnalisation". **[9]**

Le principal avantage de ce type de circuits est sa taille considérablement réduite par rapport aux composants standards, ce qui implique une plus grande rapidité de réponse et nécessairement un coût moindre.

On distingue trois types de circuits ASICs :

- les prédiffusés (Gate Arrays) : ce sont des circuits déjà fabriqués, qui comportent un ensemble de transistors ou de portes à interconnecter, avec tout ce que cela suppose comme problèmes de routage et de délais de programmation qu'il faudra résoudre.

- les précaractérisés (Standard Cell) : ce sont des circuits dont on fabrique au préalable des bibliothèques de cellules standards à placer sur les semi-conducteurs.

- les circuits conçus sur mesure (Full Customs) : entièrement définissables par le concepteur. Ces circuits permettent la réalisation de tous les composants VLSI (Very Large Scale Integration) comme les microprocesseurs.

La conception de CI type ASIC repose sur les étapes suivantes :

- Décomposition de l'application en fonctions principales ;
- Construction de réseaux de transistors ;
- Dimensionnement des transistors ;
- Dessin des masques.

I-2 Circuits PLDs :

Les circuits logiques programmables (PLDs : Programmable Logic Dvices) sont capables de réaliser plusieurs fonctions logiques dans un seul circuit. Si ces fonctions étaient réalisées à base de circuits de logique classique, il en faudrait plusieurs.

Ces circuits sont à architecture programmable à partir d'un ordinateur, et sont aussi re-programmables sans être extraits de leur environnement. Le principe de base de cette programmation est de connecter logiquement entre eux les différents blocs de ces circuits présentant des fonctions de base.

Exploitant les technologies les plus performantes du moment, leur architecture doit sans cesse évoluer pour offrir les meilleures performances, en terme de vitesse, de capacité et d'utilisation.

On trouve sur le marché plusieurs centaines d'architectures différentes que l'on peut regrouper dans quelques familles génériques : PAL, CPLD et FPGA :

- PAL : (programmable array logic) Ce sont les circuits logiques programmables les plus anciens. Les PAL sont programmés par destruction de fusibles. Ils ne sont donc programmables qu'une seule fois.

- CPLD : c'est une intégration de plusieurs PLD en un seul circuit, reliés entre eux par une matrice centrale.

- FPGA : les FPGAs sont assimilables à des ASICs programmables par l'utilisateur. Les blocs logiques constituant ces circuits sont plus nombreux et moins complexes que ceux des CPLDs.

NB : Ce dernier type de circuits programmables (FPGA) étant le type choisi pour notre application, des détails d'architecture interne ainsi que leurs principales caractéristiques seront présentés dans ce qui suit.

Les circuits FPGAs comptent de nombreuses familles appartenant à de nombreux fabricants de circuits intégrés. Dans ce qui suit, il sera présenté l'un des FPGAs existants sur le marché actuellement, le circuit **FPGA Virtex-II de Xilinx** (voir Annexe 1). C'est le circuit qui a été utilisé pour notre application.

Circuits FPGA Virtex-II :

La famille Virtex-II est l'ensemble de ce qu'on appelle des plates-formes FPGA.

Développée pour de hautes performances, pour des conceptions basées sur des blocs d'IP (Intellectual Property) et des blocs re-programmables.

La famille Virtex-II fournit des FPGAs à de diverses densités d'intégration allant de la plus faible à la plus importante dépassant les 10 millions de portes

Ces FPGAs possèdent les fonctions les plus avancées qu'une conception pourrait demander :

des DCMs (Digiatl Clock Manager), Xtreme Multipliers , XCITE technology ((1) annexe 2);
des CLBs (Configurable Logic Bloc) de Virtex-II possèdent deux fois plus de slices (voir plus loin) ; une plus grande capacité mémoire ;...

Les FPGAs de Virtex-II se caractérisent d'une très grande souplesse d'adaptation aux applications du traitement du signal.

NB : des informations sur les principales caractéristiques des circuits Virtex-II sont présentées dans la partie annexée N° 2.

- Description de l'architecture des FPGAs Virtex-II :

Les FPGAs de Virtex-II sont des circuits intégrés constitués de cinq éléments de base qui sont : (voir fig.1) **[4]**

- Les blocs d'entrée/sortie (les IOB) ;
- Les blocs mémoire RAM ;
- **Les multiplieurs** ;
- **Les blocs logiques configurables (les CLB)** ;
- **Les blocs contrôle d'horloge (les DCM)** ;

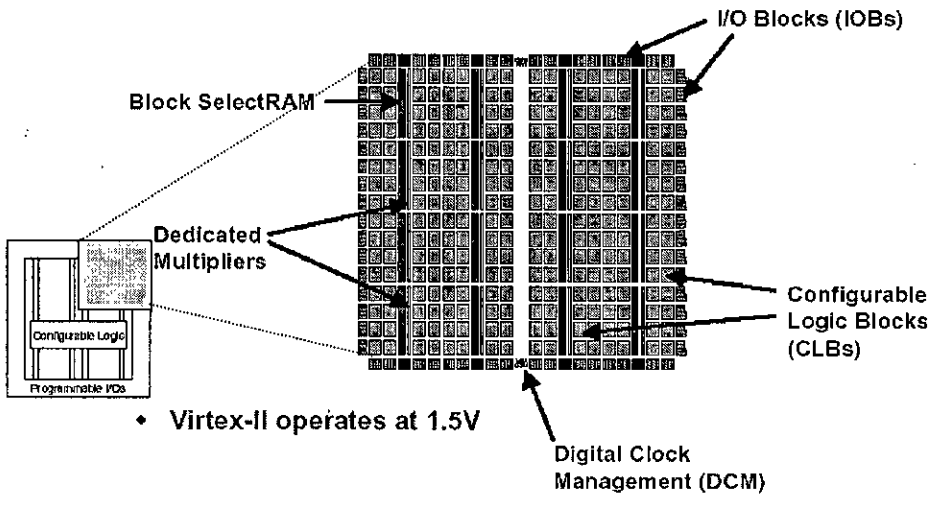


Fig. 1.1 : Architecture d'un FPGA

- **Les CLBs :**

Chaque CLB contient 4 slices (voir fig. 1.3), soit deux fois plus de ce que contenaient les précédents FPGAs, caractérisé par : **[4]**

- de rapides connections;
  internes (entre ses différents slices) ;
  externes(le reliant aux CLB voisins) ;
- un bloc d'échange(Switch Matrix) lui permettant l'accès aux ressources de routage ;
- des buffers accessibles par les seize sorties du CLB ;(voir fig.1.4)
- deux propagations indépendantes de la retenue ;

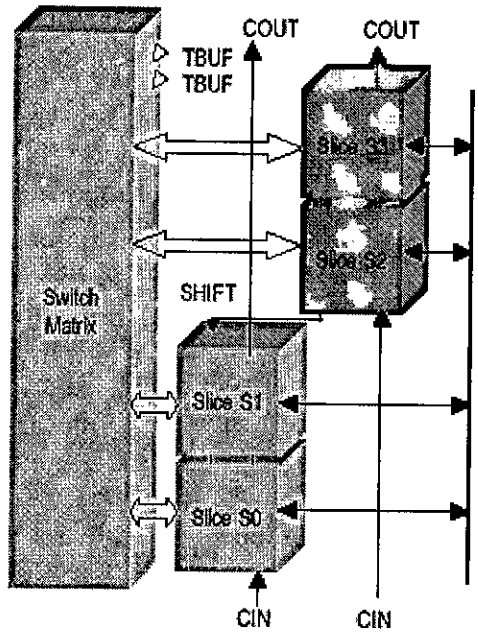


Fig.1.2 : Architecture simplifiée d'un CLB.

- Les slices :

Chaque slice constituant le CLB est composé par : **[4]**

- deux bascules D de stabilisation ;
- quatre sorties ; deux sorties stabilisées (par des bascules D).
  deux sorties non stabilisées.
- propagation verticale de la retenue (carry) ;
- deux blocs de logique combinatoire(LUT : Look-Up Table);

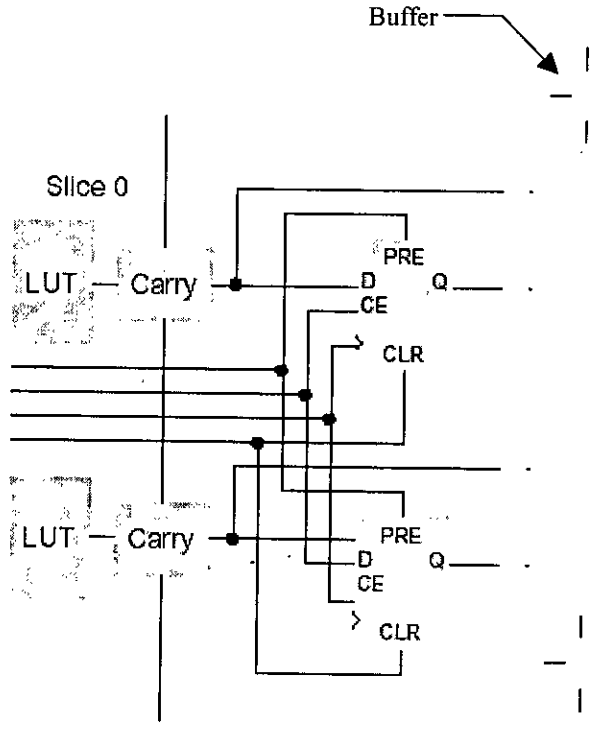


Fig. 1.3: Architecture simplifiée d'un slice.

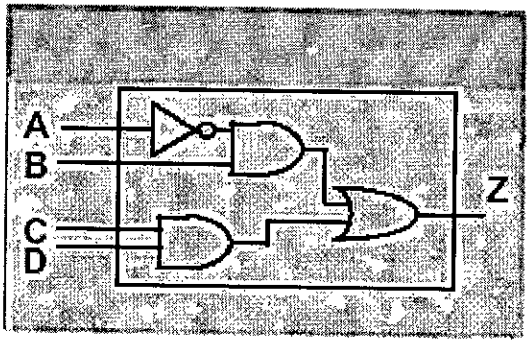


Fig. 1.4 : Logique combinatoire.

Les fonctions élémentaires de ces circuits sont incluses dans des blocs appelés des Look-Up Tables (LUT) à 4 entrées.
Chaque opérations effectuée par le slices est donc un ensemble des fonctions booléennes de ces blocs. De ce fait, tout délais de propagation est indépendant des opérations effectuées.

- Les Blocs E/S (IOB) : [2]

Les blocs E/S des circuits Virtex-II sont groupés par deux ou par quatre dans la périphérie du circuit. Chaque IOB peut être utilisé comme entrée et/ou comme sortie, et chaque paire d'IOB peut être utilisée un groupe différentiel (voir Fig.1.5).

Chaque bloc d'E/S (IOB) contient un ensemble d'éléments de stockage. (voir Fig.1.6)

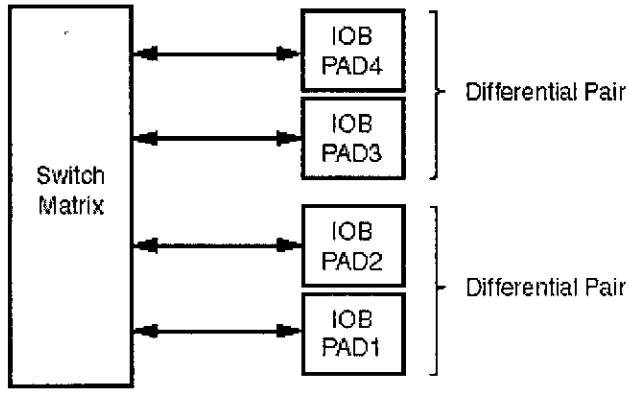


Fig.1.5 : Elément E/S (IOB).

Chaque élément mémoire peut être configuré soit en bascule D sensible à un niveau de basculement, soit en latch sensible au front montant ou descendant. Pour les entrées , les sorties et les branches trois états , un ou deux registres DDR peuvent être utilisés. **[2]**

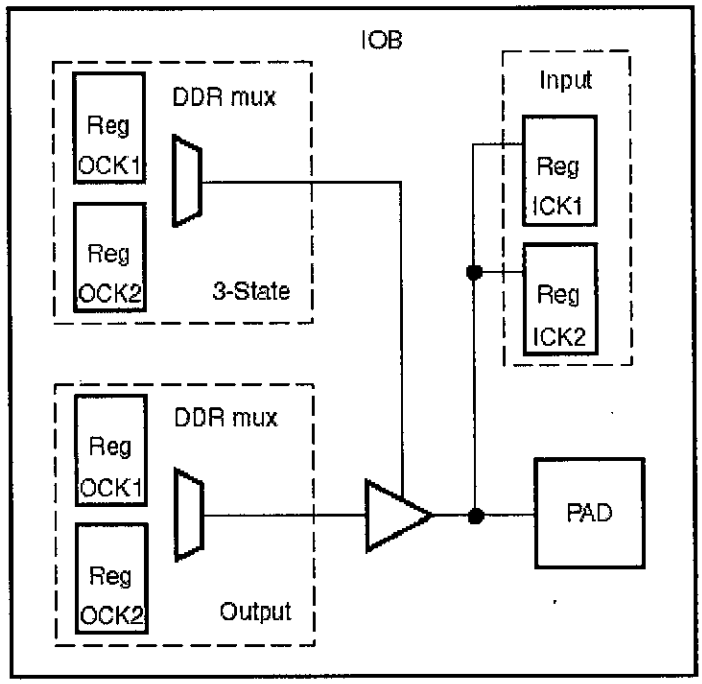


Fig.1.6 : Bloc d'E/S

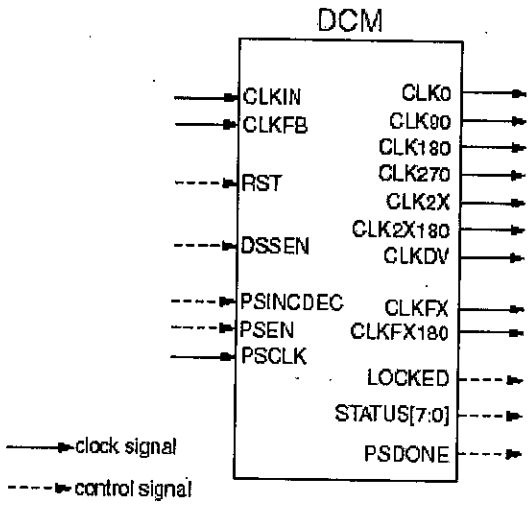


Fig.1.7 : Digital Clock Manager.

- **Le Bloc DCM :**

Le bloc DCM offre plusieurs options pour une bonne manipulation de l'horloge : **[2]**

- Clock De-skew : la DCM crée de nouveaux signaux Clock (internes ou externes au FPGA) en phase avec le signal Clock principal.
- Frequency Synthesis : la DCM offre la possibilité de créer une large gamme de fréquences de signaux d'horloge en multipliant ou en divisant la fréquence du signal horloge principal.
- Phase Shifting : cette option permet de déphaser le signal horloge, par un contrôle dynamique, soit d'un déphasage grossier soit d'un déphasage fin et très précis.

- **Bloc Multiplieur :**

Les blocs multiplieurs des circuits Virtex-II sont des multiplieurs 18 x 18 bits à complément à 2. Ces blocs multiplieurs peuvent être associés à des blocs SelectRAM de 18 Kbit , comme ils peuvent travailler indépendamment.

L'utilisation des multiplieurs associés aux blocs SelectRAM , avec accumulateur , dans les blocs LUTs permet l'implémentation de MAC (multiplier-accumulator) des DSP (Digital Clock Manager) qui sont généralement utilisés dans l'implémentation des filtres à réponse impultionnelle finie ou infinie.

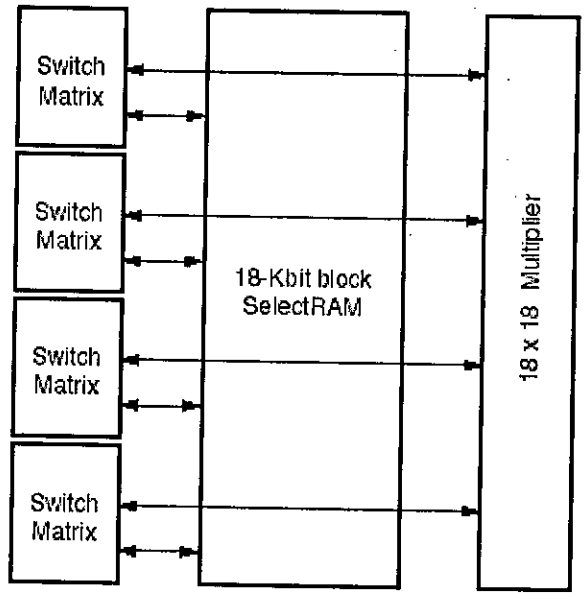


Fig.1.8 : Bloc SelectRAM / Bloc Multiplieur.

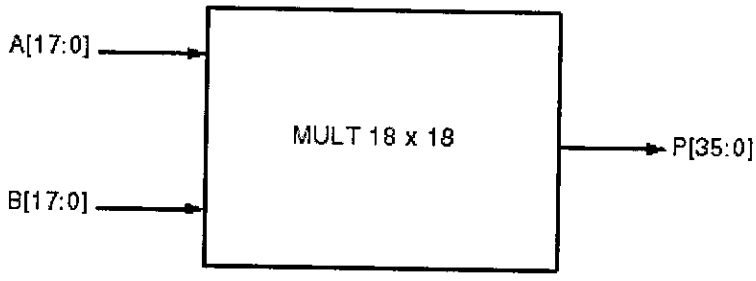


Fig.1.9 : Bloc Multiplieur.

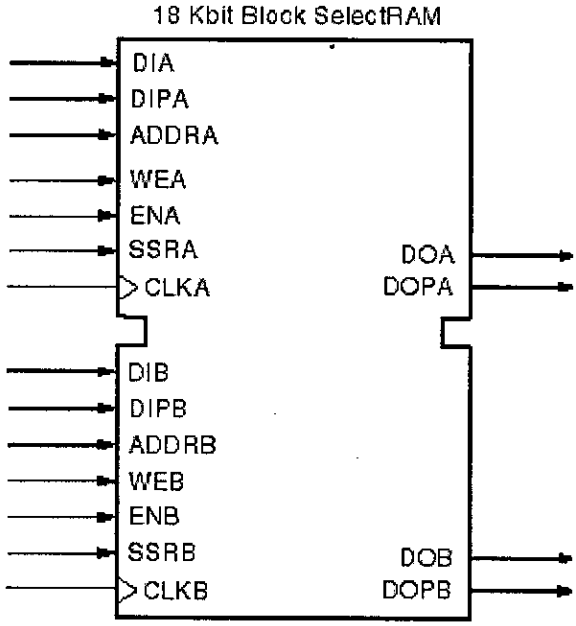


Fig.1.10 : Bloc SelectRAM en mode Dual-port.

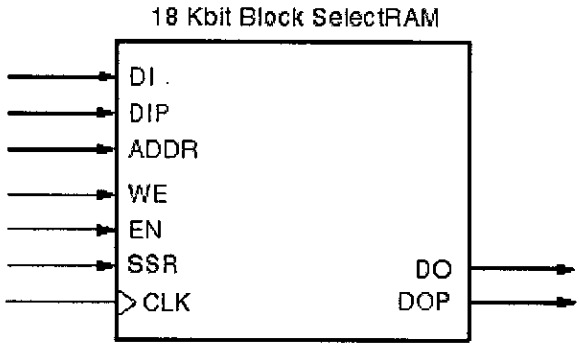


Fig.1.11 : Bloc SelectRAM en mode Single-port.

- **Bloc SelectRAM :**

Les circuits Virtex-II sont munis de blocs mémoire appelés SelectRAM. Ces blocs , de taille 18 Kbits chacun, viennent compléter la mémoire de bas niveau que possède chaque CLB.
Pouvant être utilisés en deux mode (Fig.1.10, Fig.1.11) , la mémoire de ces blocs peut être exploitée selon différentes configuration possibles (Tab.1.1) :
En mode Single-port la mémoire peut être de 18 Kbits ou de 16 Kbits (Tab.1.1) ;
En mode Dual-port la mémoire est exploitée entièrement.
Les deux ports constituant le bloc SelectRAM possèdent des signaux d'horloge et des signaux de contrôle indépendants, mais l'exploitation de ces signaux est identique.
Les deux ports possédant donc les signaux suivants : Clock et Clock Enable, Write Enable, Set/Reset, et Address, mais aussi DATA/PARITY pour les données en entrée et DATA/PARITY pour les données en sortie.

16 K x 1 bit	2 K x 9 bits
8 K x 2 bits	1 K x 18 bits
4 K x 4 bits	512 x 36 bits

Tab.1.1 : Les différentes configuration du blocs SelectRAM.

I-3 Conclusion :

Dans ce chapitre, nous avons d'abord cité les deux styles de conceptions les plus importants, également les différents types de circuits qu'ils englobent, dans le but de :
situer et introduire la conception de circuits intégrés programmables;
étudier leur architecture afin de comprendre la manière dont cette « programmation » pouvait s'effectuer au niveau physique.

Nous avons ensuite détaillé quelque peu l'architecture interne des circuits FPGA afin de relever un point important qui les caractérise, il s'agit de la rapidité et la fiabilité de conception que ces circuits, qui sont dues en partie au fait qu'ils contiennent des blocs dédiés à des fonctions bien précises (exp : les blocs DCM pour la gestion des signaux d'horloge) permettant d'éviter l'utilisation de circuits combinatoires pouvant alourdir l'architecture du système et introduire des retards indésirables.
Cette facilité est aussi due au fait que la conception de ce type de circuits soit assistée par des outils de développement de plus en plus efficaces. La conception devient alors de plus en plus souple ce qui est dangereux pour un concepteur qui n'adopte aucune méthode pour sa conception. C'est pourquoi le prochain chapitre est consacré à la description du cycle de conception d'un circuit programmable en citant les principes fondamentaux qu'il faut prendre en considération.

Chapitre II :

Cycle de conception des FPGAs

II-1 Langage de description VHDL 55

II-2 Approche de conception 56

II-3 Outil de conception ISE Foundation 57

II-4 Conclusions 59

Chapitre II :

Cycle de conception des FPGAs

II-1 Langage de description VHDL :

VHDL = VHSIC(Very High Speed Integrated Circuit) Hardware Description Language.
Ce langage est un langage de spécification, de simulation et de conception.
Il a été écrit dans les années 70 pour réaliser la simulation de circuits électroniques. On l'a ensuite étendu en lui rajoutant des extensions pour permettre la conception (synthèse) de circuits logiques programmables.
Standardisé en 1987, le VHDL est ensuite révisé et normalisé en 1993 et en 1997 pour améliorer sa portabilité et le rendre ainsi lisible quel que soit le fabricant de circuits.

Le langage VHDL présente un certain nombre d'avantages :
Il permet une organisation hiérarchique avec la possibilité de vérification et de simulation du comportement de chacun des composants individuellement ce qui simplifie la recherche des erreurs ; n'est pas spécifique à une technologie particulière tout en offrant la possibilité de caractériser l'une d'elle en particulier ; permet une description et une synthèse rapide ; permet l'écriture de bibliothèques de composants réutilisables d'un projet à l'autre ; permet une description sans se soucier de l'architecture interne du circuit ; ...

Pour avoir une vue globale de ce qu'est une description en VHDL, voici la structure la plus basique d'une telle description.

- Structure d'une description en VHDL :

Une description en VHDL est globalement constituée :

- d'une déclaration des bibliothèques(1).
- d'une déclaration d'une entité(2) et des Entrées/Sorties (I/O).
  - Le NOM_DU_SIGNAL.
  - Le SENS du signal.
  - Le TYPE
  - Affectation des numéros de broches en utilisant des attributs supplémentaires.
- d'une déclaration de l'architecture(3) correspondante à l'entité : description du fonctionnement.

(1) : Elles contiennent les définitions des types de signaux électroniques, des fonctions et sous programmes permettant de réaliser des opérations arithmétiques et logiques.

(2) : L'entité donne les informations concernant les signaux d'entrées et de sorties de la structure ainsi que leurs noms et leurs types.

(3) : L'architecture décrit le comportement de l'entité. Il est possible de créer plusieurs architectures pour une même entité. Chacune de ces architectures décrira l'entité de façon différente.

II-2 Approche de conception :

Avant de commencer une quelconque conception, il est utile de se demander ; de quelle manière celle-ci sera faite, quelles sont les différentes étapes qui la constituent, dans quel ordre va-t-on les exécuter, quelle type de descriptions faut il adopter pour telle ou telle partie du circuit, ...

Toutes ces questions nous amènent à adopter une certaine méthode de travail pour minimiser les erreurs de développement ou de faciliter leur détection.

L'organigramme suivant permet de résumer cette méthode de travail :

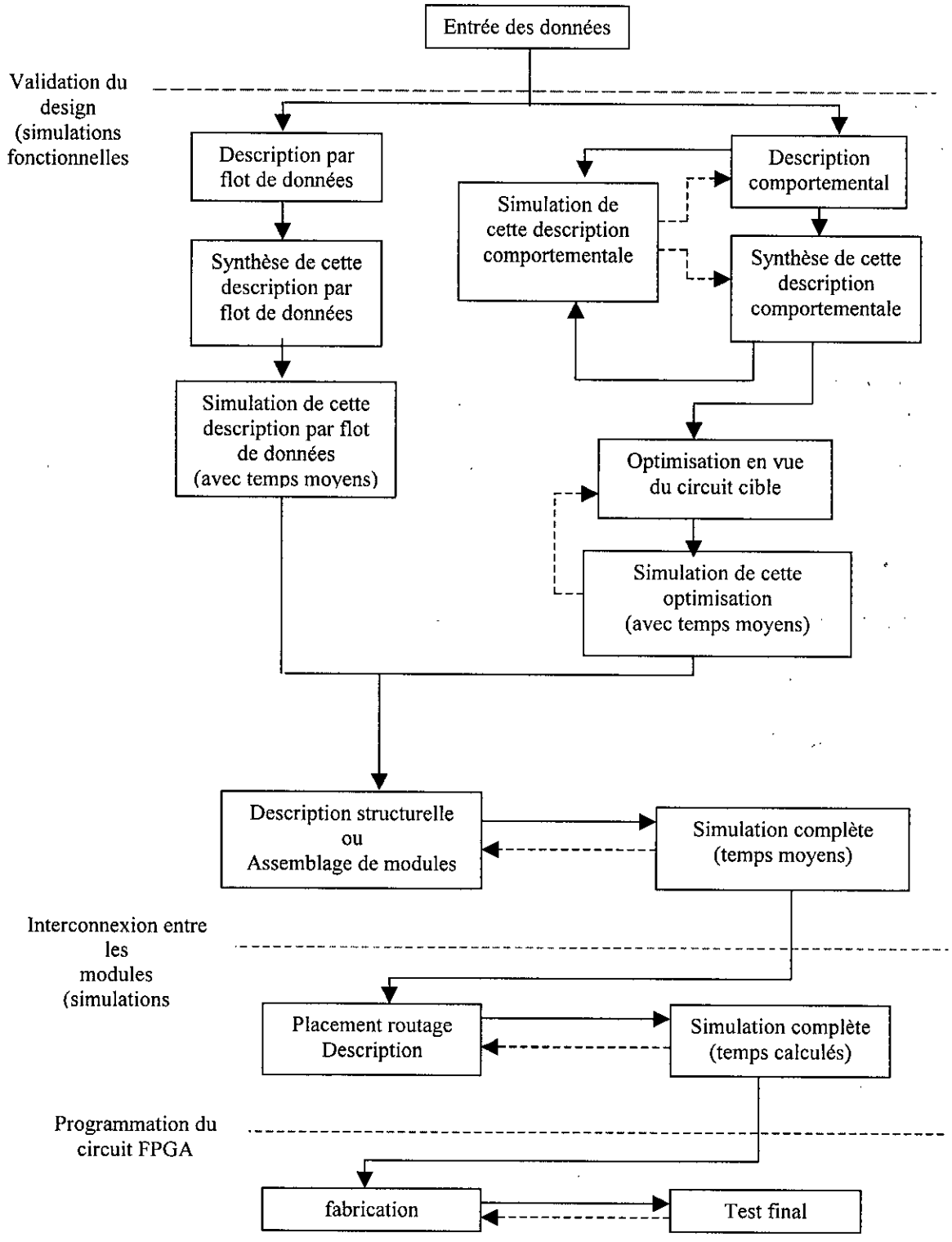


Fig.2.1 : Organigramme présentant une méthode de conception.

Le développement se décompose donc de la manière suivante :

- La première étape consiste à départager les différentes fonctions qui constituent le système globale afin de choisir le type de description à adopter pour chacune d'elles.
  Certaines des fonctions seront décrites en description dite par flots de données, laquelle est consacrée aux bus et registres dont on défini les entrées, les sorties et les entrées/sorties.
  Les autres seront décrites en description comportementale, dans laquelle les instructions s'exécutent de manière séquentielle.

- Un assemblage de tous les modules est alors réalisé par une description structurelle.

- L'étape suivant est le placement-routage.

- La dernière étape est la fabrication du circuit.

Il vient s'ajouter à chaque étape, des simulations pour vérifier chacune des descriptions mentionnées précédemment.

Il existe deux sortes de simulations :

- Simulation fonctionnelle : permet de vérifier la validité du circuit par rapport au cahier de charge, et ne tient pas compte des capacités de liaison dues au routage entre les différentes cellules, c'est à dire ne tient pas compte des retards.
- Simulation temporelle : consiste à vérifier la fonctionnalité du circuit en prenant en compte les longueurs d'interconnexions et les retards apportés par les capacités liées au routage.

La méthode qui vient d'être présentée est assistée automatiquement par l'outil de développement ISE FONDATION (Integrated Software Engineering), la synthèse, la simulation et l'implémentation deviennent donc automatiques.

- ***La synthèse*** **: [6],[7]**

C'est un processus de translation qui permet de translater une description comportementale en une description structurelle ou physique. C'est l'équivalent d'une compilation d'un programme écrit en langage évolué.

II-3 Outil de conception ISE Fondation :

Le cycle de conception ISE Fondation est un package complet qui intègre toutes les étapes de conception d'un circuit FPGA de manière automatique, ce dernier contient plusieurs modules qui fonctionnent selon un processus itératif. (fig. 2.2).**[8]**

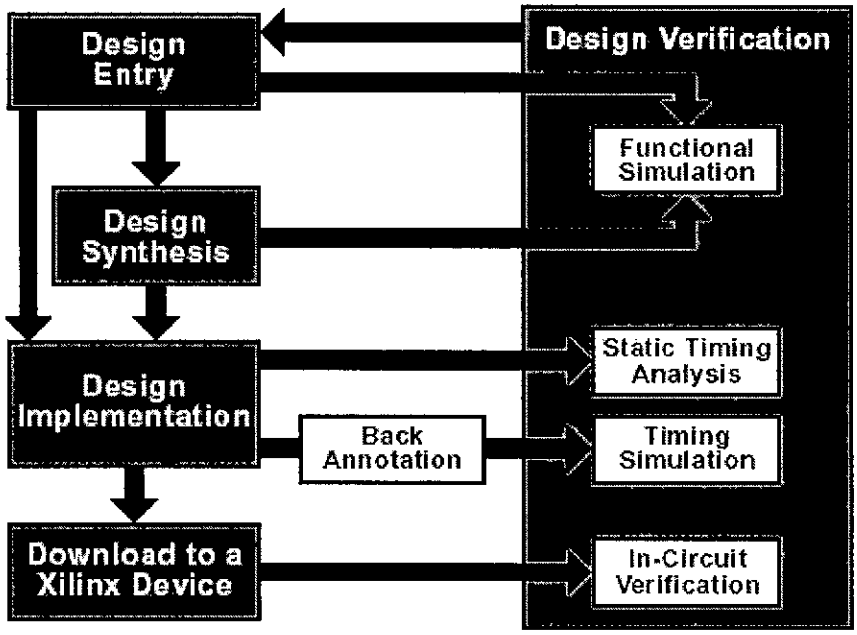
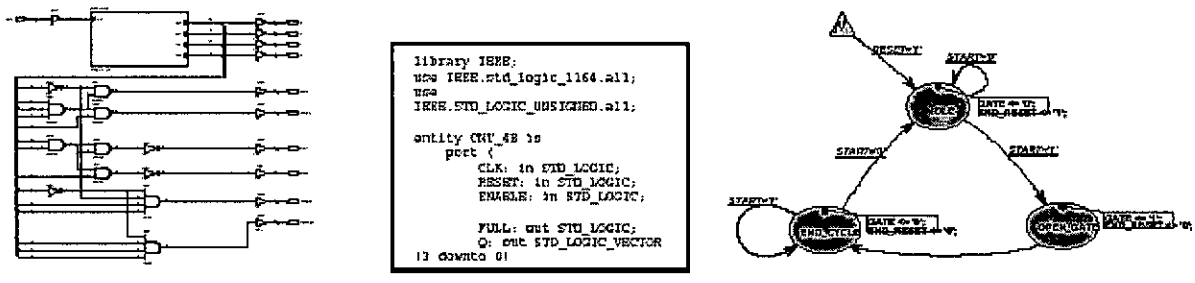


Fig.2.2 : Méthode de conception avec l'outil de conception ISE Fondation.

Les différents qui constituent le cycle de développement de ISE Fondation retrace les différentes étapes de la méthode de travail mentionnée précédemment.

- *Module d'entrée "design entry"* :

  Ce module permet la description d'une architecture en : code VHDL ; éditeur de schéma ou en en diagramme d'état. (Fig.2.3)



Entrée schématique. Entrée en code VHDL. Diagramme d'état.

Fig.2.3 : Les différents models d'entrées.

- *Module de synthèse "design synthesis"* :

Ce module transforme la description VHDL en portes logiques (description proche des ressources matérielles) et optimise (les signaux inutilisés sont retirés, les expressions booléennes sont simplifiées et les signaux équivalent sont détectés).

- *Module d'implémentation "design implementation"* :

Il réalise le mapping, c'est la projection des équations sur les différents blocs de circuits.
Ainsi il réalise le placement-routage qui consiste à attribuer les blocs du circuit à chaque équation délivrée par la projection et à définir les connexions.
L'algorithme de placement dispose physiquement les différentes cellules et les chemins d'interconnexions dessinées entre ces cellules afin de faciliter le routage.**[7]**

- *Module de vérification "design verification"* :

Ce module permet d'analyser le temps d'exécution (static timing analysis), et de réaliser la simulation.

- *Module implémentation sur FPGA "download to Xilinx device"* :

dans ce module on réalise la programmation du circuit FPGA.
Ce module est lié au module design verification pour une vérification finale du circuit.

II-4 Conclusion :

Dans ce chapitre, nous avons d'abord présenté un des langages de conception, le langage VHDL, qui à été utilisé dans notre application, nous avons aussi exposé une méthodologie de travail afin d'optimiser la conception, éviter au maximum les erreurs et faciliter leur détection. Ensuite nous avons décrit le cycle adopté par l'outil de conception ISE Foundation qui a été utilisé dans l'application exposée dans le prochain chapitre.

Chapitre III :

Implémentation de l'algorithme CR sur circuit FPGA de Xilinx famille VirtexII

III-1 Présentation de l'algorithme CR 61

III-2 Architecture proposée 62

III-2-1 Dimensionnement des données 62

III-2-2 Choix des fonctions 64

III-2-3 Simulations 74

III-2-4 Synthèse 80

III-2-5 Implémentation 80

III-3 Conclusions 81

Chapitre III :

Implémentation de l'algorithme CR sur circuit FPGA de Xilinx famille VirtexII :

Lors de l'implémentation, il est important de prendre en considération les caractéristiques internes du circuit utilisé. De ce fait, des aller-retours entre l'algorithme et l'architecture qui devait être implémentée étaient nécessaires non seulement pour réaliser l'implémentation en adéquation avec les caractéristiques du circuit et optimiser l'architecture du système globale, mais surtout pour que la version de l'algorithme modifié garde les même performances que la version initiale.

Dans ce qui suit, nous allons présenter l'algorithme CR tel qu'il a était présenté dans la première partie de ce rapport dans le but de relever les opérateurs mathématiques principales de cet algorithme, ensuite nous allons introduire au fur et à mesure les modifications qui lui ont été apportées ainsi que leur impact sur ses performances et son architecture .

Il faudrait aussi que l'architecture réponde à un certain nombre de conditionspour qu'elle soit performante:
Elle doit avoir une grande : régularité(2), localité(3), efficiency(7), scalabilité(5) ;
une faible latency(1), I/O bandwidth(4) ;
et doit posséder la propriété ψ(6). (voir définitions 1, 2, 3, 4, 5, 6, 7 Annexe 2 Partie B)

III-1 Présentation de l'algorithme CR :

Mathématiquement, l'algorithme CR consiste à calculer un vecteur $\boldsymbol{h}_{CR}$ tel que :

$$\boldsymbol{h}_{CR} = \text{argmin}\, \boldsymbol{h}^H \boldsymbol{R}^H \boldsymbol{R} \boldsymbol{h} \qquad (3.1)$$

Une des solutions pour cette équation, serait d'imposer une contrainte au vecteur $\boldsymbol{h}$ tel que : $\|\boldsymbol{h}\|=1$,ce qui revient à calculer le vecteur propre associé à la plus petite valeur propre de la matrice carrée $\boldsymbol{G}=\boldsymbol{R}^H \cdot \boldsymbol{R}$ de rang $q \cdot (M+1)$ tel que :

M : mémoire du canal.
q : nombre de capteurs.

Une telle solution est difficile à implémenter à cause du calcul de la valeur propre, car les algorithmes qui existent pour un tel calcul impliquent des architectures très compliquées.

Cette difficulté nous a obligé à explorer une autre solution à l'équation (3.1) en imposant une nouvelle contrainte au vecteur h qui est la suivante : $\boldsymbol{e}^T.\boldsymbol{h}=1$ ce qui revient à résoudre le système d'équation : $\boldsymbol{G} \cdot \boldsymbol{h}=\boldsymbol{e}$

avec : $\boldsymbol{e}=[1\,0\cdots 0]^T$

De ce fait, l'algorithme se résume essentiellement en une multiplication matricielle et une résolution d'un système d'équations, ce qui résulte en une architecture relativement simple que nous décrivons dans la section suivante.

Afin d'illustrer les pertes en performances dues au changement apporté à l'algorithme, voici les courbes suivantes représentant la variation de l'erreur quadratique moyenne en fonction du rapport signal sur bruit pris dans l'intervalle [0,40]dB , pour une trame de taille T=128, un canal de mémoire M=4 et un nombre de capteurs q=2.

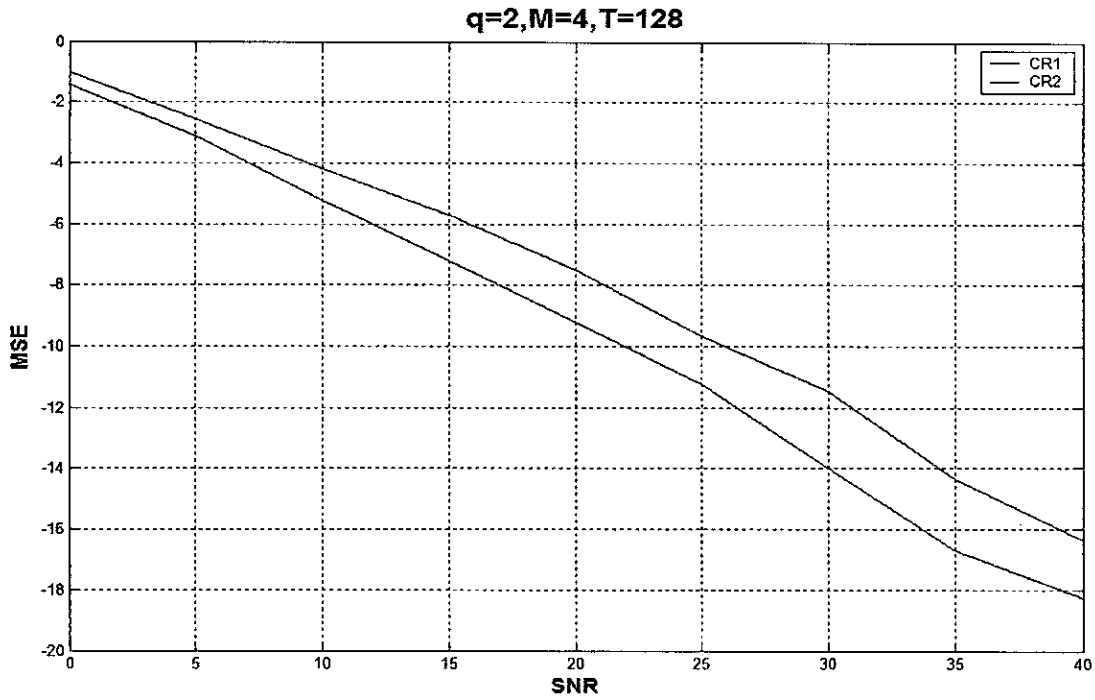


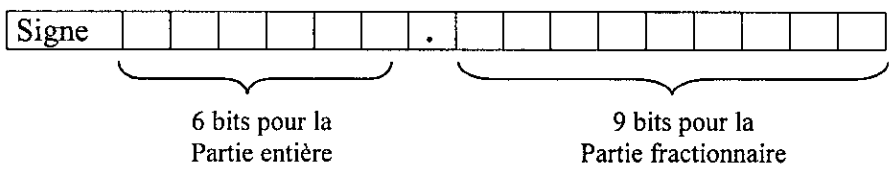
Fig. 3.1 : Courbe de comparaison :
Algorithme CR initial (CR1)/ Algorithme CR modifié (CR2).

Selon ces courbes, on constate que l'algorithme modifié engendre des pertes en performances comparables à celles de la première version, ce qui tolère son utilisation.

III-2 Architecture :

III-2-1 Dimensionnement des données :

La représentation des données en entrée est sur 16 bits organisée comme suit :



Ce dimensionnement a été choisi après un certain nombre de simulations sous Matlab, de telle sorte qu'il n'y ait pas d'overflow sur les résultats, et que ces derniers restent comparables aux valeurs que doivent prendre les coefficients du canal.

La dimension choisie pour les données d'entrée est de 16 bits (les données varient donc dans l'intervalle [-64 ; +63] avec une précision choisie de $2^{-9} \approx 0{,}001953125$) et la dimension des données en sortie est automatiquement le double (c'est à dire 32bits).

Afin d'illustrer l'impact que peut avoir le choix de la dimension, nous avons pris comme exemple deux quantifications à nombres de bits différents et nous avons mesuré l'erreur quadratique moyenne variant entre 0 et 40 dB.

Résultats de la simulation :

- Première quantification :

Quantification à 16 bits : 1 bit signe; 6 bits pour la partie entière ; 9 bits pour la précision (partie fractionnelle).

- Deuxième quantification :

Quantification à 14 bits : 1 bit signe; 6 bits pour la partie entière ; 7 bits pour la précision (partie fractionnelle).

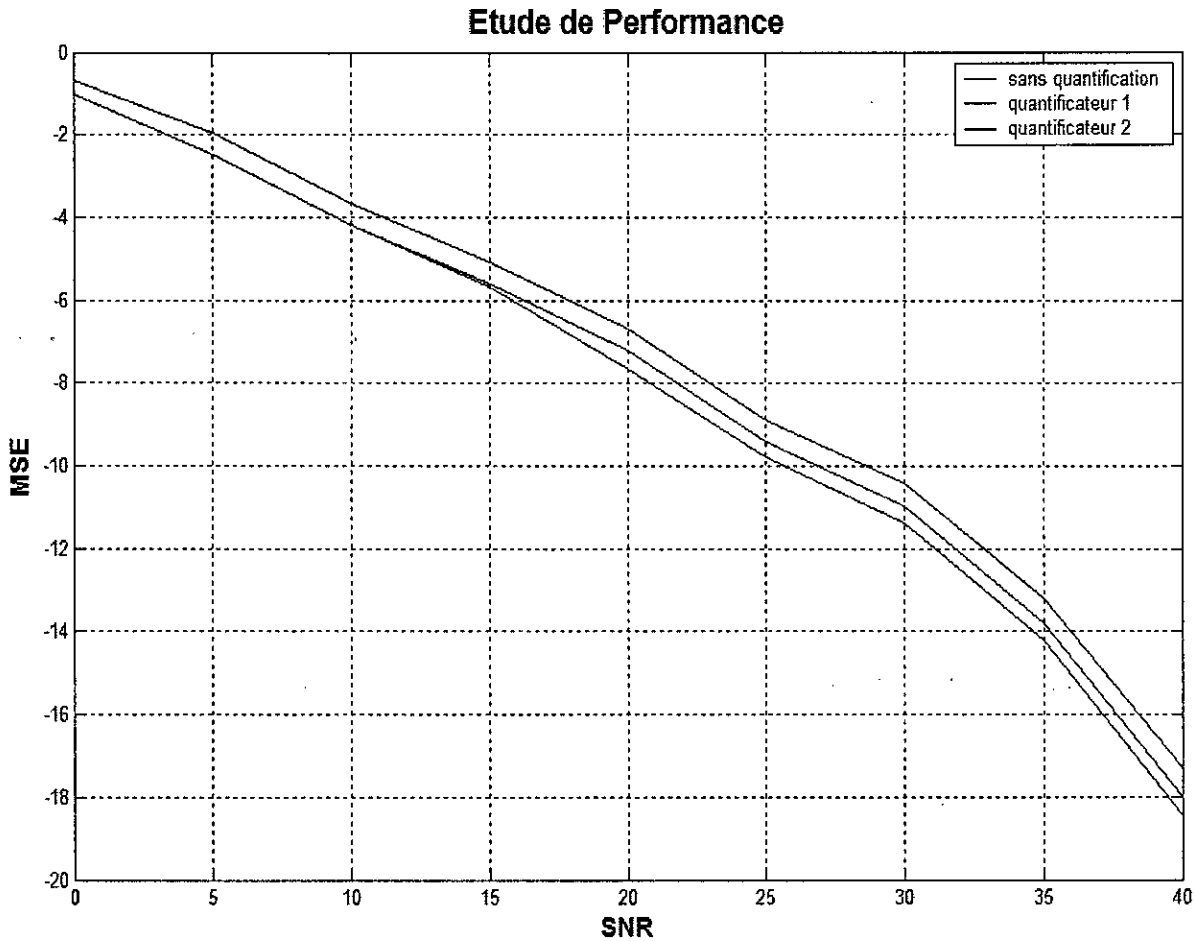


Fig. 3.2 : Comparaison des deux quantifications.

D'après ces courbes de comparaison, on voit bien que la première quantification donne des résultats qui rapprochent le plus les données quantifiées des données réelles.

III-2-2 Choix des fonctions :

La fonction globale de l'algorithme CR peut être divisée en trois blocs principaux, et chacun d'eux se compose de sous fonctions(voir arborescence Fig. 3.3).

Le premier bloc, appelé 'Loader', représente le bloc chargement des données. Ces données seront ensuite envoyées vers le deuxième bloc, appelé 'cal_G', pour la construction de la matrice $\boldsymbol{R}$ par le sous bloc Form_Y et le calcul de la matrice $\boldsymbol{G=R^H \cdot R}$ par le sous bloc Multi_trans. La matrice $\boldsymbol{G}$ sera injectée dans le dernier bloc, appelé resol_system, pour la résolution du système d'équations $\boldsymbol{G \cdot h = e}$. Ainsi la sortie du système global sera le canal de transmission estimé.

Ces blocs sont gérés par des signaux de commande et leur fonctionnement est synchronisé par un signal horloge global. Le détail de chacun des trois blocs cités ainsi que leurs signaux de contrôle est expliqué et illustré dans ce qui suit.

Arborescence du programme principal :

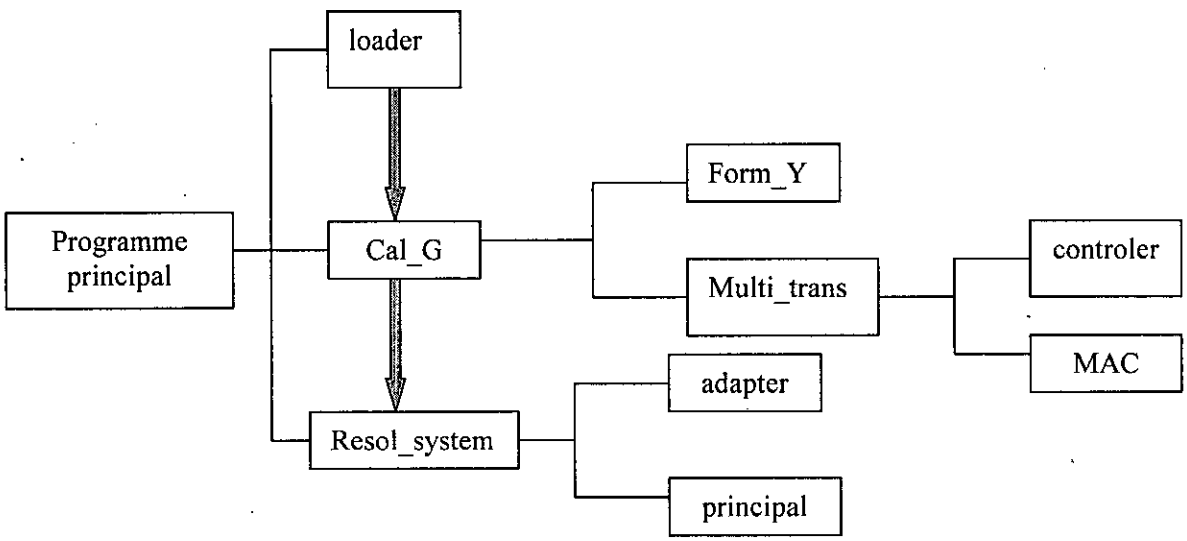


Fig. 3.3 : Hiérarchie de l'architecture globale.

Le code VHDL correspondant au bloc principal est donné dans la partie annexée n°3.

- Premier bloc:

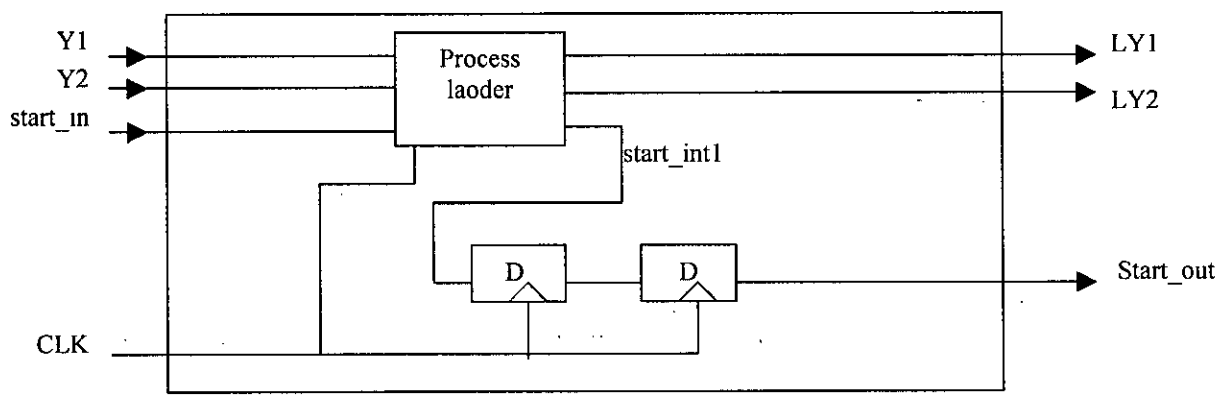


Fig. 3.4: Bloc loader.

Constitué de deux bascules D et d'un bloc Process (Annexe 3-3), ce bloc prépare les données aux prochains blocs en les stockant le temps que toute la trame soit prête.

Le signal start_out est le signal de contrôle du bloc suivant (le bloc cal_G), lui signalant qu'une trame est prête en mémoire et qu'il peut commencer sa fonction.

Latency:

La latency de ce bloc est de O(T), tel que la valeur de T (taille de la trame) est définie dans le package « My_Type.vhd » (Annexe 3- 1) (def. Package Annexe 2 Partie B).

- Deuxième bloc : (Annexe 3-4)

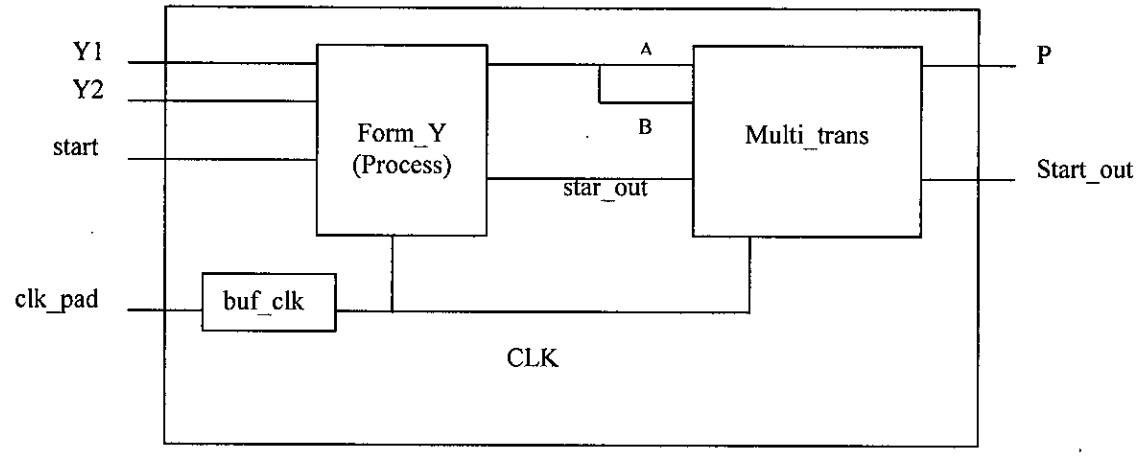


Fig. 3.5 : Bloc cal_G.

Ce bloc est formé de trois blocs principaux :

- bloc buf_clk : beuferiser le signal clk ;
- bloc form_Y : construction de la matrice **R** (Annexe 3-5), sous la forme donnee par l'expression (4.5) de la partie 1;
- bloc multi_trans : calcul de la matrice $\boldsymbol{G=R^H \cdot R}$ (Annexe 3-6).

L'acquisition des données dans ce bloc est faite de telle sorte que les entrées A et B du bloc MAC (Annexe 3-7) soient respectivement la matrice **R** et sa transposée.**[11]**

Le bloc controler (Annexe 3-8) envoi un signal au bloc MAC pour que celui-ci entame la multiplication.

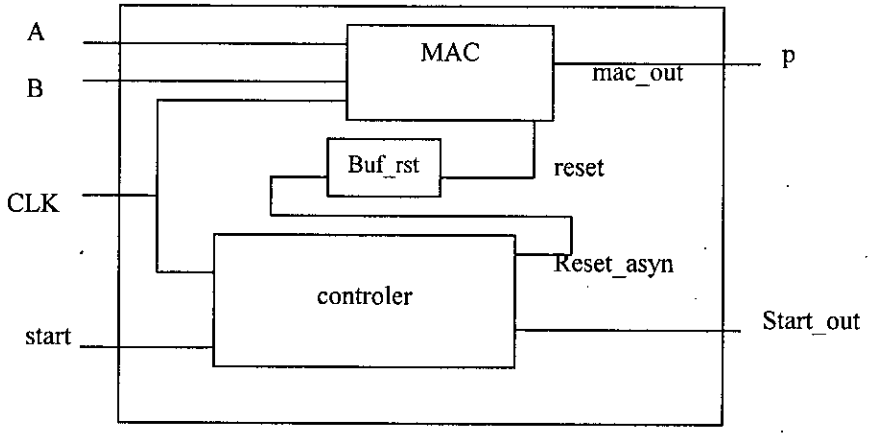


Fig. 3.6 : Bloc multi_trans.

Latency :
La latency du bloc cal_G est de O(T).

- Troisième bloc : (Annexe 3-9)

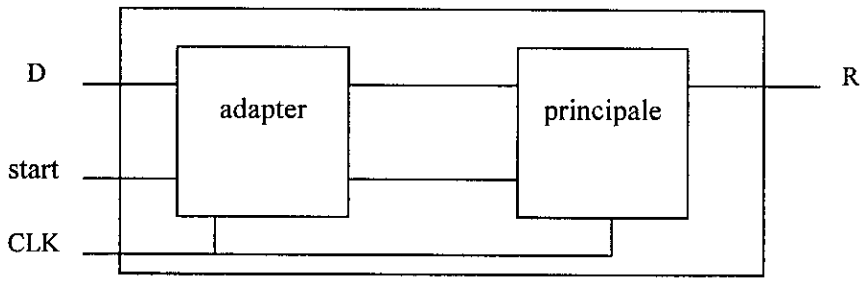


Fig. 3.7 : Bloc resol_system.

Ce bloc est constitué de deux blocs principaux ; bloc adapter et bloc principale.

- Bloc adapter (Annexe 3-10): ce bloc sert à disposer les données à la manière illustrée fig.3.8 Annexe1. Cette disposition est propre à l'architecture de l'algorithme d'élimination de Gauss.

- Bloc principale(Annexe 3-11) : ce bloc effectue la résolution du système d'équations, et donne à sa sortie le vecteur du canal estimé.

Latency :
La latency du bloc Resol_syst est de $N^2+3/2N$, tel que $N=2(M+1)$ et M étant la mémoire du canal.

Latency globale :
En sommant toutes les latency trouvées précédemment, la latency globale est de $2T+N^2+3/2N$.

NB : Pour plus de compréhension, des simulations ont été établies pour chacun des blocs cités précédemment.

- Architecture du bloc Resol_syst :

Architecture de l'algorithme de l'élimination de Gauss **[13]** :

Pour la résolution du système d'équations : $\mathbf{Q} \cdot \mathbf{V}_{pmin} = \mathbf{e}$, nous avons choisi l'architecture qui se base sur l'algorithme de l'élimination de Gauss. Celui-ci se présente comme suit :

Soit Q une matrice non singulière de taille NxN, et $\mathbf{V}_{pmin}$ l'unique solution du système. L'élimination de Gauss consiste à transformer la matrice **Q** en une matrice triangulaire supérieure par une série d'opérations élémentaires sur les lignes de cette matrice.
Ces opérations se résument comme : multiplier une des lignes par un scalaire; échanger deux lignes entre elles ; additionner le multiple d'une ligne avec une autre ligne. A chaque étape, le

résultat d'une opération peut être exprimé comme la multiplication **Q.R**, telle que **R** est une matrice représentant ces opérations.

Si la matrice **R** est choisie de telle sorte que **Q.R=I** (**I** matrice identité), nous aurons la résolution de notre système, et ceci se fera de la manière suivante:

$$\mathbf{Q} \cdot \mathbf{V}_{pmin} = \mathbf{e} \Rightarrow \mathbf{R} \cdot \mathbf{Q} \cdot \mathbf{V}_{pmin} = \mathbf{R} \cdot \mathbf{e} \Rightarrow \mathbf{V}_{pmin} = \mathbf{R} \cdot \mathbf{e}$$ et le système sera résolu.

Les différentes opérations effectuées durant le procédé d'élimination de Gauss peuvent être facilement illustrées en adoptant la configuration de la disposition de la matrice **Q** et du vecteur e suivante : **A=[Q | e]** , de telle sorte que les différents changements que subissent ces derniers y soient représentés. De cette manière, la même opération qui sera appliquée à **Q** et à **e** sera représentée comme une seule opération sur une ligne de A, et à la fin de toutes ces opérations nous obtiendrons le résultat suivant : **R . A = [I | e']**, et la solution de notre système sera le vecteur e' . En d'autres termes, le résultat du système d'équations **Q. $\mathbf{V}_{pmin}$ = e** sera la dernière colonne de la matrice **(R . A)**.

Soit l'exemple illustratif suivant :

Considérons le système d'équations suivant : **Q .V = e.**
Tel que :

$$Q = \begin{pmatrix} 2 & 4 & -7 \\ 3 & 6 & -10 \\ -1 & 3 & -4 \end{pmatrix} \qquad e = \begin{pmatrix} 3 \\ 4 \\ 6 \end{pmatrix} \qquad A = \begin{pmatrix} 2 & 4 & -7 & 3 \\ 3 & 6 & -10 & 4 \\ -1 & 3 & -4 & 6 \end{pmatrix}$$

$$\text{Etape 1} : A = \begin{pmatrix} 1 & 2 & -7/2 & 3/2 \\ 3 & 6 & -10 & 4 \\ -1 & 3 & -4 & 6 \end{pmatrix} = A^{(1)} \qquad \text{Etape 2} : A = \begin{pmatrix} 1 & 2 & -7/2 & 3/2 \\ 0 & 0 & 1/2 & -1/2 \\ -1 & 3 & -4 & 6 \end{pmatrix} = A^{(2)}$$

$$\text{Etape 3} : A = \begin{pmatrix} 1 & 2 & -7/2 & 3/2 \\ 0 & 0 & 1/2 & -1/2 \\ 0 & 5 & -15/2 & 15/2 \end{pmatrix} = A^{(3)} \qquad \text{Etape 4} : A = \begin{pmatrix} 1 & 2 & -7/2 & 3/2 \\ 0 & 0 & 1/2 & -1/2 \\ 0 & 5 & 1/2 & -1/2 \end{pmatrix} = A^{(4)}$$

$$\text{Etape 5} : A = \begin{pmatrix} 1 & 2 & -1/2 & 3/2 \\ 0 & 1 & -3/2 & 3/2 \\ 0 & 0 & 1/2 & -1/2 \end{pmatrix} = A^{(5)} \qquad \text{Etape 6} : A = \begin{pmatrix} 1 & 0 & -1/2 & -3/2 \\ 0 & 1 & -3/2 & 3/2 \\ 0 & 0 & 1/2 & -1/2 \end{pmatrix} = A^{(6)}$$

Etape 7 : $A = \begin{pmatrix} 1 & 0 & -1/2 & -3/2 \\ 0 & 1 & -3/2 & 3/2 \\ 0 & 0 & 1 & -1 \end{pmatrix} = A^{(7)}$

Etape 8 : $A = \begin{pmatrix} 1 & 0 & 0 & -2 \\ 0 & 1 & -3/2 & 3/2 \\ 0 & 0 & 1 & -1 \end{pmatrix} = A^{(8)}$

Etape 9 : $A = \begin{pmatrix} 1 & 0 & 0 & -2 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -1 \end{pmatrix} = A^{(9)}$

Et le résultat de la résolution du systeme d'equations sera donc:

$$V = \begin{pmatrix} -2 \\ 0 \\ -1 \end{pmatrix}$$

La résolution par la méthode d'élimination de Gauss pour un système de taille N x (N+1) requière N étapes.

Le déroulement du procédé s'effectue comme suit :

Première étape : identifier le premier élément de la première colonne(on l'appellera premier pivot) ; diviser tous les éléments de la ligne qui lui correspond de manière à le rendre égal à 1.

Seconde étape : multiplier les éléments de la ligne qui correspondent au premier pivot par un scalaire de sorte qu'en soustrayant le résultat aux éléments de la deuxième ligne on annule l'élément (2,1).

Troisième étape : suivre la même procédure qu'en seconde étape afin d'annuler l'élément (3,1).

Quatrième étape : rechercher le deuxième pivot qui correspond à l'élément (2,2) si celui-ci n'est pas nul, dans le cas contraire échanger la deuxième ligne avec une ligne d'ordre (2+i, i>2) qui donne un pivot non nul.(dans le cas de notre exemple le deuxième pivot est nul, et la deuxième ligne est échangée avec la troisième ligne).

Cinquième étape : répéter le même procédé qu'en étape 1 pour rendre le deuxième pivot égal à un.

Sixième étape : refaire l'étape 2 pour annuler l'élément (1,2).

Septième étape : refaire l'étape 3 pour annuler l'élément (3,2).

Huitième étape : retrouver le troisième pivot qui correspond à l'élément (3,3) si celui-ci n'était pas nul.

Neuvième étape : annuler l'élément (2,3) en refaisant l'étape 2.

Et la solution est la dernière colonne de la matrice A.

A cet algorithme correspond une architecture constituée de plusieurs blocs, et qui satisfait à la condition de régularité. De ce fait, nous nous contenterons d'expliquer seulement deux de ses blocs.

D'abord énumérons les différentes opérations effectuées durant cet algorithme :

- Inversion du premier élément (considéré comme étant le pivot) d'une colonne.
- Multiplier par ce pivot les éléments de la ligne correspondant à ce pivot et les soustraire aux éléments des lignes suivantes afin d'annuler certains élément.
- Tester si l'élément qui sera pris comme pivot est nul ou pas.

Ces fonctions seront introduites dans deux blocs différents, qui seront les blocs élémentaires de l'architecture.

Les éléments de la matrice A seront disposés à l'entrée de chaque bloc comme suit :

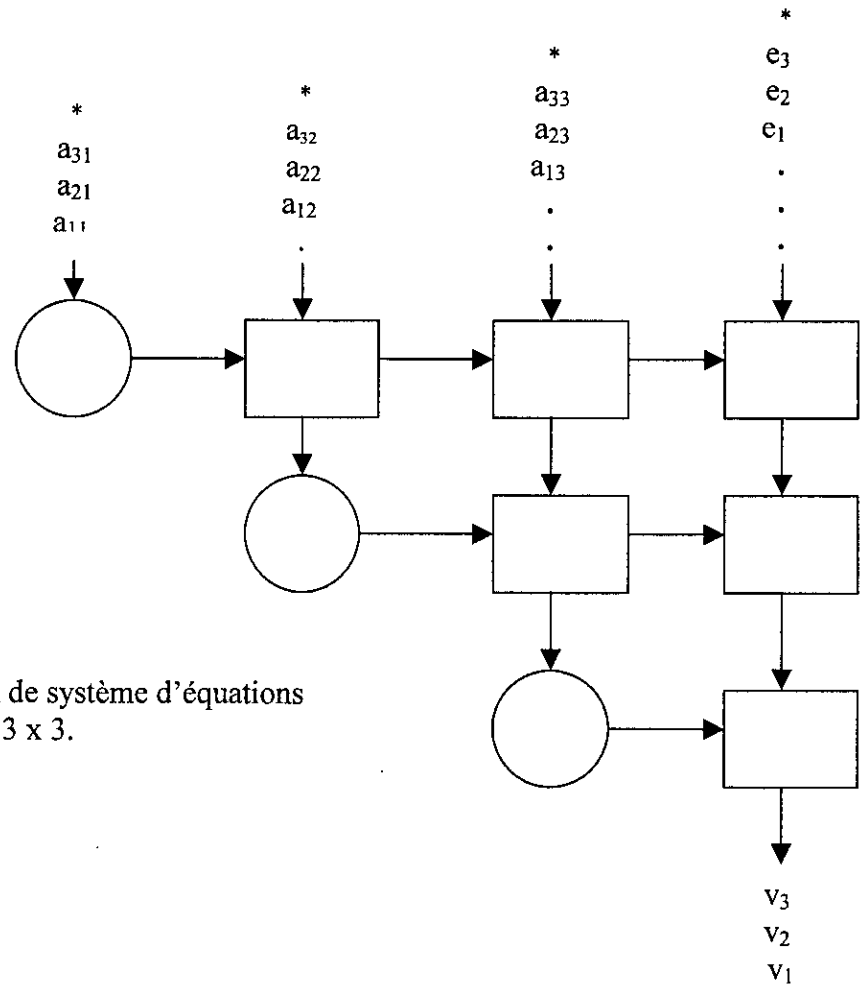
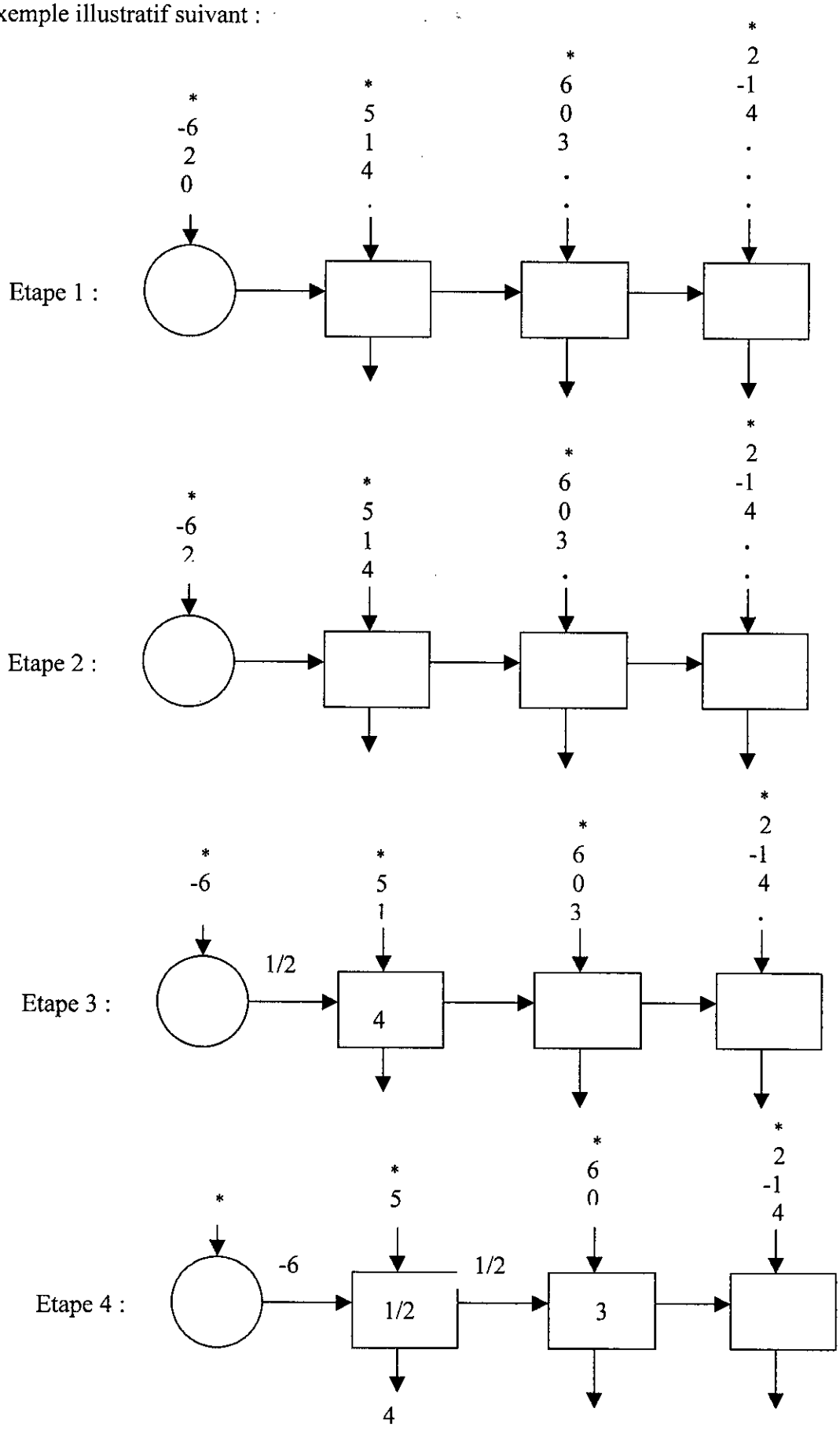


Fig. 3.8 : Cas de résolution de système d'équations de taille 3 x 3.

Soit l'exemple illustratif suivant :



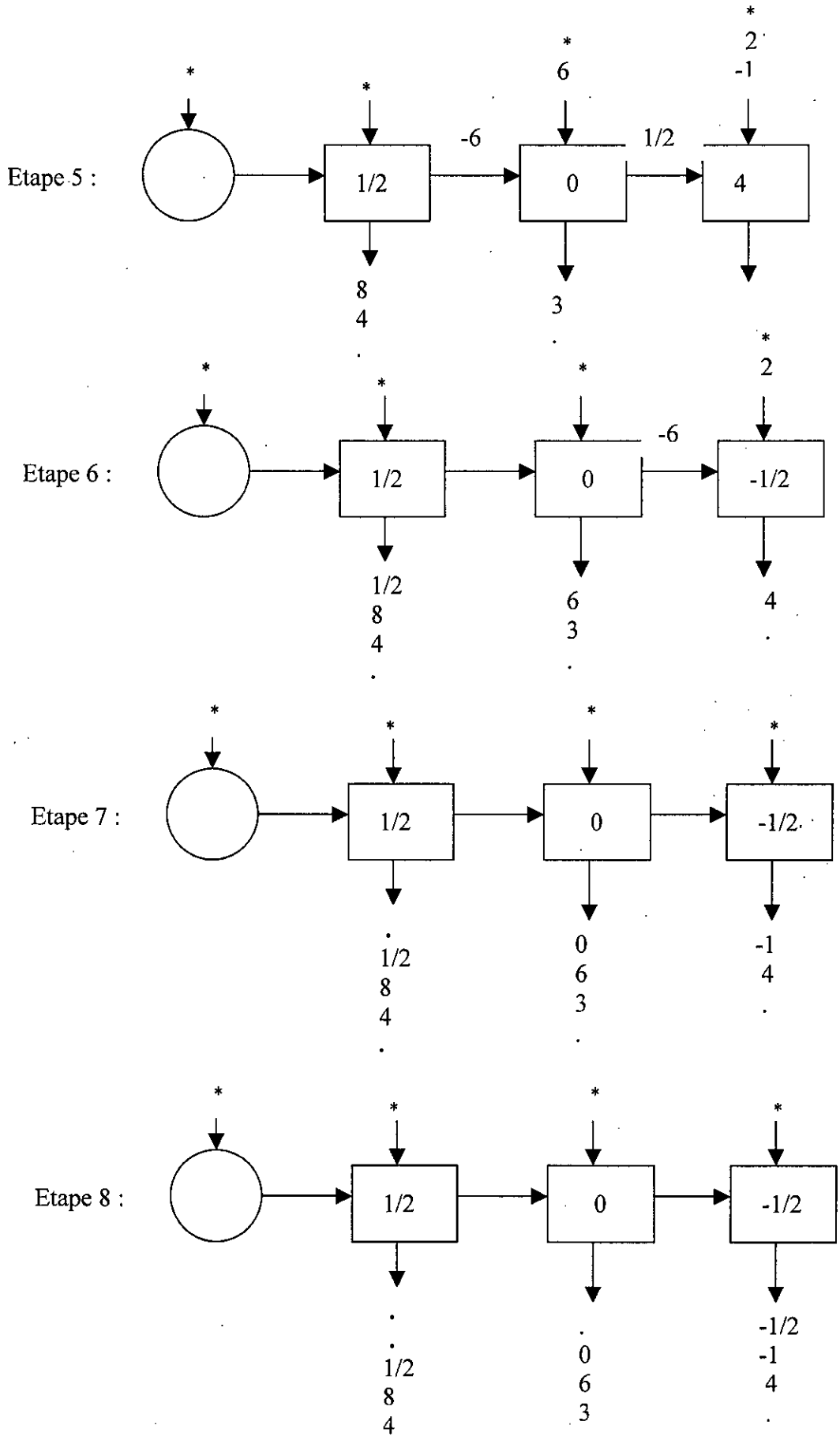


Fig. 3.9: Etapes de calcul.

En résumé, à la k^{ieme} étape de l'algorithme on aura :

1- prendre $A^{(k-1)}$ comme entrée en commençant avec k,k+1,...,N et en finissant avec 1,2,...,N-1 ;
2- calculer et stocker la k^{ieme} ligne de $A^{(k)}$ en calculant $1/ a^{(k-1)}$ dans le bloc (k,k) et $a_{kj}^{(k)}=a_{tj}^{(k-1)} / a_{tk}^{(k-1)}$ dans le bloc (k,j) pour j > k tel que t est la plus petite valeur de (k,k+1,...,N) avec $a_{tk}^{(k-1)}\neq 0$;

3- calculer et faire sortir les éléments de $A^{(k)}$ en commençant par les lignes k+1,...,N et finissant par les lignes d'ordre 1,..,k en calculant : $a_{ij}^{(k)}= a_{ij}^{(k-1)}- a_{ik}^{(k-1)} a_{kj}^{(k)}$ pour $i\neq k$ dans le bloc j pour j > k.

La solution du système sera donnée par le bloc se trouvant tout à fait à droite dans la dernière ligne de la maille (l'architecture) dans l'ordre suivant : $x_1, x_2, x_3, ..., x_N$.
Le temps pour la totalité de cette application est de N x (N-1).

1. Présentation du bloc circulaire : (Annexe 3-13)

Le bloc circulaire englobe certaines des fonctions du système citées précédemment, qui consistent à :

- repérer le premier élément non nul d'une colonne ;
- inverser cet élément et le faire passer au bloc suivant;
- faire passer le reste des éléments sans effectuer aucune opération sur ces derniers ;

Du fait que ce bloc effectue des tests sur les éléments qu'on lui présente à son entrée, celui-ci produit un signal de commande qui sera interprété par un bloc, appelé bloc de contrôle, qui à son tour génère trois signaux de commande qui seront transmis au reste des blocs se trouvant dans la même ligne.

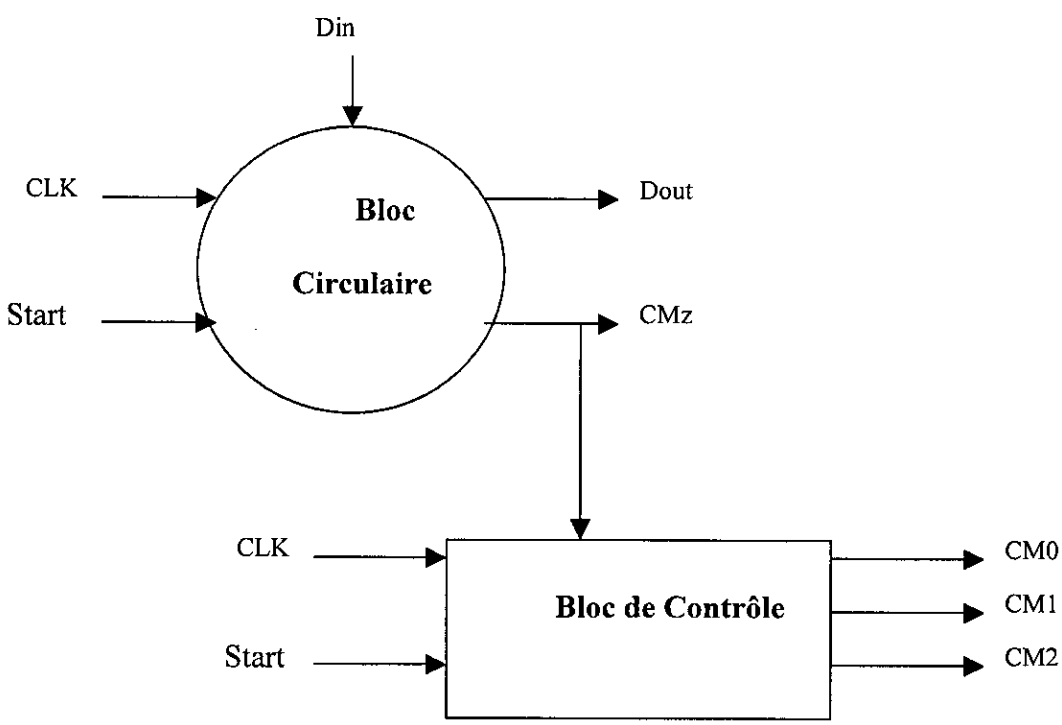


Fig. 3.10: Bloc circulaire et bloc de contrôle.

Remarque :

Les différentes opérations effectuées par le bloc circulaire nécessitent un nombre de cycles élevé (de l'ordre de 20 cycles) par rapport aux nombre de cycles nécessaires aux autres opérations effectuées par les autres blocs (blocs carrés fig. 3.11). Pour cela, l'utilisation de la DCM est nécessaire (Annexe 3-14), afin de gérer des signaux horloge à des fréquences différentes qui correspondent aux exigences de chaque bloc.

2. Présentation du bloc carré : (Annexe 3-12)

Le bloc carré englobe les fonctions suivantes :

- Multiplier par un pivot les éléments de la ligne lui correspondant à ce pivot et les soustraire aux éléments des lignes suivantes afin d'annuler les élément qu'il faut.
- Faire passer le premier élément présent à son entrée, dans le cas où le pivot est nul.

Ce bloc reçoit des signaux de commande lui provenant d'un bloc appelé bloc de contrôle lui indiquant l'ordre dans lequel il doit effectuer ses opérations.

Schéma bloc :

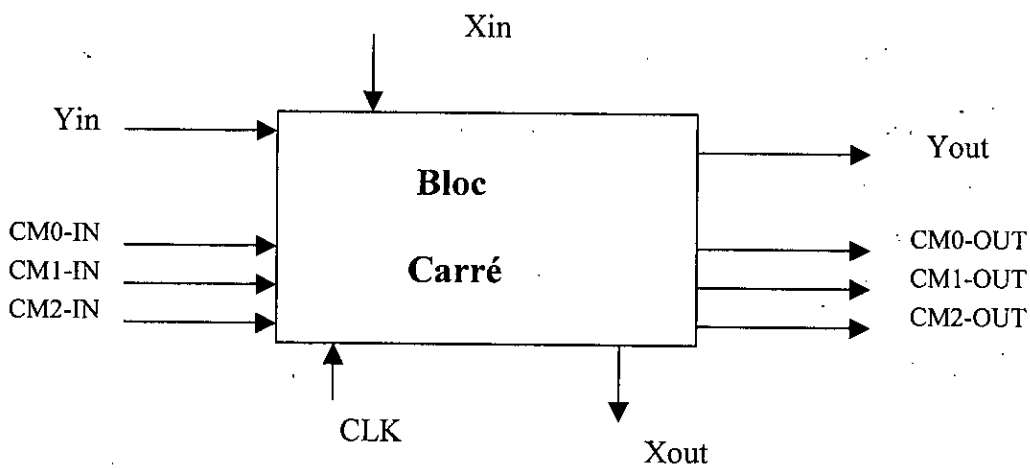


Fig. 3.11 : Bloc carré.

Dans le cas où le pivot est non nul, les opérations effectuées sont les suivantes :

- Multiplication des premiers éléments Xin * Yin = Z et stockage du résultat dans un registre.
- Opération Xin – (Yin * Z) = Xout sur le reste des éléments.
- Sortir de la donnée Z qui était stockée.

III-2-3 Simulations :

La simulation est une étape nécessaire dans le processus de conception. Elle permet de détecter les erreurs et de vérifier le bon fonctionnement de l'architecture.

Dans ce qui suit, nous allons présenter la simulation fonctionnelle des différents blocs de façon hiérarchique suivant l'arborescence fig.3.3, autrement dit, nous allons simuler d'abord les blocs élémentaires de chacun des blocs principaux, ensuite simuler les trois blocs principaux, et enfin simuler l'architecture globale, avec les paramètres suivants :

Taille de la trame : 128 ;
Memoire du canal : 4 ;
Nombre de capteurs : 2.

La simulation a été effectuée par l'outil de simulation ModelSim.

- Simulation A :

Cette simulation a pour but de tester le bloc cal_G, qui se fera de façon hiérarchique.

- Simulation A-1 : bloc form_Y

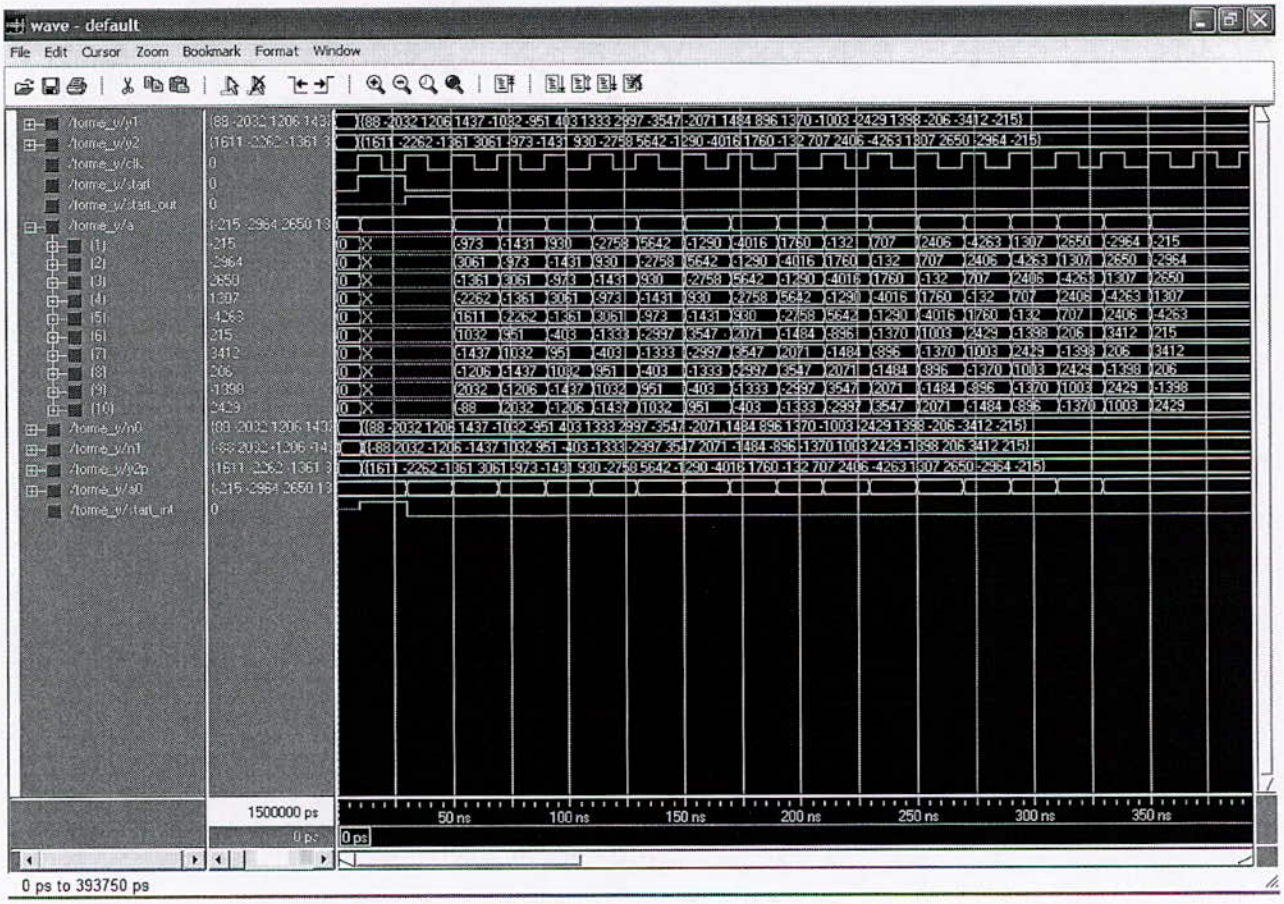


Fig. 3.12 : Simulation fonctionnelle du bloc Form_Y.

- Simulation A-2 : bloc Multi_trans

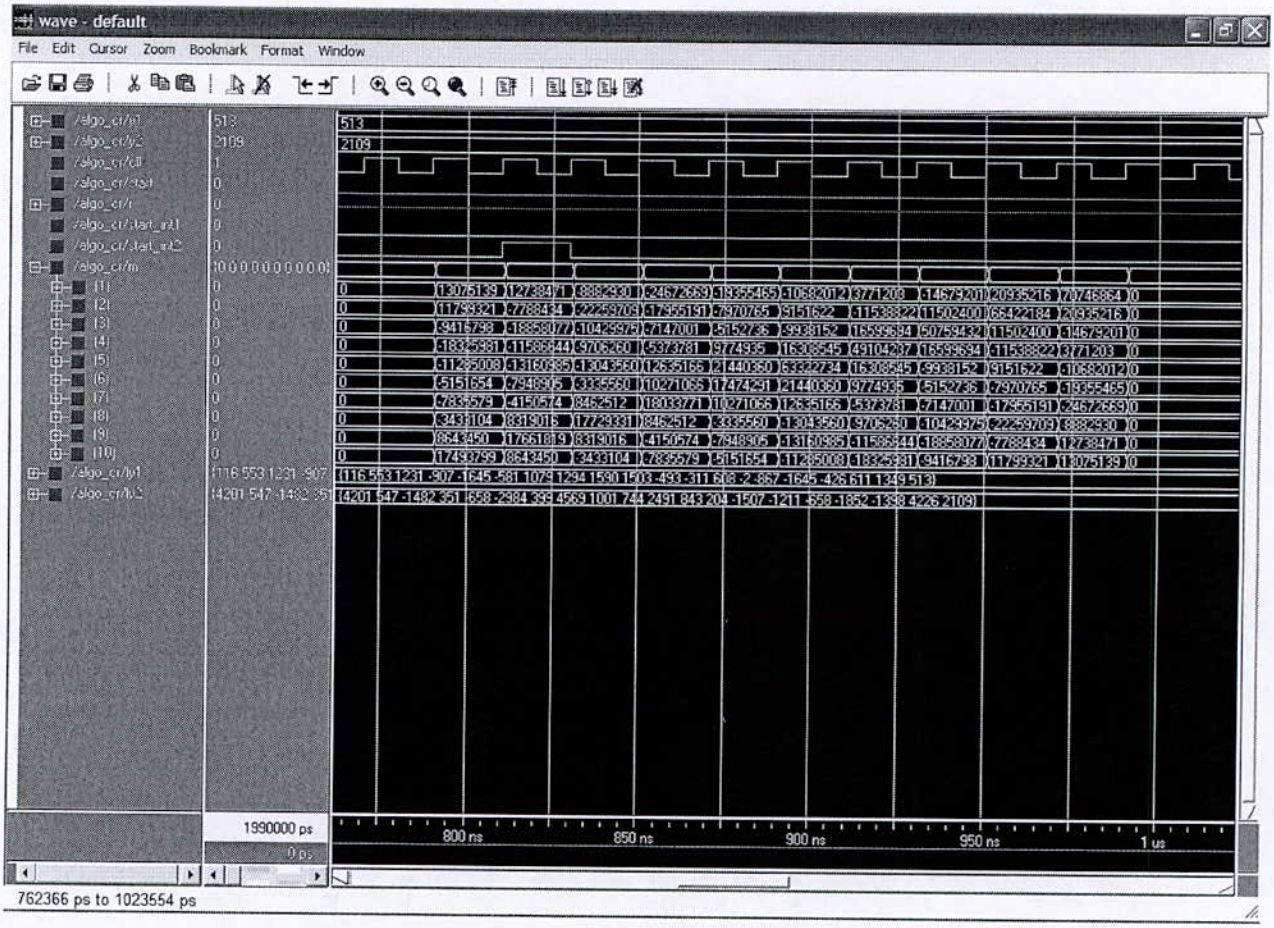


Fig. 3.13 : Simulation fonctionnelle du bloc Multi_trans.

On distingue les signaux suivants :

- Les signaux d'entrée :

  Les signaux **y1** et **y2** : Ils représentent les signaux d'entrée du bloc cal_G (signaux en provenance des capteurs).

  Le signal clk : signal d'horloge de période 20 ns.

  Le signal start : après un temps < 50 ns du front montant de ce signal, les résultats du bloc Form_Y commencent à sortir. Ces résultats représentent la matrice Y (partie 1 équation 4.5.

Les signaux **n**0, **n**1, **y**2p et **a**0 : sont des signaux internes au bloc Form_Y qui servent à la construction de la matrice **Y**.

- Les signaux de sortie :

  Signal **a**: représentent la matrice Y.

- Simulation A globale : bloc cal_G

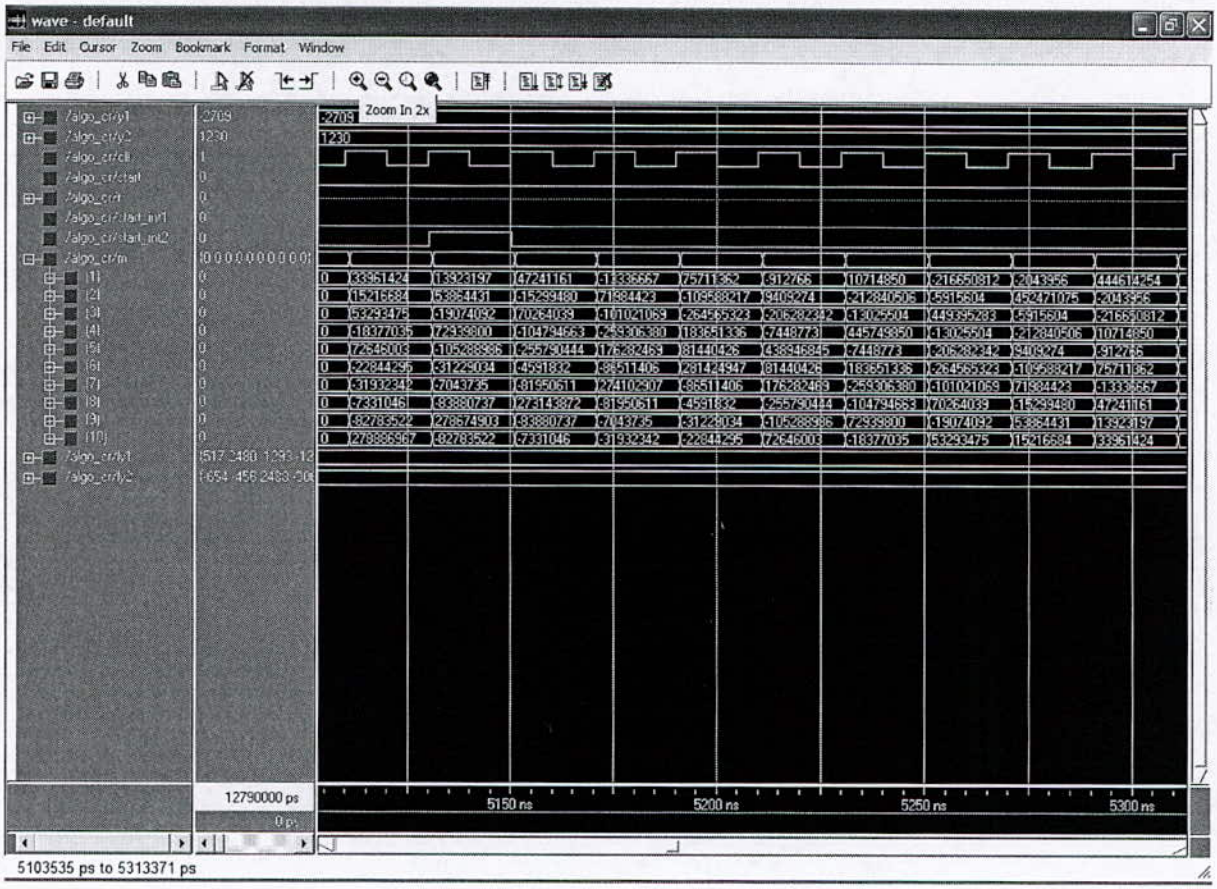


Fig. 3.14 : Simulation fonctionnelle du bloc Cal_G.

On distingue les signaux suivants :

- Les signaux d'entrée :

  Signal **Clk** : l'horloge.
  Signaux **y1,y2** : ce sont les matrices de données de tel sorte que chaque élément a une taille de 16 bits.
  Signal **Start** : c'est un signal qui indique au système le début des opérations.

- Les signaux de sortie :

  Signal **m** : représente la matrice résultante dont les éléments on une taille de 32 bits.

- Simulation B : simulation du bloc resol_system.

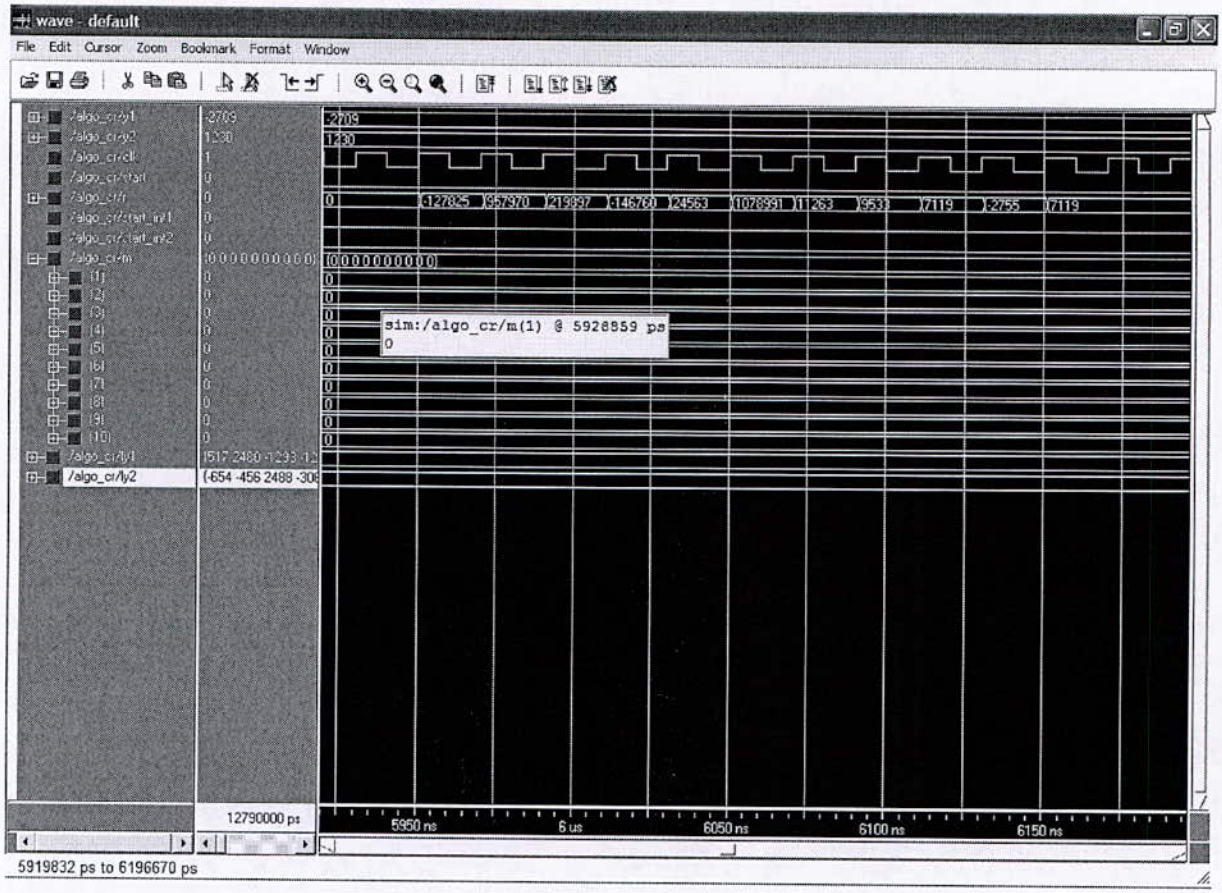


Fig. 3.15 : Simulation fonctionnelle du bloc Resol_syst.

Pour cette simulation, nous avons les signaux suivants :

- Les signaux d'entrée :

  Signal Clk : l'horloge.
  Signal m : représente la matrice G dont les éléments sont de taille 32 bits.
  Signal Start : pour signaler le début des opérations.

- Les signaux de sortie :

  Signal r : représente le résultant dont les éléments sont de taille 32 bits.

- Etudes des simulations fonctionnelles du système globale :

Pour l'étude des simulations du système globale, on considère deux cas :

Premier cas :

On utilise comme entrée un signal non bruité et on compare les résultats obtenus par simulation fonctionnelle. Ceux trouvés par simulation Matlab et les valeurs réelles.

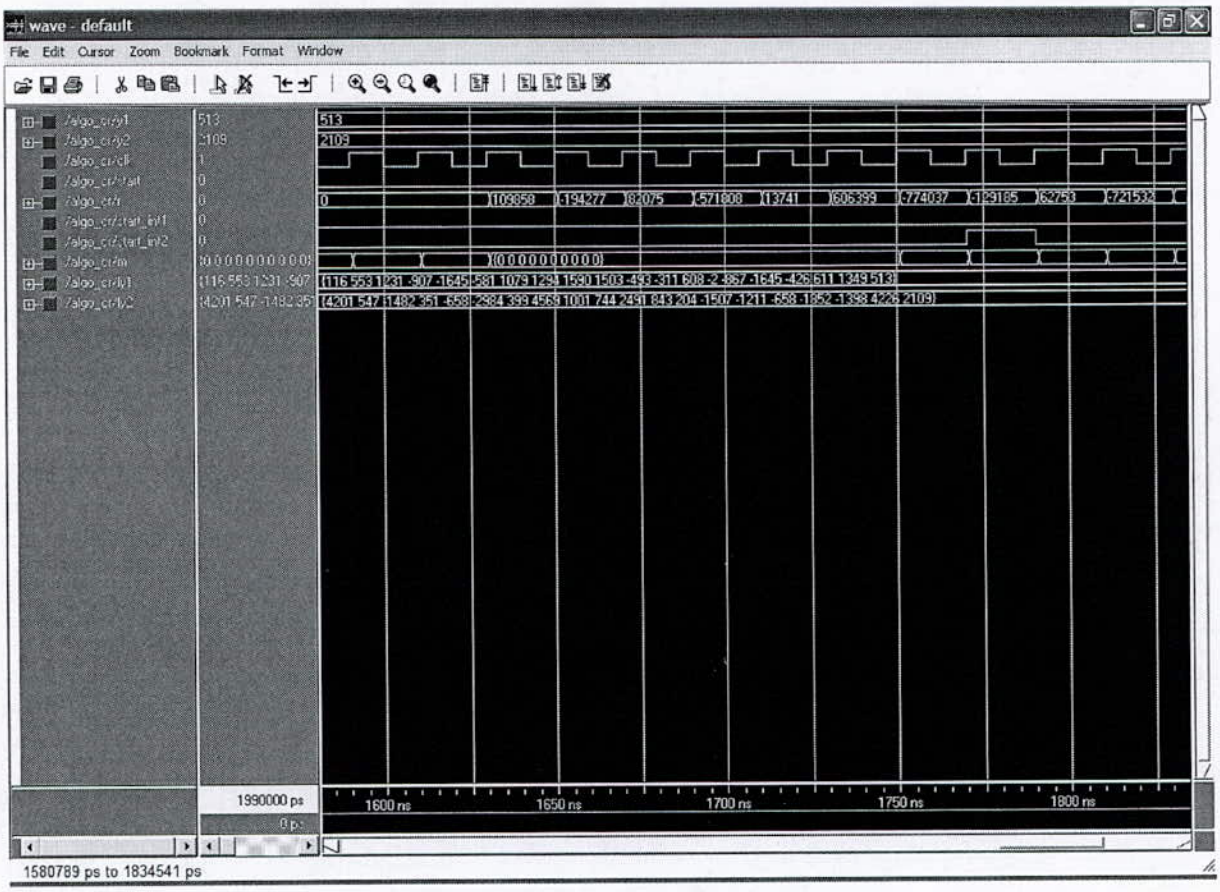


Fig. 3.16 : Simulation fonctionnelle 1 du système globale « Résultats»

Tableau des résultats :

Le tableau suivant regroupe les valeurs des coefficients du canal réel et ceux obtenus par simulation Matlab (valeurs quantifiées) et simulation fonctionnelle de l'implémentation :

Réelles	0.0520	-0.1576	0.0183	-0.4146	0.0137	0.3810	-0.6044	-0.1154	-0.0084	-0.5255
Matlab	0.0708	-0.1468	0.0440	-0.4156	0.0106	0.4206	-0.5780	-0.1014	0.0250	-0.5261
FPGA	0.0799	-0.1412	0.0597	-0.4157	0.0100	0.4409	-0.5628	-0.0939	0.0456	-0.5246

On voit bien que les résultats de l'implémentation sont proches de l'estimation obtenue par l'algorithme CR sous Matlab.

Pour mieux évaluer ce résultat, on calcule l'erreur quadratique en dB de l'estimation Matlab et celle des résultats de l'implémentation :

	Matlab	FPGA
Erreur Quadratique (dB)	-23,27 dB	-19,40 dB

Ces deux valeurs sont différentes mais restent comparables, ce qui signifie que l'estimation du canal donnée par le circuit reste fiable.

Deuxième cas :

Le signal d'entrée dans ce cas est bruité avec un rapport signal sur bruit SNR= 10dB.

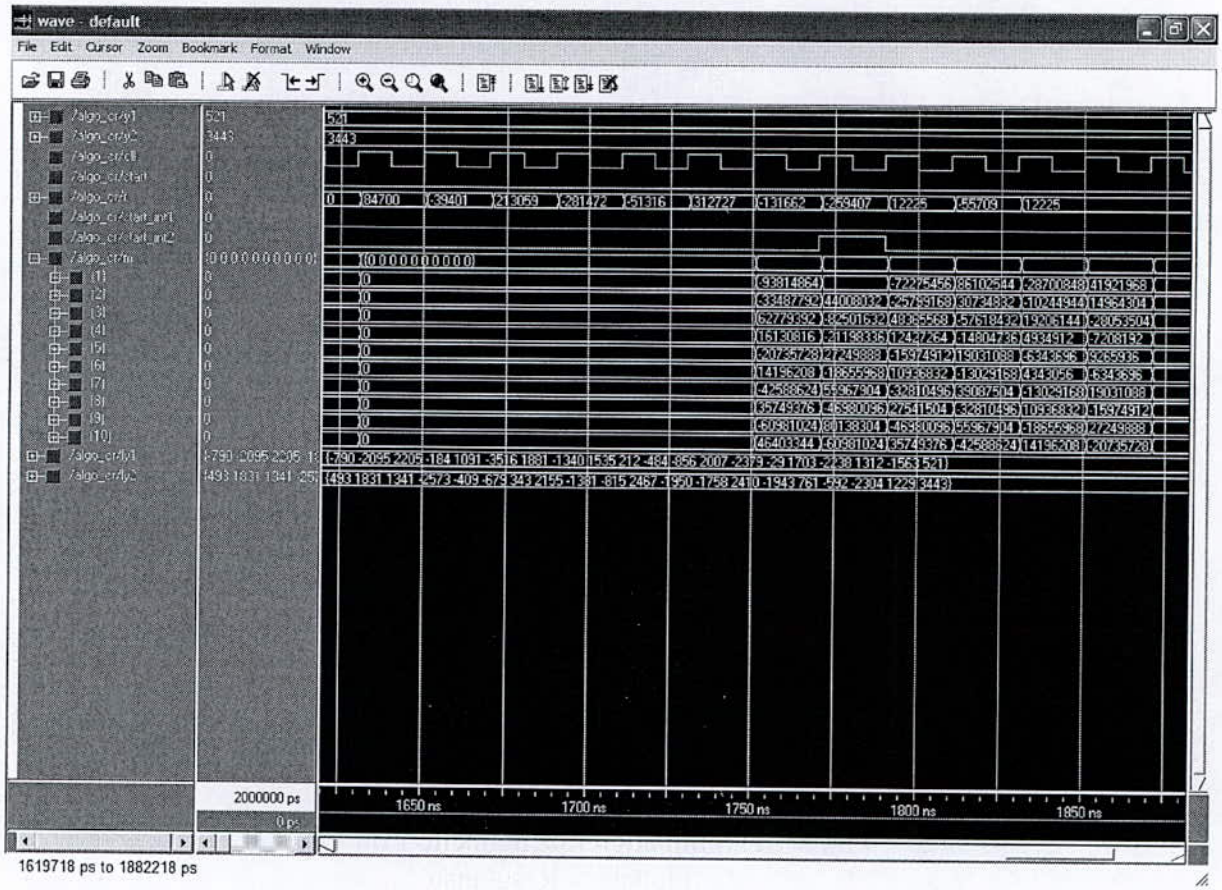


Fig. 3.17 : Simulation fonctionnelle 2 du système globale « Résultats»

Le tableau des résultats de la simulation Matlab et de la simulation de l'implémentation est le suivant :

Réelles	0.1596	-0.0462	0.4230	-0.4388	-0.1208	0.6145	-0.1061	-0.4394	0.0139	-0.0644
Matlab	0.1876	0.0307	0.5617	-0.2115	-0.0515	0.7143	0.2007	-0.2040	0.0795	-0.0499
FPGA	0.1493	-0.0695	0.3757	-0.4963	-0.0905	0.5514	-0.2321	-0.4574	0.0216	-0.0982

D'après les résultats de ce tableau, on voit que même en présence de bruit les résultats de l'implémentation restent comparables aux résultats attendus.

De même que précédemment, on calcul l'erreur quadratique moyenne et on la compare à celle de la simulation Matlab.

	Matlab	FPGA
Erreur Quadratique (dB)	-6,08 dB	-5,45 dB

D'après ces résultats, on constate que l'erreur quadratique en présence de bruit reste comparable à celle obtenue par la simulation Matlab.

- **Etudes des simulations temporelles du système globale :**

On procède à la simulation temporelle après l'implémentation (placement et routage). C'est une simulation identique au simulation fonctionnelle mais cette fois-ci la simulation prend en compte les délais des portes logiques et des interconnexions. Les résultats sont suffisamment précis pour qu'on les acceptes comme référence sans qu'ils soient nécessaires de recourir à des simulations au niveau électrique.

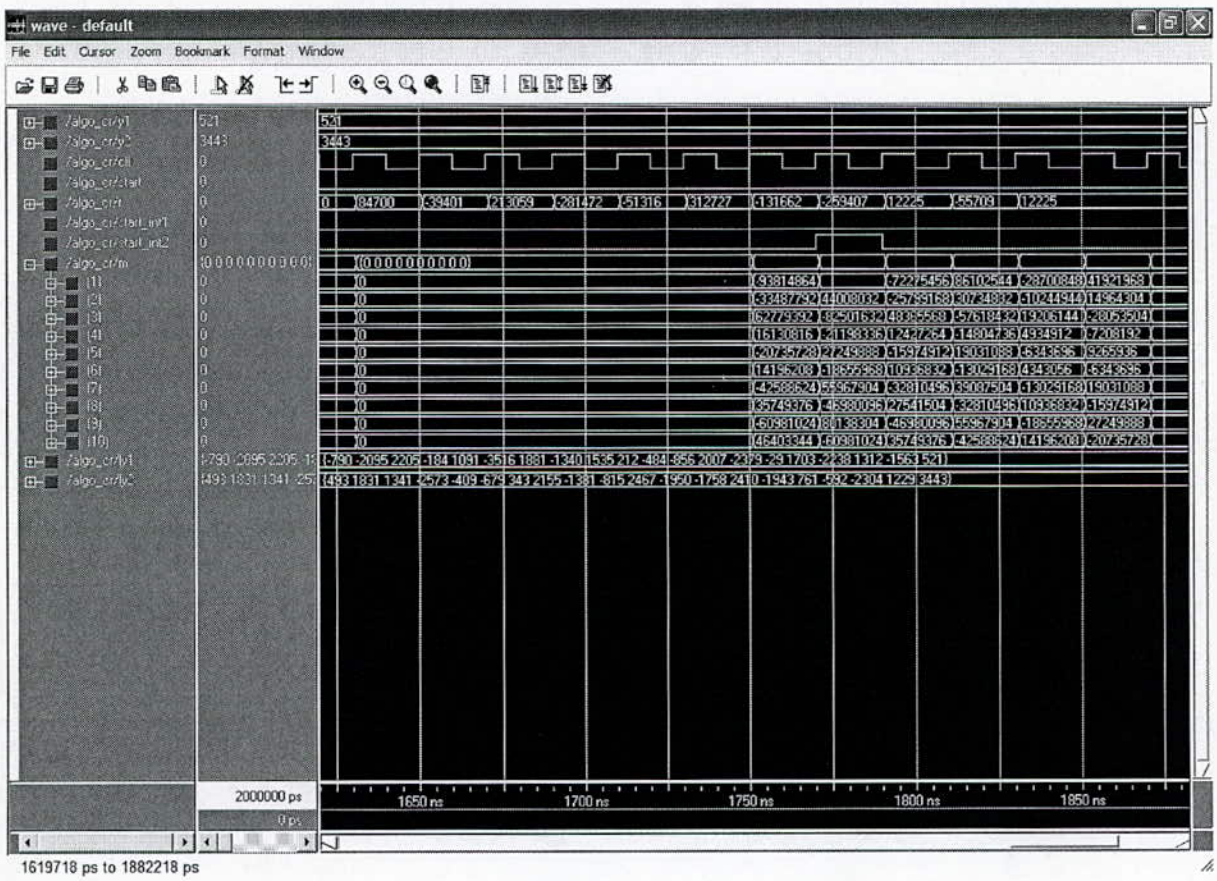


Fig.3.18 : simulation temporelle du système Global.

III-2-4 Synthèse :

La synthèse est l'une des étapes de conception comme mentionné précédemment.
Elle permet d'avoir une description physique de l'architecture conçue, elle prend en entrée le code VHDL et elle génère en sortie une netlist (voir def. Annexe 2) optimisée. Notons que la synthèse se fait de façon hiérarchique de la même manière que la simulation, on synthétise les cellules élémentaires puis les macros avant de synthétisé l'architecture globale.

III-2-5 Implémentation :

La phase de projection dépend du circuit utilisé, les équations de la "netlist" sont transformées, regroupées en de nouvelles équations ayant un nombre d'arguments inférieurs ou égal au nombre de paramètres du bloc logique correspondant à la famille du circuit utilisée. Le mapping estime la surface de l'architecture en terme de slice, registre, multiplieur, DCM…etc.
Cette étape et suivie par l'étape placement et routage où l'on détermine l'emplacement des différents blocs et leurs connexions. A ce niveau de l'implémentation on pourra exécuté l'analyse temporelle qui consiste à déterminer les délais engendrés après placement et routage. Ces délais comme le montre le rapport de l'analyse temporelle (Annexe 5), où on observe un délai maximum de 36.788**ns** qui se décompose en deux type ;

- Le délai engendré par la logique (représente le temps nécessaire à la donnée pour qu'elle soit stable) et qui est égale à 3.107**ns** ce qui représente 8.4% du délai total .
- Le délai engendré par le routage et qui est égale à 33.681**ns** ce qui représente 91.6% du délai total.

Ce rapport détermine la période minimum de l'horloge et qui est égale à 37.594**ns** ce qui représente une fréquence maximum de 26.6**MHz**.

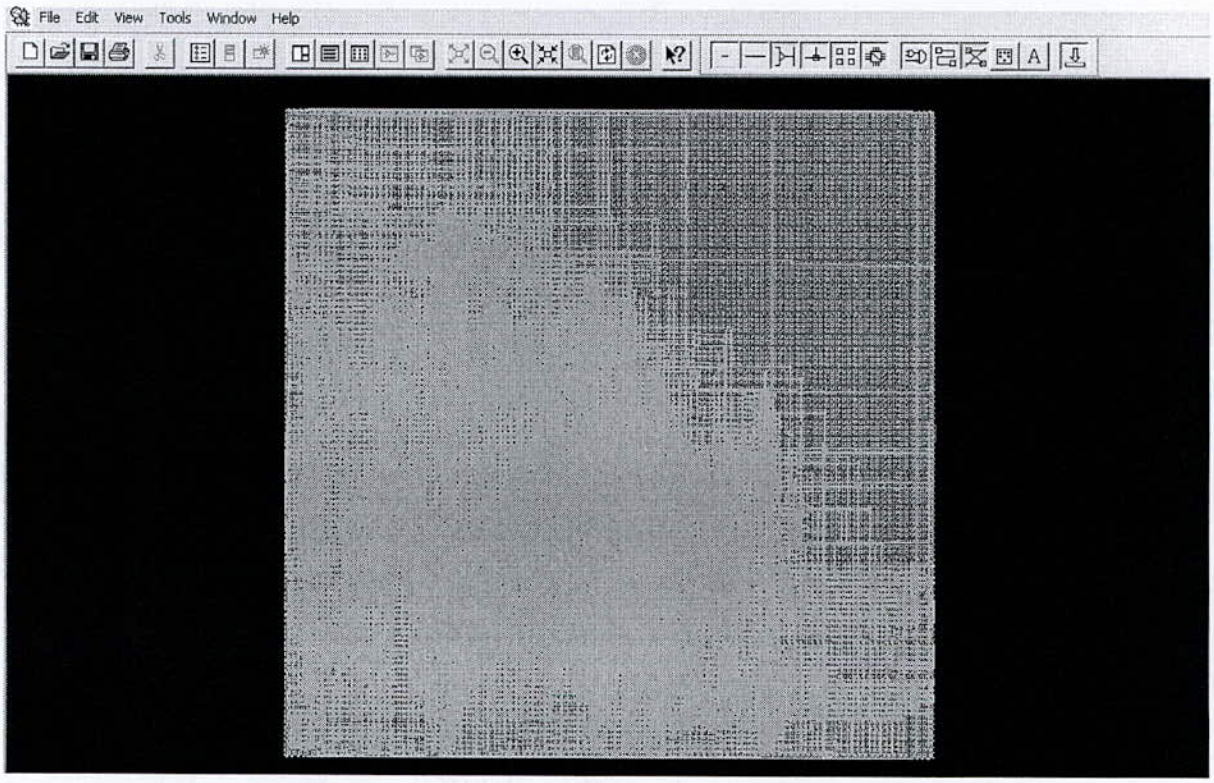


Fig.3.19 : Image du circuit après placement et routage sur le circuit xc2v8000 de Xilinx.

III-3 Conclusion :

Dans ce chapitre, nous avons exposé les étapes suivies durant l'implémentation de l'algorithme CR.

En commençant par donner les opérateurs définis dans les expressions mathématiques, nous avons pu relever les opérations principales de l'algorithme de telle sorte à pouvoir départager les différentes fonctions qui constituent le système globale afin de choisir le type de description (section II-2 partie 2) à adopter pour chacune d'elles.

Mais avant d'aborder l'implémentation de ces fonctions, nous avons d'abord étudié le dimensionnement des données en utilisant le logiciel Matlab pour évaluer l'erreur moyenne introduite par la quantification choisie.

Afin de vérifier le fonctionnement des différents blocs constituant le système globale, nous avant effectuer plusieurs simulations en utilisant le logiciel ModelSim à chacune des étapes du développement, et nous n'avons relevé que celles qui correspondent aux blocs finaux.

Conclusion Générale

Conclusion générale:

Au cours de notre travail, nous avons considéré le problème de l'égalisation aveugle au second ordre de systèmes type SIMO. Autour de ce sujet, nous avons développé notre travail en quatre points essentiels:
Le premier point consistait à effectuer une recherche bibliographique traitant des méthodes d'égalisation aveugle multicapteurs. Cette première étape du travail nous a permit de faire l'étude de plusieurs méthodes d'égalisation aveugle du second ordre pour les systèmes SIMO.
Le second, consistait à élaborer une plate forme logicielle permettant l'évaluation des algorithmes étudiés et de faire une étude comparative entres elles.
Le troisième point a était de développer une étude autour de l'une des méthode étudiée, il s'agit de la méthode des relations croisées, et de lui apporter des modifications afin de minimiser son coût de calcul ce qui a donné naissance à deux nouvelles versions à cette méthode.
Quant au quatrième point, il s'agissait de réaliser l'implémentation de l'algorithme CR sur un circuit FPGA, ce qui nous a poussé à faire l'étude d'une telle conception en réalisant les étapes du cycle de conception en commençant d'abord par l'étude des caractéristiques de ce type de circuits.

Perspectives:

De nombreuses perspectives permettent de prolonger ce travail afin d'étendre son application aux cas les plus générales. Ces perspectives concernent la plate forme conçue ainsi que l'implémentation de l'algorithme CR sur circuit FPGA. Nous les regroupons comme suit:

- Au niveau de la plate forme:

  - Rajouter une option démonstration à la plate forme en utilisant des signaux parole ou image afin de mettre en évidence les distorsions que subissent ces signaux par l'effet du canal de transmission, mais aussi mettre en évidence la capacité des algorithmes présents sur la plate forme à minimiser ces distorsions et reconstituer les signaux d'origine.
  - Réaliser la plate forme sous un autre langage que Matlab afin de permettre la portabilité du logiciel et rendre ainsi son application indépendante de son environnement.

- Au niveau de l'implémentation sur FPGA:

  - Réaliser une implémentation de l'algorithme pour un nombre de capteurs supérieur à deux, et rendre ainsi son application plus étendue.
  - Implémenter l'algorithme CR sans apporter de changement à sa solution (c'est à dire garder la contrainte initiale développée au chapitre 4 de la première partie), ainsi préserver les performances de l'algorithme.
  - réaliser une conception complète d'un circuit dont l'application est l'algorithme CR.

Annexes

Annexe 1 :

Etat de l'art des FPGAs de Xilinx :

1 - Introduction (propriétés) :

Xilinx est le pionnier et le leader des solutions logiques programmables complètes – circuits intégrés avancés, outils de conception de logiciel, fonctions systèmes prédéfinis et assistance sur site à la conception inégalée. Fondée en 1984, Xilinx a son siège à San José, en Californie et est la société qui a inventé le FPGA (Field Programmable Gate Array). Elle contrôle actuellement plus de la moitié du marché mondial de ces composants. Les solutions Xilinx permettent à sa clientèle de réduire sensiblement le temps nécessaire au développement de produits destinés aux marchés informatique, des périphériques, des télécommunications, des réseaux, du contrôle industriel, de l'instrumentation, de l'aérospatiale, de la Défense et des produits électroniques grand public.

Pour ses besoins de simulation, Xilinx utilise ModelSim, outil de simulation HDL le plus utilisé dans le monde, permet aux concepteurs de vérifier de grands blocs d'IP (Intellectual Property) et d'accélérer ainsi leur intégration aux FPGA de Xilinx. ModelSim permet aux concepteurs de FPGA de simuler les IP de Xilinx dans les environnements en langage Verilog, VHDL ou mixte.

Xilinx s'est lancé depuis le début dans la fabrication des circuits FPGAs, et n'a cessé de progresser et d'améliorer de plus en plus la conception de ces derniers. De ce fait, on voit régulièrement la mise sur le marché de nouvelles séries ou de nouvelles familles de FPGAs :

XC2000 : première génération de LCA (Logic Cells Array) (1985), elle comprend 2 produits d'une complexité allant de 600 à 1.500 portes logiques utilisables ;
XC3000 : deuxième génération de LCA (1987), la gamme de produit est plus étoffée que la famille précédente et couvre une gamme de complexité allant de 1.000 à 6.000 portes logiques utilisables ;
XC4000 : troisième génération de LCA (1991), cette famille couvre une gamme de complexité allant de 2.000 à 500.000 portes logiques utilisables ;
XC5200 : quatrième génération de LCA (1995), cette famille couvre une gamme de complexité allant de 6.000 à 18.000 portes logiques utilisables ;
XC6200 : cinquième génération de FPGA, cette nouvelle approche de composant programmable (1996) vise le créneau du « coprocessing », c'est à dire l'environnement de processeurs ; la gamme de complexité de la famille s'étend de 10.000 à 100.000 portes ;
XC8100 : nouvelle orientation de *Xilinx* en 1996 : l'approche « mer de portes » mais *Xilinx* a abandonné cette voie en 1997 ;
SPARTAN : sixième génération de FPGA (1998), cette famille vise la souplesse d'utilisation et des performances élevées ;
VIRTEX : septième génération de FPGA (1999), cette famille vise les très fortes capacités (4 millions de portes) et les hautes vitesses (> 100 MHz).

Ces différentes familles se distingues les unes des autres principalement par ce que chacune d'elles apporte par rapport à ses prédécesseurs, et les caractéristiques les plus distinctives sont entre autres : un moindre coût , plus de capacité mémoire, une plus grande vitesse de transfert, plus de blocs d'IP permettant un plus grand contrôle et une plus grande flexibilité de conception, l'ajout de certaines fonctions élémentaires (comme les multiplieurs dans le cas la

famille Virtex-II), une plus grande vitesse d'horloge, utilisation d'une technologie plus avancée,...

La technologie employée est une technologie CMOS performante :
2 µm avec 2 niveaux de métallisation en 1986 ;
1 µm avec 2 niveaux de métallisation en 1991 ;
0,8 µm avec 2 niveaux de métallisation en 1992 ;
0,6 µm avec 3 niveaux de métallisation en 1995 ;
0,5 µm avec 4 niveaux de métallisation en 1997 ;
0,35 µm avec 4 niveaux de métallisation en 1998 ;
0,22 µm avec 5 niveaux de métallisation en 1999 ;
0,15 µm avec 8 niveaux de métallisation en 2002 ;
0,09 µm ;

Dans ce qui suit, nous allons nous intéresser à l'une des famille de FPGAs de Xilinx qui nous a intéressé lors de l'implémentation de notre algorithme, et qui est la famille Virtex-II, celle-ci étant la dernière version de FPGAs disponibles.

2 - La famille Virtex-II :

2-1 Généralités sur les produits Virtex-II :

La famille Virtex-II fournit des FPGAs de plus en plus élaborés , possédant les fonctions les plus avancées qu'une conception pourrait demander :
multiplieurs intégrés ; blocs contrôle d'horloge DCMs (advanced Digital Clock Managers) ;XCITE technology (Digitally Controlled Impedance) ; ...

La famille Virtex-II représente le premier groupe de circuits FPGAs dans la série Virtex-II , elle est constituée de onze types de circuits classés selon la densité de portes utilisables (system gate) des circuits allant de 40K portes à 8M portes. Cet ensembles de circuits (les 11 types) offrent tous les même fonctions indépendamment de leur densité.

Virtex-II Device	Maximum System Gates	Logic Cells	Embedded BRAM Memory	Maximum Distributed Memory	18x18 Multiplier Blocks	DCMs	Maximum I/Os
XC2V8000	8M	104,832	3.024 Mbits	1.456 Mbits	168	12	1,108
XC2V6000	6M	76,032	2.592 Mbits	1.056 Mbits	144	12	1,104
XC2V4000	4M	51,840	2.160 Mbits	720 Kbits	120	12	912
XC2V3000	3M	32,256	1.728 Mbits	448 Kbits	96	12	720
XC2V2000	2M	24,192	1.008 Mbits	336 Kbits	56	8	624
XC2V1500	1.5M	17,280	864 Kbits	240 Kbits	48	8	528
XC2V1000	1M	11,520	720 Kbits	160 Kbits	40	8	432
XC2V500	500K	6,912	576 Kbits	96 Kbits	32	8	264
XC2V250	250K	3,456	432 Kbits	48 Kbits	24	8	200
XC2V80	80K	1,152	144 Kbits	16 Kbits	16	4	120
XC2V40	40K	596	72 Kbits	8 Kbits	8	4	88

Tab.1 : Produits Virtex-II.

Cette diversité de circuits permet aux utilisateurs une grande flexibilité de déplacement d'un circuit à un autre, plus petit ou plus grand, sans se soucier des fonctions que chacun pourrait fournir.

Ce que Virtex-II apporte de plus par rapport à ses prédécesseurs ? :

Virtex-II est la première à avoir intégré : les DCMs, Xtreme Multipliers , XCITE technology ((1) annexe 2) ;

Les CLBs (Configurable Logic Bloc) de Virtex-II possèdent deux fois plus de slices ;

Une plus grande capacité mémoire ;(voir figure comparative fig.1)

Les FPGAs de Virtex-II se caractérisent d'une très grande souplesse d'adaptation aux applications du traitement du signal.

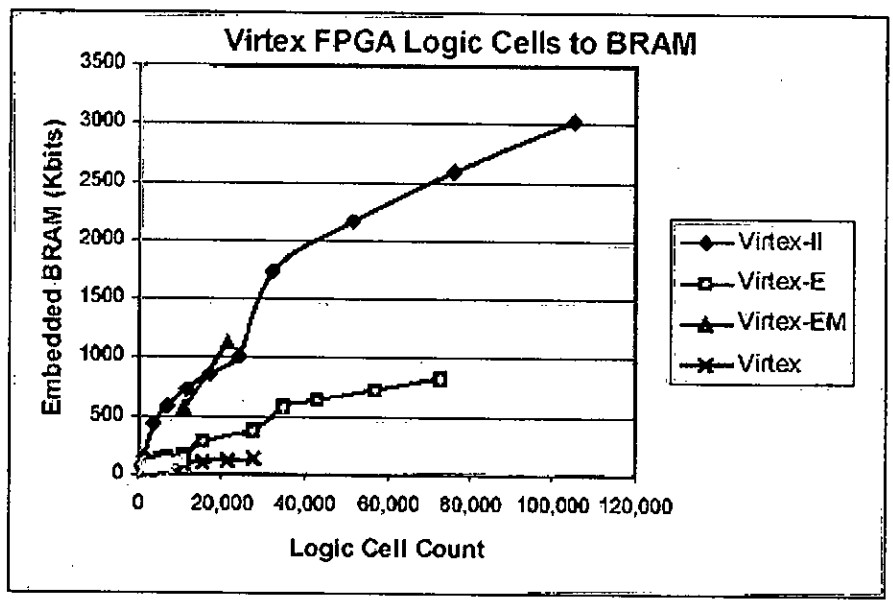


Fig.1 : Capacité mémoire.

2-2 Les circuits Virtex-II :

La famille Virtex-II est l'ensemble de ce qu'on appelle des plates-formes FPGA.

Développée pour de hautes performances, pour des conceptions basées sur des blocs d'IP (Intellectual Property) et des blocs re-programmables.

La famille Virtex-II fournit des FPGAs à de diverses densités d'intégration allant de la plus faible à la plus importante dépassant les 10 millions de portes

- Résumé des principales caractéristiques des circuits Virtex-II :

Les circuits FPGA Virtex-II de Xilinx sont principalement caractérisés par **[1]** :

- Première plate-forme proposée dans l'industrie mondiale.
- Grand usage des architectures à base de blocs d'IP :
  - densité d'intégration allant de 40 K à 8 M de portes ;
  - une horloge interne de 420 MHz ;
  - transfert de haut débit 840 Mb/s ;

- Bonne distribution des blocs mémoire :

  • une mémoire de 3Mb de dual-port RAM composée de blocs de 18 Kb de SelectRAM ;

  • plus de 1.5Mb de blocs SelctRAM distribuée ;

- Bonne interfaçage avec les ressources mémoire externes :
  • Interface DRAM :
    - SDR / DDR SDRAM ;
    - Network FCRAM ;
    - DRAM à Latency ((1) annexe 1) réduite ;
  • Interface SRAM ;
  • Interface CAM ;
- Fonctions arithmétiques :
  • des blocs de multiplieurs de taille 18 bits x 18 bits ;
  • propagation de la retenus très rapide ;

- Une grande disponibilité de ressources logiques :
  • plus de 93.184 registres internes avec l'option Clock Enable ;
  • plus de 93.184 LUTs (voir définition plus loin) ou registres à décalage de 16 bits ;
  • un grand nombre de multiplexeurs ;

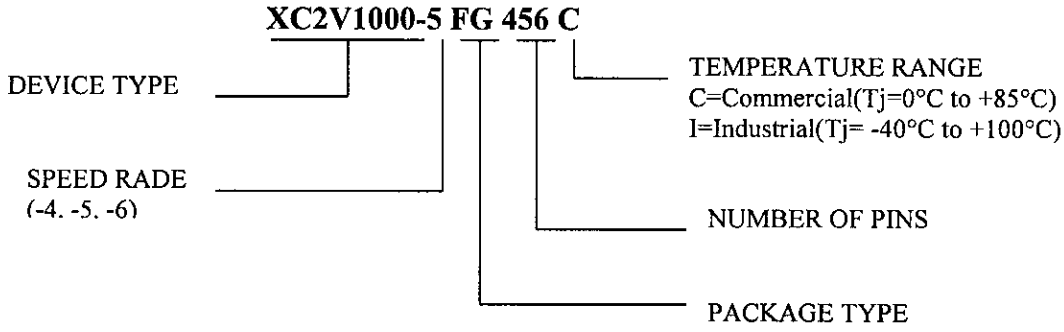
- Un bloc de contrôle du signal d'horloge (DCM) :
  • plus de 12 modules DCM (Digital Clock Manager) ;
  • 16 global clock multiplexer buffers ;

Comme mentionné précédemment , la famille Virtex-II se compose de 11 éléments . Le tableau suivant **[1]** , résume certaines des caractéristiques de chacun de ces éléments :

Device	System Gates	CLB (1 CLB = 4 slices = Max 128 bits)			Multiplier Blocks	SelectRAM Blocks		DCMs	Max I/O Pads[1]
		Array Row x Col.	Slices	Maximum Distributed RAM Kbits		18 Kbit Blocks	Max RAM (Kbits)		
XC2V40	40K	8 x 8	256	8	4	4	72	4	88
XC2V80	80K	16 x 8	512	16	8	8	144	4	120
XC2V250	250K	24 x 16	1,536	48	24	24	432	8	200
XC2V500	500K	32 x 24	3,072	96	32	32	576	8	264
XC2V1000	1M	40 x 32	5,120	160	40	40	720	8	432
XC2V1500	1.5M	48 x 40	7,680	240	48	48	864	8	528
XC2V2000	2M	56 x 48	10,752	336	56	56	1,008	8	624
XC2V3000	3M	64 x 56	14,336	448	96	96	1,728	12	720
XC2V4000	4M	80 x 72	23,040	720	120	120	2,160	12	912
XC2V6000	6M	96 x 88	33,792	1,056	144	144	2,592	12	1,104
XC2V8000	8M	112 x 104	46,592	1,456	168	168	3,024	12	1,108

Tab.2 : Famille Virtex-II.

3 - Nomenclature des circuits FPGA :



4 - Derniers FPGAs de Xilinx :

En mars 2002 , les deux géants de la micro-électronique, IBM fabriquant numéro 1 mondial des circuits ASIC et Xilinx , fabriquant numéro 1 mondial de circuits programmables type PLD, et avec plus de 15 années d'expérience sur les outils de développement à base des FPGAs, décident de s'unir pour donner naissance à une nouvelle génération de circuits programmables appelés "FPGA Virtex-II Pro". **[3]**

Les FPGAs Virtex-II Pro contiennent, en plus des éléments déjà existant dans les anciennes version de circuits FPGA (IOB, blocs RAM, multiplieurs, CLB, DCM,...) , des microprocesseurs dans leur architecture interne.

Les FPGAs Virtex-II Pro contiennent :
4 microprocesseurs PowerPC d'IBM ;
24 blocs Rocket I/O Multi-Gigabit Transceivers ;
12 blocs DCM ;
556 blocs multiplieurs de 18 x18 bits ;
des blocs RAM de 10 Mb ;
XCITE Digitally Controlled Impedance Technology.

Le tableau suivant donne la liste de tous les circuits FPGA de la famille Virtex-II Pro avec les principales performances de chacun :

Circuits	XC2VP2	XC2VP4	XC2VP7	XC2VP20	XC2VP30	XC2VP40	XC2VP50	XC2VP70	XC2VP 100	XC2VP 125
Logic Cells	3,168	6,768	11,088	20,880	30,816	43,362	53,136	74,448	99,216	125.136
Block RAM(Kbits)	216	504	792	1,584	2,448	3,456	4,176	5,904	7,992	10,008
18x18 Multiplieurs	12	28	44	88	136	192	232	328	444	556
Digital Clock Manager Blocks	4	4	4	8	8	8	8	8	12	12
Configuration Memory (Mbits)	1,31	3,01	4,49	8,21	11,36	15,56	19,02	25,6	33,65	42,78
IBM PowerPC Processors	0	1	1	2	2	2	2	2	2	4
Multi-Gigabit Transceivers	4	4	8	8	8	12	16	20	20	24
Max Available User I/O	204	348	396	564	692	804	852	996	1164	1200

Tab.3 : Nouveaux produits.

Les cases marquées en rouge représentent les 5 derniers circuits de la famille Virtex-II Pro.

- *Perspective :*

Les FPGAs de Xilinx intégrés à la dernière version des ASICs de IBM , sont une nouvelle génération de circuits combinant la flexibilité des FPGAs et la densité , la performance et le moindre coût à la fabrication des ASICs.
Avec 8 niveaux de métallisation séparés par des isolants de haute technologie, cette nouvelle génération de circuits programmables supportera plus de 72 millions de portes pour des CI de très grande complexité.
Mis sur le marché début 2004, ce nouveau type de circuits prouvera, d'après ses concepteurs , son efficacité et sa simplicité dans l'industrie de la télécommunication où différents protocoles et caractéristiques d'interfaçage ne cessent d'évoluer. Sans oublier tous les autres produits électroniques à consommation publique qui bénéficieront de même de cette nouvelle approche.

Annexe 2 :

Définitions :

(1) : Latency : c'est la durée qui sépare l'entrée du premier élément et la sortie du dernier élément calculé .en d'autres termes c'est le temps que prendrait une donnée pour se déplacer le long du trajet critique.

(2) : Régularité : elle a une relation avec l'espace, indépendamment du temps. La connaissance du schéma élémentaire d'un ou de quelques processeur élémentaires ,avec leurs interconnexions, permet de générer l'architecture complète.

(3) : Localité : possède des caractéristiques spatio-temporelles ;
- spatiale : chaque processeur ne communique qu'avec ses voisins immédiats.
- temporelle : le transfert des données d'un processeur à un autre se fait en un seul top d'horloge.

(4) : I/O bandwidth : c'est le nombre d'éléments pouvant être transférés à chaque cycle d'horloge.

(5) : Scalabilité : c'est la capacité à maintenir la Latency proportionnelle au nombre de processeurs élémentaires utilisés quelque soit la dimension du problème.

(6) : Problem-size-independence (ψ) : un algorithme parallèle ou une architecture est considérée ψ, si pour une taille donnée du réseau utilisé, n'importe quel exemple introduit peut être résolu.

(7) : Efficiency : est définie comme étant le rapport entre la vitesse et le nombre de processeurs utilisés .

(8) : XCITE technology : les FPGAs de Virtex-II sont les seuls à fournir une technologie permettent le contrôle d'impédance appelée XCITE technology. Les circuits intégrés de cette technologie sont munis, à leurs pinnes d'entrée et de sortie, d'un nombre variable d'impédances mises en parallèle , pouvant être configurées soit en parallèle soit en série, offrant divers avantages : une plus large BP pour les E/S ; immunité contre les variations indésirables de température et de tension à ses bornes ; une plus grande précision en réduisant le nombre de composants ; élimination des pertes par réflexion en éliminant la distance entre charge et résistances ; **[5]**

(9) : Package : le package est une bibliothèque pouvant être créée (en plus des bibliothèques existant déjà) pour définir un nouveau type de variables, des sous programmes (procédures et function) . **[11]**

(10) : Netlist : description des cellules logiques et des différentes connexions.**[11]**

Annexe 3 :

Programmes VHDL :

1- My_Type:

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;

package my_type is

 constant Trame : integer := 128;
 constant Memoire : integer :=4;
 constant colonne : integer := (Trame-Memoire);
 constant Matrix_Size : integer :=2*(Memoire + 1);

 subtype donnee is std_logic_vector (15 downto 0);
 subtype element is std_logic_vector(31 downto 0);
 subtype result is std_logic_vector(63 downto 0);
 subtype elem_cmd is std_logic;

 type trame_lin is array (1 to Trame) of donnee;
 type mat_col_d is array (1 to colonne) of donnee;
 type mat_lin_d is array (1 to Matrix_Size) of donnee;
 type mac_mac_lin is array (1 to Matrix_Size*(Matrix_Size-1)) of element;
 type mat_lin_in is array (1 to Matrix_Size) of element;
 type mat_lin is array (1 to Matrix_Size+1) of element;
 type mas_mas_lin is array (1 to ((Matrix_Size+1)*(Matrix_Size/2))) of element;
 type mas_mas_cmd is array (1 to ((Matrix_Size+1)*(Matrix_Size/2))) of elem_cmd;
 type mas_mas_start is array (1 to Matrix_Size) of elem_cmd;

end my_type;
```

2- Algo_CR :

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
use Work.My_type.all;

entity Algo_CR is
   Port ( Y1 : in donnee;
                Y2 : in donnee;
                              clk : in std_logic;
                              start : in std_logic;
                        R : out element );
```

```
end Algo_CR;
architecture Behavioral of Algo_CR is

 component cal_G
   Port ( Y1 : in trame_lin;
              Y2 : in trame_lin;
                        clk_pad : in std_logic;
                        start : in std_logic;
                        start_out : out std_logic;
                        P : out mat_lin_in);
 end component;

 component resol_system
   Port ( D : in mat_lin_in;
              clk : in std_logic;
                        start : in std_logic;
                        R : out element);
 end component;

 component loader
    Port( Y1 : in donnee;
         Y2 : in donnee;
                   LY1 : out trame_lin;
                   LY2 : out trame_lin;
                   clk : in std_logic;
                        start_in : in std_logic;
                        start_out : out std_logic);
 end component;


 signal start_int1,start_int2 : std_logic;
 signal M : mat_lin_in;
 signal LY1,LY2 : trame_lin;

begin


 U1 : loader port
map(Y1=>Y1,Y2=>Y2,clk=>clk,LY1=>LY1,LY2=>LY2,start_in=>start,start_out=>start_int1);

 U2 : cal_G port
map(Y1=>LY1,Y2=>LY2,start_out=>start_int2,clk_pad=>clk,start=>start_int1,P=>M);

 U3 : resol_system port map(D=>M,clk=>clk,start=>start_int2,R=>R);


end Behavioral;
```

3- Loader :

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
use Work.My_type.all;

entity loader is
   Port ( Y1 : in donnee;
        Y2 : in donnee;
                LY1 : out trame_lin;
                LY2 : out trame_lin;
                clk : in std_logic;
                      start_in : in std_logic;
                      start_out : out std_logic );
end loader;

architecture Behavioral of loader is

 component bascule_d
    port( D : in std_logic;
                clk : in std_logic;
                      Q : out std_logic);
 end component;

 signal start_int1 : std_logic;
 signal start_int2 : std_logic;

begin


 process(clk,start_in)

 variable i : natural range 0 to 255 :=1;
 variable j : natural range 0 to 1 :=0;

 begin

  if(clk'event and clk='1') then

    if start_in='1' then

          j:=1;
                i:=1;

                start_int1 <='0';

         end if;
```

```
      if j=1 then

          if i <= Trame then

                LY1(i) <= Y1;
                LY2(i) <= Y2;

                i:=i+1;

                if i=Trame then
                  start_int1 <= '1';
                end if;


              end if;

              if i=(Trame+1) then

                j:=0;
                i:=1;

                start_int1 <= '0';

              end if;

        end if;

 end if;

end process;

U1: bascule_d port map(D=>start_int1,Q=>start_int2,clk=>clk);

U2: bascule_d port map(D=>start_int2,Q=>start_out,clk=>clk);



end Behavioral;
```

4- cal_G :

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
use Work.My_type.all;

entity cal_G is
   Port ( Y1 : in trame_lin;
          Y2 : in trame_lin;
                clk_pad : in std_logic;
                start : in std_logic;
                start_out : out std_logic;
```

```
                        P : out mat_lin_in);
end cal_G;
architecture Behavioral of cal_G is
 component forme_y
   Port ( Y1 : in trame_lin;
          Y2 : in trame_lin;
                        clk : in std_logic;
                        start : in std_logic;
                        start_out : out std_logic;
                        A : out mat_lin_d);
 end component;

 component multi_trans
   Port ( A : in mat_lin_d;
          B : in mat_lin_d;
                        Ck : in std_logic;
                        start : in std_logic;
                        start_out : out std_logic;
                        P : out mat_lin_in);
 end component;

 component buf_ck
  port( in1 : in std_logic;
       out1 : out std_logic);
 end component;


 signal start_int : std_logic;
 signal A : mat_lin_d;
 signal clk : std_logic;

begin

 U1 : buf_ck port map(in1=>clk_pad,out1=>clk);

 U2 : forme_Y port map(Y1=>Y1,Y2=>Y2,clk=>clk,start=>start,
                start_out=>start_int,A=>A);

 U3 : multi_trans port map(A=>A,B=>A,ck=>clk,start=>start_int,
                     start_out=>start_out,P=>P);

end Behavioral;
```

5- Form_Y :

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
use Work.My_type.all;

entity forme_Y is
   Port ( Y1 : in trame_lin;
```

```
            Y2 : in trame_lin;
                    clk : in std_logic;
                    start : in std_logic;
                    start_out : out std_logic;
                    A : out mat_lin_d);
end forme_Y;

architecture Behavioral of forme_Y is

 component negation
  port( P : in donnee;
       N : out donnee);
 end component;

 component tram_reg
 port ( A : in trame_lin;
       clk : in std_logic;
            B : out trame_lin);
 end component;

 component size_reg
  port( A : in mat_lin_d;
       clk : in std_logic;
            B : out mat_lin_d);
 end component;

 component bascule_d
 port( D : in std_logic;
      clk : in std_logic;
            Q : out std_logic);
 end component;

 signal N0,N1,Y2p : trame_lin;
 signal A0 : mat_lin_d;
 signal start_int : std_logic;

begin

 U1 : for s in 1 to trame Generate

  U2 : negation port map (P=>N0(s),N=>N1(s));

  end Generate;

 U3 : tram_reg port map(A=>Y1,B=>N0,clk=>clk);
 U4 : tram_reg port map(A=>Y2,B=>Y2p,clk=>clk);
 U5 : bascule_d port map(D=>start,Q=>start_int,clk=>clk);
 U6 : bascule_d port map(D=>start_int,Q=>start_out,clk=>clk);
 U7 : size_reg port map(A=>A0,B=>A,clk=>clk);
```

```
process(clk)

variable i : integer range 0 to 2047 :=0;
variable z : integer range 0 to 2047 :=0;

 begin

 if (clk'event and clk='1') then

-- pour la sortie A

 if (z <= colonne and z>0) then

T3 : for jj in 1 to Matrix_Size loop

    if jj <= (Memoire+1) then

                  A0(jj) <= Y2p(Memoire+z-jj+1);

                end if;

                if (jj > (Memoire+1) and jj<=Matrix_Size) then

                  A0(jj) <= N1(Matrix_Size+z-jj);

                end if;

        end loop;

 end if;

 if (start='1') then

                z:=0;

 end if;
 z:=z+1;

        end if;

 end process;


end Behavioral;
```

6- Multi_trans :

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
use Work.My_type.all;
```

```
entity multi_trans is
   Port ( A : in mat_lin_d;
                B : in mat_lin_d;
                        Ck : in std_logic;
                        start : in std_logic;
                        start_out : out std_logic;
                        P : out mat_lin_in);
end multi_trans;

architecture Behavioral of multi_trans is

 component buf_rst
  port( in1 : in std_logic;
      out1 : out std_logic);
 end component;

 component MAC
  port (A,B : in donnee;
       clk,reset : in std_logic;
                   mac_in : in element;
                   mac_out : out element);
 end component;

 component controler
  port( start : in std_logic;
       start_out : out std_logic;
       reset_asyn : out std_logic;
                   clk : in std_logic);
 end component;

 signal zero : element :=X"00000000";
 signal reset_syn_I,reset_syn : std_logic;
 signal mac_mac : mac_mac_lin;

begin

 U2 : buf_rst port map(in1=>reset_syn_I,out1=>reset_syn);

 U3 : controler port map(clk=>ck,start_out=>start_out,start=>start,reset_asyn=>reset_syn_I);

 S1 : for I in 1 to Matrix_Size Generate
 S2 : for J in 1 to Matrix_Size Generate
 S3 : if J=1 Generate
          M1 : mac port map(A=>A(I),B=>B(1),clk=>ck,reset=>reset_syn,
                      mac_in=>zero,mac_out=>mac_mac(1+(Matrix_Size-1)*(I-1)));

                         end Generate;

          S4 : if (J >=2 and J<= (Matrix_Size-1)) Generate
```

```
        M2 : mac port map(A=>A(I),B=>B(J),clk=>ck,reset=>reset_syn,
                    mac_in=>mac_mac((Matrix_Size-1)*(I-1)+J-
1),mac_out=>mac_mac(J+(Matrix_Size-1)*(I-1)));

            end Generate;

        S5 : if J=Matrix_Size Generate

        M3 : mac port map(A=>A(I),B=>B(J),clk=>ck,reset=>reset_syn,
                    mac_in=>mac_mac((Matrix_Size-1)*I),mac_out=>P(I));

            end Generate;

  end Generate;

 end Generate;

end Behavioral;
```

7- Mac :

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
use Work.My_type.all;

entity mac is
   Port ( A : in donnee;
        B : in donnee;
                    clk : in std_logic;
                    reset : in std_logic;
                    mac_in : in element;
                    mac_out : out element);
end mac;

architecture Behavioral of mac is

 component multi_16
  port( A : in donnee;
      B : in donnee;
              O : out element);
 end component;

 component mac_adder
  port( A,B : in element;
       S : out element);
 end component;

 component reg_16
  port( D : in donnee;
       Q : out donnee;
              clk : in std_logic);
```

```
end component;
component reg_mac
port( D : in element;
    Q : out element;
                clk,sclr : in std_logic);
end component;

component reg_32
 port( D : in element;
      Q : out element;
                clk : in std_logic);
end component;

component bus_mux
 port ( MA,MB : in element;
       S : in std_logic;
                        O : out element);
end component;

signal RM1,RM2 : donnee;
signal MA : element;
signal AR,RA : element;
signal MR : element;

begin

U1 : reg_16 port map(D=>A,Q=>RM1,clk=>clk);
U2 : reg_16 port map(D=>B,Q=>RM2,clk=>clk);
U3 : multi_16 port map(A=>RM1,B=>RM2,O=>MA);
U4 : mac_adder port map(A=>MA,B=>RA,S=>AR);
U5 : reg_mac port map(D=>AR,Q=>RA,clk=>clk,sclr=>reset);
U6 : reg_32 port map (D=>MR,Q=>mac_out,clk=>clk);
U7 : bus_mux port map (MA=>mac_in,MB=>AR,S=>reset,O=>MR);

end Behavioral;
```

8- Controler :

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
use Work.My_type.all;

entity controleur is
   Port ( clk : in std_logic;
        start : in std_logic;
        cmz : in std_logic;
        cm0 : out std_logic;
        cm1 : out std_logic;
        cm2 : out std_logic);
end controleur;
```

```
architecture Behavioral of controleur is

component bascule_d_con
 port(clk : in std_logic;
     D : in std_logic;
                Q : out std_logic);
end component;

signal int : std_logic;
signal cm0_int : std_logic;

begin

 process(clk)

 variable i : natural range 0 to 255 := 1;
 variable j : natural range 0 to 1 := 0;
 variable k : natural range 0 to 255 := 1;

 begin

  if(clk'event and clk='1') then

    if start='1' then

          j:=1;

         end if;

         if j=1 then

          if cmz='1' then

            k:=k+1;

          end if;

          if i < Matrix_Size then

            cm0_int <= '0';
                i:=i+1;

          else

            cm0_int <= '1';
                i:=1;
                j:=0;

         end if;
```

```
        else

          cm0_int <= '0';

        end if;

  if (k-i+1>0 and j=1) then

          int <= '1';

        else

          int <= '0';
                k:=1;

        end if;

 end if;

end process;


cm1 <= int;
cm2 <= not(cmz) and int;
S1 : bascule_d_con port map(clk=>clk,D=>cm0_int,Q=>cm0);


end Behavioral;
```

9- Resol_syst :

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
use Work.My_type.all;

--This Bloc it's OK 100%

entity resol_system is
   Port ( D : in mat_lin_in;
              clk : in std_logic;
                        start : in std_logic;
                        R : out element);
end resol_system;

architecture Behavioral of resol_system is

 component adapter
   Port ( Xin : in mat_lin_in;
              Xout : out mat_lin;
                        clk : in std_logic;
                        start : in std_logic;
```

```
                        start_out : out std_logic);
end component;

 component Principale
   Port ( X : in mat_lin;
                        R : out element;
                        start : in std_logic;
                        clk : in std_logic
              );
 end component;

  signal S : mat_lin;
  signal start_int : std_logic;

begin



 U2 : adapter port map(Xin=>D,clk=>clk,Xout=>S,start=>start,start_out=>start_int);
 U3 : principale port map(X=>S,R=>R,clk=>clk,start=>start_int);



end Behavioral;
```

10- Adapter :

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
use Work.My_type.all;

-- pour la sortie Xout(N+1) il suffit de metrre une serie de
-- bascule pour le start au lieu des regitres et fair " & & "

entity adapter is
   Port ( Xin : in mat_lin_in;
              Xout : out mat_lin;
                        clk : in std_logic;
                        start : in std_logic;
                        start_out : out std_logic
                           );
end adapter;

architecture Behavioral of adapter is

 component reg_32
  port ( D : in element;
              Q : out element;
                        clk : in std_logic);
 end component;

 component bascule_d
```

```
 port ( D : in std_logic;
        Q : out std_logic;
        clk : in std_logic);
end component;

signal R : mas_mas_lin;
signal L : mat_lin;
signal F : std_logic_vector(1 to 2*Matrix_Size);
signal T : mat_lin_in;

begin

F(1) <= start;

Xout(1) <= L(1);

L(Matrix_Size+1) <= '0' & X"000" & F(2*Matrix_Size)& "00" & X"0000";

start_out <= F(1);

M1 : for J in 1 to Matrix_Size Generate

S1 :    if J=1 Generate

R1 : reg_32 port map (D=>Xin(1),Q=>L(1),clk=>clk);

         end Generate;


S3 : if J >= 2 Generate

         M2 : for I in 1 to J Generate

   S4 : if I=1 Generate

            R2 : reg_32 port map (D=>Xin(J),Q=>R(((J-2)*(J-1)/2)+I),clk=>clk);

               end Generate;

            S5 : if (I >= 2 and I < J) Generate

            R3 : reg_32 port map (D=>R(((J-2)*(J-1)/2)+I-1),Q=>R(((J-2)*(J-1)/2)+I),clk=>clk);


               end Generate;

            S6 : if I=J Generate
```

```
        R4 : reg_32 port map (D=>R(((J-2)*(J-1)/2)+I-1),Q=>L(J),clk=>clk);

            end Generate;

    end Generate;

  end Generate;

end Generate;

U1 : for K in 1 to (2*Matrix_Size-1) Generate

        R5 : bascule_d port map (D=>F(K),Q=>F(K+1),clk=>clk);

end Generate;

  U3 : for n in 1 to Matrix_Size Generate

        R7 : T(n) <= L(n+1);

end Generate;

 U2 : for h in 1 to Matrix_Size Generate

        R6 : reg_32 port map (D=>T(h),Q=>Xout(h+1),clk=>clk);

end Generate;


end Behavioral;
```

11- Principale:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
use Work.My_type.all;

entity Principale is
   Port ( X : in mat_lin;
                        R : out element;
                        start : in std_logic;
                        clk : in std_logic
              );
end Principale;

architecture Behavioral of Principale is
-----------------------------------------
 component Multi_clk
  port( CLK_IN   : in std_logic;
        RST      : in std_logic;
        CLK1X    : out std_logic;
```

```
                CLKFX    : out std_logic;
                CLKFX180    : out std_logic;
      LOCKED     : out std_logic;
                PSDONE   : out std_logic);
end component;

component Ronds_32
 port(    Din : in element;
        Dout : out element;
        clk1x : in std_logic;
                        clkfx : in std_logic;
        start : in std_logic;
        start_out : out std_logic;
        cmd0 : out std_logic;
        cmd1 : out std_logic;
        cmd2 : out std_logic);

 end component;


 component MAS
  port( Xin : in element;
             Yin : in element;
                        clk : in std_logic;
                        cm0_in : in std_logic;
                        cm1_in : in std_logic;
                        cm2_in : in std_logic;
                        Xout : out element;
                        Yout : out element;
                        cm0_out : out std_logic;
                        cm1_out : out std_logic;
                        cm2_out : out std_logic);
 end component;

component MAS_P
  port( Xin : in element;
             Yin : in element;
                        clk : in std_logic;
                        cm0_in : in std_logic;
                        cm1_in : in std_logic;
                        cm2_in : in std_logic;
                        Xout : out element;
                        Yout : out element;
                        cm0_out : out std_logic;
                        cm1_out : out std_logic;
                        cm2_out : out std_logic);
 end component;
```

```
 component MAS_F
  port( Xin : in element;
        Yin : in element;
              clk : in std_logic;
              cm0_in : in std_logic;
              cm1_in : in std_logic;
              cm2_in : in std_logic;
              Xout : out element);
 end component;
---------------------------------------
 signal Clk1X : std_logic;
 signal ClkFX : std_logic;
 signal GND,clkfx_180,LOCK,PSDONE : std_logic;


 signal MAS_y : mas_mas_lin;
 signal MAS_x : mas_mas_lin;
 signal MAS_cm0 : mas_mas_cmd;
 signal MAS_cm1 : mas_mas_cmd;
 signal MAS_cm2 : mas_mas_cmd;
 signal MAS_start : mas_mas_start;

begin

 GND <= '0';

 U1: Multi_clk
  port map (
       CLK_IN =>   clk,
                   CLK1X => Clk1X,
                   RST =>    GND,
                   CLKFX =>   ClkFX,
                   CLKFX180 => clkfx_180,
       LOCKED =>  LOCK,
                   PSDONE =>  PSDONE);


 matrix_lin : for I in 1 to Matrix_Size Generate
  matrix_col : for J in 1 to (Matrix_Size+2-I) Generate

        S1 : if I=1 Generate
             S2 : if J=1 Generate

                  S3 : Ronds_32 port map ( clk1x=>Clk1X,clkfx=>ClkFX,start=>start,
                  Din=>X(1),start_out=>MAS_start(1),Dout=>MAS_y(1),

cmd0=>MAS_cm0(1),cmd1=>MAS_cm1(1),cmd2=>MAS_cm2(1));

                                end Generate;
                        S20 : if J=2 Generate
```

```
                    S21 : MAS_P port map(clk=>Clk1X,Xin=>X(J),Yin=>MAS_y(J-1),
                            Yout=>MAS_y(J),Xout=>MAS_x(J-
                                1),cm0_in=>MAS_cm0(J-1),

                cm1_in=>MAS_cm1(J-1),cm2_in=>MAS_cm2(J-
                                1),cm0_out=>MAS_cm0(J),

cm1_out=>MAS_cm1(J),cm2_out=>MAS_cm2(J));

                                                    end Generate;


                    S4 : if (J>2 and J<= Matrix_Size) Generate

                    S5 : MAS port map(clk=>Clk1X,Xin=>X(J),Yin=>MAS_y(J-1),
                            Yout=>MAS_y(J),Xout=>MAS_x(J-
                                1),cm0_in=>MAS_cm0(J-1),
                cm1_in=>MAS_cm1(J-1),cm2_in=>MAS_cm2(J-
                                1),cm0_out=>MAS_cm0(J),

cm1_out=>MAS_cm1(J),cm2_out=>MAS_cm2(J));

                                                    end Generate;

                    S6 : if J=Matrix_Size+1 Generate

                    S7 : MAS_F port
map(clk=>Clk1X,Xin=>X(Matrix_Size+1),Yin=>MAS_y(Matrix_Size),

Xout=>MAS_x(Matrix_Size),cm0_in=>MAS_cm0(Matrix_Size),
                cm1_in=>MAS_cm1(Matrix_Size),cm2_in=>MAS_cm2(Matrix_Size));

                                                    end Generate;
                            end Generate;

            S8 : if (I >= 2 and I < (Matrix_Size)) Generate
                    S9 : if J=1 Generate

                    S10 : Ronds_32 port map (
clk1x=>Clk1X,clkfx=>ClkFX,start=>MAS_start(I-1),
                    Din=>MAS_x((Matrix_Size*(Matrix_Size+1)/2)-((Matrix_Size+2-
                            I)*(Matrix_Size+3-I)/2)+1),start_out=>MAS_start(I),

Dout=>MAS_y((Matrix_Size*(Matrix_Size+1)/2)-((Matrix_Size+1-I)*(Matrix_Size+2-I)/2)+1),

cmd0=>MAS_cm0((Matrix_Size*(Matrix_Size+1)/2)-((Matrix_Size+1-I)*(Matrix_Size+2-
                            I)/2)+1),

cmd1=>MAS_cm1((Matrix_Size*(Matrix_Size+1)/2)-((Matrix_Size+1-I)*(Matrix_Size+2-
                            I)/2)+1),

cmd2=>MAS_cm2((Matrix_Size*(Matrix_Size+1)/2)-((Matrix_Size+1-I)*(Matrix_Size+2-
                            I)/2)+1));

                                    end Generate;
```

```
                    S22: if J=2 Generate

        S23 : MAS_P port map(clk=>Clk1X,Xin=>MAS_X((Matrix_Size*(Matrix_Size+1)/2)-
((Matrix_Size+2-I)*(Matrix_Size+3-I)/2)+J),


Yin=>MAS_y((Matrix_Size*(Matrix_Size+1)/2)-((Matrix_Size+1-I)*(Matrix_Size+2-I)/2)+J-1),
                                        Yout=>MAS_y((Matrix_Size*(Matrix_Size+1)/2)-
((Matrix_Size+1-I)*(Matrix_Size+2-I)/2)+J),

Xout=>MAS_x((Matrix_Size*(Matrix_Size+1)/2)-((Matrix_Size+1-I)*(Matrix_Size+2-I)/2)+J-1),

cm0_in=>MAS_cm0((Matrix_Size*(Matrix_Size+1)/2)-((Matrix_Size+1-I)*(Matrix_Size+2-
I)/2)+J-1),
                    cm1_in=>MAS_cm1((Matrix_Size*(Matrix_Size+1)/2)-((Matrix_Size+1-
I)*(Matrix_Size+2-I)/2)+J-1),

cm2_in=>MAS_cm2((Matrix_Size*(Matrix_Size+1)/2)-((Matrix_Size+1-I)*(Matrix_Size+2-
I)/2)+J-1),

cm0_out=>MAS_cm0((Matrix_Size*(Matrix_Size+1)/2)-((Matrix_Size+1-I)*(Matrix_Size+2-
I)/2)+J),

cm1_out=>MAS_cm1((Matrix_Size*(Matrix_Size+1)/2)-((Matrix_Size+1-I)*(Matrix_Size+2-
I)/2)+J),

cm2_out=>MAS_cm2((Matrix_Size*(Matrix_Size+1)/2)-((Matrix_Size+1-I)*(Matrix_Size+2-
I)/2)+J));

                                                    end Generate;

                    S11 : if J>2 and J<=(Matrix_Size+1-I) Generate

                        S12 : MAS port
map(clk=>Clk1X,Xin=>MAS_X((Matrix_Size*(Matrix_Size+1)/2)-((Matrix_Size+2-
I)*(Matrix_Size+3-I)/2)+J),
                                        Yin=>MAS_y((Matrix_Size*(Matrix_Size+1)/2)-
((Matrix_Size+1-I)*(Matrix_Size+2-I)/2)+J-1),
                                        Yout=>MAS_y((Matrix_Size*(Matrix_Size+1)/2)-
((Matrix_Size+1-I)*(Matrix_Size+2-I)/2)+J),

Xout=>MAS_x((Matrix_Size*(Matrix_Size+1)/2)-((Matrix_Size+1-I)*(Matrix_Size+2-I)/2)+J-1),

cm0_in=>MAS_cm0((Matrix_Size*(Matrix_Size+1)/2)-((Matrix_Size+1-I)*(Matrix_Size+2-
I)/2)+J-1),
                    cm1_in=>MAS_cm1((Matrix_Size*(Matrix_Size+1)/2)-((Matrix_Size+1-
I)*(Matrix_Size+2-I)/2)+J-1),

cm2_in=>MAS_cm2((Matrix_Size*(Matrix_Size+1)/2)-((Matrix_Size+1-I)*(Matrix_Size+2-
I)/2)+J-1),

cm0_out=>MAS_cm0((Matrix_Size*(Matrix_Size+1)/2)-((Matrix_Size+1-I)*(Matrix_Size+2-
I)/2)+J),
```

```
cm1_out=>MAS_cm1((Matrix_Size*(Matrix_Size+1)/2)-((Matrix_Size+1-I)*(Matrix_Size+2-
I)/2)+J),

cm2_out=>MAS_cm2((Matrix_Size*(Matrix_Size+1)/2)-((Matrix_Size+1-I)*(Matrix_Size+2-
I)/2)+J));

                                                    end Generate;

                                S13 : if J=(Matrix_Size+2-I) Generate

                            S14 : MAS_F port
map(clk=>Clk1X,Xin=>MAS_x((Matrix_Size*(Matrix_Size+1)/2)-((Matrix_Size+2-
I)*(Matrix_Size+3-I)/2)+J),
                                                Yin=>MAS_y((Matrix_Size*(Matrix_Size+1)/2)-
((Matrix_Size+1-I)*(Matrix_Size+2-I)/2)+J-1),
                                                 Xout=>MAS_x((Matrix_Size*(Matrix_Size+1)/2)-
((Matrix_Size+1-I)*(Matrix_Size+2-I)/2)+J-1),

cm0_in=>MAS_cm0((Matrix_Size*(Matrix_Size+1)/2)-((Matrix_Size+1-I)*(Matrix_Size+2-
I)/2)+J-1),
                        cm1_in=>MAS_cm1((Matrix_Size*(Matrix_Size+1)/2)-((Matrix_Size+1-
I)*(Matrix_Size+2-I)/2)+J-1),

cm2_in=>MAS_cm2((Matrix_Size*(Matrix_Size+1)/2)-((Matrix_Size+1-I)*(Matrix_Size+2-
I)/2)+J-1));

                                                    end Generate;
                                end Generate;
                            S15 : if I=Matrix_Size Generate
                        S16 : if J=1 Generate

                            S17 : Ronds_32 port map (
clk1x=>Clk1X,clkfx=>ClkFX,start=>MAS_start(I-1),
                            Din=>MAS_x((Matrix_Size*(Matrix_Size+1)/2)-
2),start_out=>MAS_start(I),

Dout=>MAS_y((Matrix_Size*(Matrix_Size+1)/2)),

cmd0=>MAS_cm0((Matrix_Size*(Matrix_Size+1)/2)),

cmd1=>MAS_cm1((Matrix_Size*(Matrix_Size+1)/2)),

cmd2=>MAS_cm2((Matrix_Size*(Matrix_Size+1)/2)));

                                    end Generate;

                                S18 : if J=2 Generate

                            S19 : MAS_F port
map(clk=>Clk1X,Xin=>MAS_x((Matrix_Size*(Matrix_Size+1)/2)-1),
                                                Yin=>MAS_y((Matrix_Size*(Matrix_Size+1)/2)),
                                                 Xout=>MAS_x((Matrix_Size*(Matrix_Size+1)/2)),

cm0_in=>MAS_cm0((Matrix_Size*(Matrix_Size+1)/2)),
```

```
                    cm1_in=>MAS_cm1((Matrix_Size*(Matrix_Size+1)/2)),
cm2_in=>MAS_cm2((Matrix_Size*(Matrix_Size+1)/2)));
                                                        end Generate;
                                end Generate;
            end Generate;
  end Generate;

        R <= MAS_x(Matrix_Size*(Matrix_Size+1)/2);


end Behavioral;
```

12- MAS:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
use Work.My_type.all;

entity mas is
   Port (  Xin : in element;
               Yin : in element;
                           clk : in std_logic;
                           cm0_in : in std_logic;
                           cm1_in : in std_logic;
                           cm2_in : in std_logic;
                           Xout : out element;
                           Yout : out element;
                           cm0_out : out std_logic;
                           cm1_out : out std_logic;
                           cm2_out : out std_logic);
end mas;

architecture Behavioral of mas is

 component reg_32
  port( D : in element;
       clk : in std_logic;
                  Q : out element);
 end component;

 component reg_32_ce
   port ( D : in element;
              clk : in std_logic;
                          ce : in std_logic;
                          Q : out element);
  end component;

  component multi
   port( A : in element;
```

```
            B : in element;
                    O : out result);
 end component;

 component mux1
  port( A : in element;
            B : in element;
                    C : out element;
                    cmd : in std_logic);
       end component;

        component mux2
         port( A : in element;
                  B : in element;
                          C : in element;
                          D : out element;
                          cmd1 : in std_logic;
                          cmd2 : in std_logic);
        end component;

        component soustra
         port( A : in element;
                  B : in element;
                          S : out element);
              end component;

              component bascule_d
    port( D : in std_logic;
                  clk : in std_logic;
                          Q : out std_logic);
              end component;

signal x1,x2,y1,m1,m2,X_out1,X_out2 : element;
signal s0,p1,p2 : element;
signal p : result;
signal clk1 : std_logic;


begin

U1 : reg_32 port map (D=>Xin,Q=>x1,clk=>clk);
U2 : reg_32 port map (D=>Yin,Q=>y1,clk=>clk);
U3 : reg_32_ce port map (D=>p1,Q=>p2,clk=>clk,ce=>cm2_in);
U4 : mux1 port map(A=>p2,B=>Yin,C=>m2,cmd=>cm2_in);
U5 : mux2 port map(A=>x2,B=>s0,C=>p2,D=>X_out1,cmd1=>cm1_in,cmd2=>cm0_in);
```

```
U6 : mux1 port map(A=>Yin,B=>Xin,C=>m1,cmd=>cm2_in);
U7 : multi port map(A=>m1,B=>m2,O=>p);
U8 : soustra port map(A=>xin,B=>p1,S=>s0);
U9 : reg_32 port map(D=>x1,Q=>x2,clk=>clk);
U10: bascule_d port map(D=>cm0_in,Q=>cm0_out,clk=>clk);
U11: bascule_d port map(D=>cm1_in,Q=>cm1_out,clk=>clk);
U12: bascule_d port map(D=>cm2_in,Q=>cm2_out,clk=>clk);
U13 : reg_32 port map (D=>X_out1,Q=>X_out2,clk=>clk);
U14 : reg_32 port map (D=>X_out2,Q=>Xout,clk=>clk);

Yout <= y1;
p1 <= p(63) & p(48 downto 18);
clk1 <=  not(clk);


end Behavioral;
```

13- Ronds_32 :

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
use Work.My_type.all;

entity Ronds_32 is
   Port ( Din : in std_logic_vector(31 downto 0);
        Dout : out std_logic_vector(31 downto 0);
                        clkfx : in std_logic;
        clk1x : in std_logic;
        start : in std_logic;
                        start_out : out std_logic;
        cmd0 : out std_logic;
        cmd1 : out std_logic;
        cmd2 : out std_logic);
end Ronds_32;

architecture Behavioral of Ronds_32 is

component inversor
port( dividend : in std_logic_vector (18 downto 0);
     divisor : in std_logic_vector (30 downto 0);
                C : in std_logic;
                quot : out std_logic_vector (18 downto 0);
                remd : out std_logic_vector (17 downto 0));
end component;

component bascule_D
 port( D : in std_logic;
      clk : in std_logic;
                 Q : out std_logic);
end component;
```

```
component controleur
 port ( clk : in std_logic;
       start : in std_logic;
                        cmz : in std_logic;
                        cm0 : out std_logic;
                        cm1 : out std_logic;
                        cm2 : out std_logic);
end component;

component reg_32
 port( D : in std_logic_vector(31 downto 0);
      Q : out std_logic_vector(31 downto 0);
                clk : in std_logic);
 end component;

 component comp2
 port(A : in std_logic_vector(30 downto 0);
     BYPASS : in std_logic;
               clk : in std_logic;
               Q : out std_logic_vector(31 downto 0));
 end component;


--
 signal un : std_logic_vector (18 downto 0);
 signal quot1 :std_logic_vector (18 downto 0);
 signal remd1 :std_logic_vector (17 downto 0);
 signal div : std_logic_vector (30 downto 0);
 signal S,S1 : std_logic_vector (31 downto 0);
 signal S0,Gdiv : std_logic_vector(31 downto 0);
--
signal start_cont,start_fin,start_afin : std_logic;
signal cmd_zero,sig : std_logic;

--

signal D_int : std_logic_vector(31 downto 0);
signal D_test : std_logic_vector(30 downto 0);

begin
```

```
un <= "1000000000000000000";

div <= Gdiv(30 downto 0);

D_test <= D_int(30 downto 0);

S1 <= Din(31) & quot1(12 downto 0) & remd1;

S <= S0;

Z1 : comp2 port map(A=>D_int(30 downto 0),Q=>Gdiv,clk=>clkfx,BYPASS=>D_int(31));
Z2 : comp2 port map(A=>S1(30 downto 0),Q=>S0,clk=>clkfx,BYPASS=>D_int(31));


U1: inversor port map(dividend=>un, divisor=>div,C=>clkfx,quot=>quot1,remd=>remd1);
U3 : bascule_D port map (clk=>clk1x,D=>start,Q=>start_cont);
U4 : bascule_D port map (clk=>clk1x,D=>start_cont,Q=>start_fin);
U7 :bascule_D port map (clk=>clk1x,D=>D_int(31),Q=>sig);
U5 : controleur port map
(clk=>clk1x,cmz=>cmd_zero,start=>start_cont,cm0=>cmd0,cm1=>cmd1,cm2=>cmd2);
U6 : reg_32 port map(clk=>clk1x,D=>Din,Q=>D_int);
U8 : bascule_D port map (clk=>clk1x,D=>start_fin,Q=>start_afin);
U9 : bascule_D port map (clk=>clk1x,D=>start_afin,Q=>start_out);

process(clk1x,start)

variable i : natural range 0 to 1 :=0;
variable j : natural range 0 to 1 :=0;
variable k : natural range 0 to 255 :=0;

begin

if (clk1x'event and clk1x='1') then


  if k > Matrix_Size then

        i:=0;
              j:=0;
              k:=0;

      end if;
```

```
if start='1' then

 j:=1;
      if i=0 then

       if D_test=0 then

        Dout <= X"00000000";
             cmd_zero <= '1';

       else

        Dout <= S;
             cmd_zero <= '0';
             i:=1;

       end if;

      else

       Dout <= D_int;
       cmd_zero <= '0';

      end if;

      k:=k+1;

else

 if j=1 then

       if i=0 then

        if D_test=0 then

             Dout <= X"00000000";
                  cmd_zero <= '1';

            else

             Dout <= S;
                  cmd_zero <= '0';
                  i:=1;

            end if;
```

```
                else

                  Dout <= D_int;
                        cmd_zero <= '0';

                end if;

                k:=k+1;

              end if;

          end if;

 end if;

end process;



end Behavioral;
```

14- DCM:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

library UNISIM;
use UNISIM.VCOMPONENTS.ALL;

entity multi_clk is
    port (
       CLK_IN   : in std_logic;
       RST      : in std_logic;
--                PSEN     : in std_logic;
--                CLK180 : out std_logic;
       CLK1X    : out std_logic;
                  CLKFX    : out std_logic;
                  CLKFX180    : out std_logic;
       LOCKED     : out std_logic;
                  PSDONE   : out std_logic
      );
end multi_clk;

architecture Behavioral of multi_clk is
-- Components Declarations:
--
component BUFG
  port (
         I   : in std_logic;
       O    : out std_logic
         );
```

```
end component;
--
component DCM
-- synopsys translate_off
   generic (
         DLL_FREQUENCY_MODE : string := "LOW";
         DUTY_CYCLE_CORRECTION : boolean := TRUE;
                              CLKOUT_PHASE_SHIFT : string := "FIXED";
                              PHASE_SHIFT  : integer := 0;
                              CLKFX_MULTIPLY : integer := 60 ;
         CLKFX_DIVIDE : integer := 1
         --STARTUP_WAIT : boolean := FALSE
        );
-- synopsys translate_on

   port ( CLKIN    : in  std_logic;
        CLKFB    : in  std_logic;
        DSSEN    : in  std_logic;
        PSINCDEC  : in  std_logic;
        PSEN     : in  std_logic;
        PSCLK    : in  std_logic;
        RST      : in  std_logic;
        CLK0     : out std_logic;
        CLK90    : out std_logic;
        CLK180   : out std_logic;
        CLK270   : out std_logic;
        CLK2X    : out std_logic;
        CLK2X180  : out std_logic;
        CLKDV    : out std_logic;
        CLKFX    : out std_logic;
        CLKFX180  : out std_logic;
        LOCKED   : out std_logic;
        PSDONE   : out std_logic;
        STATUS   : out std_logic_vector(7 downto 0)
       );
end component;
--
-- Attributes

attribute DLL_FREQUENCY_MODE : string;
attribute DUTY_CYCLE_CORRECTION : string;
attribute CLKOUT_PHASE_SHIFT : string;
attribute PHASE_SHIFT  : integer;
attribute CLKFX_MULTIPLY : integer;
attribute CLKFX_DIVIDE : integer;
--attribute STARTUP_WAIT : string;
```

```
attribute DLL_FREQUENCY_MODE of U_DCM: label is "LOW";
attribute DUTY_CYCLE_CORRECTION of U_DCM: label is "TRUE";
attribute CLKOUT_PHASE_SHIFT of U_DCM: label is "FIXED";
attribute PHASE_SHIFT  of U_DCM: label is 0;
attribute CLKFX_MULTIPLY of U_DCM: label is 60;
attribute CLKFX_DIVIDE of U_DCM: label is 1;
--attribute STARTUP_WAIT of U_DCM: label is "FALSE";

--
-- Signal Declarations:
signal GND : std_logic;
--
signal CLK0_W: std_logic;
signal CLK1X_W: std_logic;
signal CLKFX_W: std_logic;
signal CLKF_W: std_logic;
signal CLKFX180_W: std_logic;
signal CLKF180_W: std_logic;

begin
GND <= '0';
--
CLK1X <= CLK1X_W;
CLKFX <= CLKF_W;
CLKFX180 <= CLKF180_W;
--

-- DCM Instantiation for the FIXED mode.Note that the PSCLK,PSEN,PSINCDEC signals
-- are tied to Ground. The PSEN,PSINCDEC,PSDONE signals have to be removed from the
-- entity declaration and port list.
        U_DCM: DCM
  port map (
                        CLKIN =>    CLK_IN,
                                        CLKFB =>    CLK1X_W,
                                        DSSEN =>    GND,
                                        PSCLK =>    GND,
                        PSEN  =>    GND,
                                        PSINCDEC => GND,
--                                      CLK180 <= CLK180,
                                    RST   =>    RST,
                                        CLK0  =>    CLK0_W,
              CLKFX =>    CLKFX_W,
                                CLKFX180 => CLKFX180_W,
                        LOCKED =>   LOCKED
                                    );
```

```
-- BUFG Instantiation for CLK0
U0_BUFG: BUFG
  port map (
            I => CLK0_W,
          O => CLK1X_W
            );

-- BUFG Instantiation for CLKFX
U1_BUFG: BUFG
  port map (
            I => CLKFX_W,
          O => CLKF_W
            );
--

-- BUFG Instantiation for CLKFX180
U2_BUFG: BUFG
  port map (
            I => CLKFX180_W,
          O => CLKF180_W
            );
--

end Behavioral;
```

15- Controleur:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
use Work.My_type.all;

entity controleur is
    Port ( clk : in std_logic;
           start : in std_logic;
           cmz : in std_logic;
           cm0 : out std_logic;
           cm1 : out std_logic;
           cm2 : out std_logic);
end controleur;

architecture Behavioral of controleur is

component bascule_d_con
 port(clk : in std_logic;
      D : in std_logic;
                  Q : out std_logic);
end component;
```

```
signal int : std_logic;
signal cm0_int : std_logic;

begin

 process(clk)

 variable i : natural range 0 to 255 := 1;
 variable j : natural range 0 to 1 := 0;
 variable k : natural range 0 to 255 := 1;

 begin

  if(clk'event and clk='1') then

    if start='1' then

            j:=1;

          end if;

          if j=1 then

            if cmz='1' then

              k:=k+1;

            end if;

            if i < Matrix_Size then

              cm0_int <= '0';
                    i:=i+1;

            else

              cm0_int <= '1';
                    i:=1;
                    j:=0;

            end if;

          else

            cm0_int <= '0';

          end if;
```

```
   if (k-i+1>0 and j=1) then

            int <= '1';

          else

            int <= '0';
                  k:=1;

          end if;

 end if;

end process;


cm1 <= int;
cm2 <= not(cmz) and int;
S1 : bascule_d_con port map(clk=>clk,D=>cm0_int,Q=>cm0);


end Behavioral;
```

Annexe 4

BLIND SYSTEM IDENTIFICATION USING CROSS-RELATION METHODS: FURTHER RESULTS AND DEVELOPMENTS

A. Aïssa-El-Bey, M. Grebici*, K. Abed-Meraim** and A. Belouchrani**

* EEE Dept., Ecole Nationale Polytechnique (El-Harrach), Algiers, Algeria.
** Signal & Image Proc. Dept., Telecom Paris, 46 rue Barrault, 75013 Paris, France.

ABSTRACT

We consider the problem of blind identification of FIR systems using the cross-relations (CR) method first introduced in [1]. Our contribution in this paper are as follows: (i) We introduce an extended formulation of the CR identification criterion which generalizes the standard CR criterion used in [1]. It can be shown that many existing multichannel blind identification methods belong to the class of generalized CR methods. (ii) We introduce a new identification method referred to as Minimum Cross-Relations (MCR) method which exploits with minimum redundancy the spatial diversity among the channel outputs. Simulation-based performance analysis of the MCR method and comparisons with CR method are also presented. (iii) Then, we present a modified version of the MCR referred to as the "unbiased MCR" (UMCR) method that leads to unbiased estimation of the channel parameters and better estimation performances without need of noise whitening as in the MCR. (iv) Finally, we discuss the multi-input case and show how additional difficulties arise due to the non-linear parameterization of the noise vectors in terms of the channel parameters.

1. INTRODUCTION

Blind system identification (BSI) is a fundamental signal processing technology aimed at retrieving a system's unknown information from its outputs only. This problem has received a lot of attention in the signal processing literature and a plethora of methods and techniques have been proposed to solve the BSI over the last 2 decades [6, 7]. Since 1991, it has been shown that using spatial and/or temporal diversity leads to efficient and simplified BSI methods using only the second order statistics of the outputs or even deterministic approaches. The CR method introduced in [1] is one of the simplest and efficient methods for blind identification of FIR SIMO systems. This paper focuses on the CR method and introduces several improvements and new developments related to this technique. At first, we reformulate the CR problem in such a way to provide a general framework where a large class of BSI methods can be seen as CR-like methods. Then we introduce several improvements /simplifications of the original CR method referred to as MCR (for Minimum Cross-Relations) and UMCR (for Unbiased Minimum Cross-Relations) method. Finally, we discuss the MIMO case and explain why the CR method cannot be extended "in a simple way" to solve the MIMO-BSI problem.

2. PROBLEM FORMULATION

Consider a SIMO system of q outputs given by:

$$\mathbf{y}(n) = \sum_{k=0}^{M} \mathbf{h}(k)s(n-k) + \mathbf{w}(n) \qquad (1)$$

where $\mathbf{h}(z) = \sum_{k=0}^{M} \mathbf{h}(k)z^{-k}$ is an unknown causal FIR $q \times 1$ transfer function satisfying $\mathbf{h}(z) \neq 0$, $\forall z$, $s(n)$ a scalar (non-observable) stationary process and $\mathbf{w}(n)$ is an additive q-dimensional spatial white noise, i.e, $E[\mathbf{w}(n)\mathbf{w}^H(n)] = \sigma^2\mathbf{I}_q$.

Given a finite set of observation vectors $\mathbf{y}(1), \ldots, \mathbf{y}(T)$ and based on the channel entries co-primness (i.e. $\mathbf{h}(z) \neq 0\ \forall z$), the objective here is to estimate the channel coefficients vector $\mathbf{h} = [\mathbf{h}(0)^T, \cdots, \mathbf{h}(M)^T]^T$ up to a scalar constant (this is an inherent indeterminacy of the BSI problem as shown in [5]) using CR-like techniques. Before proceeding, we review next the basic idea and principle of the original CR method in [5].

3. CR-LIKE METHODS

This section is devoted to the development of the MCR and UMCR methods as well as the generalized formulation of the CR criterion. For that, we start first by a brief review of the CR principle.

3.1. Review of the CR method

From (1), the noise-free outputs $y_i(k)$, $1 \leq i \leq q$ are given by:

$$y_i(k) = h_i(k) * s(k), \quad 1 \leq i \leq q \qquad (2)$$

where "$*$" denotes convolution. Using commutativity of convolution, it follows:

$$h_j(k) * y_i(k) = h_i(k) * y_j(k), \quad 1 \leq i < j \leq q \qquad (3)$$

This is a linear equation satisfied by every pairs of channels.

It was shown that based on $q(q-1)/2$ possible cross-relations, the channel parameters can be uniquely identified according to [5]:

Theorem 1 *Under the data model assumptions, the set of cross-relations (in the noise free case):*

$$y_i(k) * h'_j(k) - y_j(k) * h'_i(k) = 0, \quad 1 \leq i < j \leq q \qquad (4)$$

where $\mathbf{h}'(z)$ is a $q \times 1$ polynomial vector of degree M, is satisfied if and only if $\mathbf{h}'(z) = \alpha\mathbf{h}(z)$ for a given scalar constant α.

By collecting all possible pairs of q channels, one can easily establish a set of linear equations. In matrix form, this set of equations can be expressed as:

$$\mathbf{Y}_q \mathbf{h} = 0 \tag{5}$$

where $\mathbf{Y}_q$ is defined by:

$$\mathbf{Y}_2 = [\mathbf{Y}_{(2)}, -\mathbf{Y}_{(1)}]$$

$$\mathbf{Y}_l = \left[\begin{array}{ccc|c} & \mathbf{Y}_{l-1} & & \mathbf{0} \\ \hline \mathbf{Y}_{(l)} & & \mathbf{0} & -\mathbf{Y}_{(1)} \\ & \ddots & & \vdots \\ \mathbf{0} & & \mathbf{Y}_{(l)} & -\mathbf{Y}_{(l-1)} \end{array}\right] \tag{6}$$

with $l = 3, \ldots, q$ and:

$$\mathbf{Y}_{(l)} = \begin{bmatrix} y_l(M) & \ldots & y_l(0) \\ \vdots & & \vdots \\ y_l(N-1) & \ldots & y_l(N-M-1) \end{bmatrix} \tag{7}$$

In the presence of noise, equation (5) can be naturally solved in the least-square (LS) sense according to:

$$\hat{\mathbf{h}}_{CR} = \arg \min_{\|\mathbf{h}\|=1} \mathbf{h}^H \mathbf{Y}_q^H \mathbf{Y}_q \mathbf{h} \tag{8}$$

The CR method is referred to as the LS method in [5] because it represents the least-squares solution to the CR equation (5).

3.2. Minimum CR Method

In the same spirit as in the MNS (Minimum Noise Subspace) method [4], we show here that only $q-1$ (instead of $q(q-1)/2$) cross-relations can be used for channel identification. We have the following theorem:

Theorem 2 *Let $\{p_1, \ldots, p_{q-1}\}$, $p_i = (i_1, i_2)$, $i = 1, \ldots, q-1$, be a set of $q-1$ pairs of channels which form a tree structure, then the noise-free cross-relations ($\mathrm{h}'(z)$ being a polynomial vector of degree M):*

$$h'_{i_1}(k) * y_{i_2}(k) - h'_{i_2}(k) * y_{i_1}(k) = 0, \quad i = 1, \ldots, q-1 \tag{9}$$

yield a unique identification of the channel transfer function $\mathrm{h}(z)$, i.e. $\mathrm{h}'(z) = \alpha \mathrm{h}(z)$ for a given scalar constant α.

In figure 1, we consider the following example corresponding to $q = 5$ and $p_1 = (1,2)$, $p_2 = (1,3)$, $p_3 = (3,4)$ and $p_4 = (3,5)$ that is a set of four (i.e. : $q-1$) pairs forming a tree structure. To solve (9) in presence of noise, we estimate the channel parameters in the least squares sense according to:

$$\hat{\mathbf{h}}_{MCR} = \arg \min_{\|\mathbf{h}\|=1} \mathbf{h}^H \overline{\mathbf{Y}}^H \overline{\mathbf{Y}} \mathbf{h} \tag{10}$$

where $\overline{\mathbf{Y}}$ is a block matrix which (k,l) block entry is zero if $l \notin \{k_1, k_2\}$ with $p_k = (k_1, k_2)$ and is equal to $\mathbf{Y}_{(k_2)}$ if $l = k_1$ and $-\mathbf{Y}_{(k_1)}$ if $l = k_2$. Contrary to the CR method, the MCR requires noise whitening as the mean value of the noise term in the quadratic form of (10) is not proportional to identity, but to a positive diagonal matrix. In the illustrative example of figure 1 and under the white noise assumption the noise term satisfies:

$$E(\overline{\mathbf{W}}^H \overline{\mathbf{W}}) \propto diag(2,1,3,1,1)$$

where matrix $\overline{\mathbf{W}}$ is defined from the noise term similarly to $\overline{\mathbf{Y}}$. This would require a noise whitening to obtain unbiased estimation of the channel parameters.

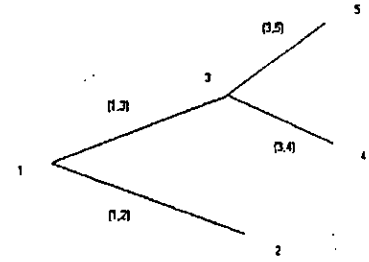


Figure 1: Example of a tree structure for $q = 5$.

3.3. Unbiased MCR Method

We introduce here a modified version of the MCR in such a way that the contribution of the noise term becomes proportional to identity. More precisely, instead of using $(q-1)$ cross-relations as in the MCR, we do use q cross-relations corresponding to the following pairs:

$$\begin{cases} p_1 = (1,2) \\ p_2 = (2,3) \\ \quad \vdots \\ p_{q-1} = (q-1, q) \\ p_q = (q,1) \end{cases} \tag{11}$$

where the first $(q-1)$ pairs correspond to the MCR, and the last one represents a redundancy chosen such that all system outputs are used similarly[1].

This has also the advantage of rendering the performances of the method independent from a specific choice of the selected tree structure of the MCR. This leads to a slight performance gain as observed in the simulation results (see section 5). In addition, in the UMCR, noise whitening is not necessary as stated by the following theorem:

Theorem 3 *Under the data model assumptions, the channel parameter estimate given by the least squares solution of the cross-relations of (11) is asymptotically unbiased.*

4. GENERALIZED CR CRITERION

Equation (3) can be rewritten in a more compact form as:

$$[\mathbf{F}(h)] * \mathbf{y}(k) = 0 \tag{12}$$

or equivalently (dual form):

$$[\mathbf{G}(y)] * \mathbf{h}(k) = 0 \tag{13}$$

[1] This is not the case in the MCR as certain system outputs are used more than others. For example, in the first $(q-1)$ pairs of (11) system outputs 1 and q are used only once while the others are used twice.

where $[\mathbf{F}(h)]$ and $[\mathbf{G}(y)]$ are two matrix-valued operators depending linearly on the channel parameters and observation signals, respectively, according to:

$$\mathbf{F}(h) = \begin{bmatrix} h_2 & -h_1 & 0 & \dots & 0 \\ h_3 & 0 & -h_1 & \dots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ h_q & 0 & \dots & 0 & -h_1 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \dots & 0 & h_q & -h_{q-1} \end{bmatrix} \quad (14)$$

$$\mathbf{G}(y) = \begin{bmatrix} y_2 & -y_1 & 0 & \dots & 0 \\ y_3 & 0 & -y_1 & \dots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ y_q & 0 & \dots & 0 & -y_1 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \dots & 0 & y_q & -y_{q-1} \end{bmatrix} \quad (15)$$

In this case $[\mathbf{F}(h)]$ and $[\mathbf{G}(y)]$ have well defined specific forms given by (14) and (15). However, this specific forms are not necessary to achieve unique identification of the system parameters and thus a large class of functions $[\mathbf{F}(h)]$ and $[\mathbf{G}(y)]$ different from those used in (14) and (15) can provide admissible identification criteria (i.e. a criterion is said to be admissible if it yields a unique identification of the system parameters).

Also, we allow $[\mathbf{F}(h)]$ and $[\mathbf{G}(y)]$ to be linear or non-linear, explicit or implicit function of the channel parameters and observation signals respectively.

Note that the second form of criterion (15) is generally preferred for channel identification since it depends linearly on the unknown channel parameters. However, this latter is not always possible to derive from the first form (14) when $[\mathbf{F}(h)]$ is a non-linear or an implicit function of $\mathbf{h}$ (see the multi-input case below).

Using this general formulation it can be shown that methods as maximum likelihood [3], Subspace [2], MNS (Minimum Noise Subspace) [4], etc. are special members of the class of generalized CR methods.

For example the subspace method in [2] consists of estimating the channel parameters using the signal and noise subspace orthogonality. The estimation criterion can be written as:

$$\mathbf{\Pi}\ \mathcal{T}_N(h) = 0 \quad (16)$$

or equivalently:

$$\overline{\mathbf{\Pi}}(k) * \mathbf{h}(k) = 0 \quad (17)$$

where $\mathbf{\Pi}$ represents the noise subspace projection, $\mathcal{T}_N(h)$ is a block Sylvester matrix, and $\overline{\mathbf{\Pi}}$ is a filtering matrix computed from $\mathbf{\Pi}$ as:

$$\overline{\mathbf{\Pi}}_{ij}(k) = \mathbf{\Pi}_{i,j+kq} \quad (18)$$

Equation (17) has the form of (13) where $[\mathbf{G}(y)]$ is given here by $\overline{\mathbf{\Pi}}$, that is a non-linear, implicit function of the observation process $\mathbf{y}(k)$ (recall that the noise projection $\mathbf{\Pi}$ is computed from the data $\mathbf{y}(k)$ through the eigen decomposition of its covariance matrix).

5. THE MULTI-INPUT CASE

In the multi-input case the transfer function becomes a $q \times p$ matrix $\mathbf{H}(z)$, $1 < p < q$ being the number of input signals. The noise-free observation can be written as:

$$\begin{aligned} \mathbf{y}_{(p)}(k) &= [\mathbf{H}_{(p)}(z)]\mathbf{s}(k) \\ y_{p+1}(k) &= [\mathbf{H}_{p+1,:}(z)]\mathbf{s}(k) \\ &\vdots \\ y_q(k) &= [\mathbf{H}_{q,:}(z)]\mathbf{s}(k) \end{aligned} \quad (19)$$

with $\mathbf{y}_{(p)}(k) = [y_1(k), \dots, y_p(k)]^T$, where $\mathbf{H}_{(p)}(z)$ is the top $p \times p$ sub-matrix of $\mathbf{H}(z)$, the $\mathbf{H}_{i,:}(z)$ is the $1 \times p$ ith row vector of $\mathbf{H}(z)$ and $[\mathbf{H}(z)]\mathbf{s}(k)$ represents the system outputs, corresponding to a transfer function $\mathbf{H}(z)$ excited by $\mathbf{s}(k) \stackrel{\text{def}}{=} [s_1(k), \cdots, s_p(k)]^T$ the p-dimensional vector of independent source signals.

For unique channel identifiability, $\mathbf{H}(z)$ is assumed to be irreducible [5] i.e:

$$Rank[\mathbf{H}(z)] = p, \quad \forall z$$

In addition, we assume here that the $p \times p$ sub-matrix $\mathbf{H}_{(p)}(z)$ is full rank, i.e:

$$\det(\mathbf{H}_{(p)}(z)) \not\equiv 0$$

In that case, the following $q - p$ cross-relations can be obtained by:

$$\mathbf{s}(k) = [H_{(p)}^{-1}(z)]\mathbf{y}_{(p)}(k) = [\frac{\text{com}(\mathbf{H}_{(p)}(z))}{\det(\mathbf{H}_{(p)}(z))}]\mathbf{y}_{(p)}(k) \quad (20)$$

or equivalently:

$$\begin{aligned} [\det(\mathbf{H}_{(p)}(z))]y_{p+1}(k) &= [\mathbf{H}_{p+1,:}(z)\text{com}(\mathbf{H}_{(p)}(z))]\mathbf{y}_{(p)}(k) \\ &\vdots \\ [\det(\mathbf{H}_{(p)}(z))]y_q(k) &= [\mathbf{H}_{q,:}(z)\text{com}(\mathbf{H}_{(p)}(z))]\mathbf{y}_{(p)}(k) \end{aligned} \quad (21)$$

where $\text{com}(\mathbf{A})$ and $\det(\mathbf{A})$ denote the co-factor matrix and determinant of $\mathbf{A}$, respectively. Under above assumptions, this set of cross-relations yields a unique identification[2] of the channel parameters (up to a constant $p \times p$ non-singular matrix which represents in fact the inherent indeterminacy of the MIMO-BSI problem).

Unfortunately, as we can see in the case $p > 1$, the noise vectors are non-linear functions[3] of channel parameters. Therefore, a simple extension of the cross-relations algorithm to the multi-input case seems not to be possible.

6. SIMULATION RESULTS

We present here some numerical simulations to assess the performances of the proposed CR-like methods. We consider a SIMO system with $q = 6$ outputs represented by polynomial transfer function of degree $M = 4$. The channel coefficients are generated randomly (at each Monte-Carlo run) following the complex gaussian distribution, i.e. the amplitude of each channel coefficient is Rayleigh distributed with unit-variance while its phase is uniformly distributed in $[0, 2\pi]$. The input signal is a 4QAM iid sequence of length $T = 256$. The observation is corrupted by addition white gaussian noise with a variance σ^2 chosen such that the $SNR = \frac{\|\mathbf{h}\|^2}{\sigma^2}$ varies in the range $[0, 30]dB$.

[2] The proof is omitted here due to space limitation.

[3] In the mono-input case, noise vectors are expressed as linear functions of the channel parameters as shown by eq.(14).

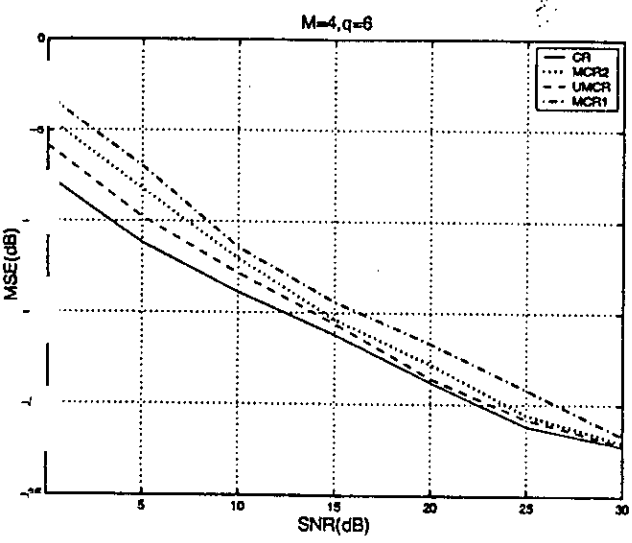


Fig : 2: Performance comparison of the CR, MCR and UMCR for = 6.

atistics are evaluated over $N_r = 1000$ Monte-Carlo runs and stimation performances are given by the normalized mean-square error criterion:

$$MSE = \frac{1}{N_r} \sum_{r=1}^{N_r} \frac{\|\hat{\mathbf{h}}_r - \mathbf{h}_r\|^2}{\|\mathbf{h}_r\|^2}$$

Wh : $\hat{\mathbf{h}}_r$ denotes the estimated channel coefficient vector at the -th Monte-Carlo run.

In figure 2, we compare the performances of the CR, the MCR refe :d to as MCR1 with $p_i = (1, i)$, $i = 2, \ldots, q$, the UMCR and .e MCR with $p_i = (i-1, i)$, $i = 2, \ldots, q$ referred to as MCR2.

e can observe that the choice of the structure underlying the CR method can affect significantly the performances of the method. Also, we observe a slight loss of estimation performances of the UMCR compared to the CR method, but the former remains con itationally much more efficient.

7. CONCLUSION

In t paper we have presented several extensions of the CR method priginally introduced in [5]. These extensions consist of a general formulation of the CR criterion, a minimum CR (MCR) method, an ı iiased MCR method and a discussion of the CR method for the IMO case.

The MCR and UMCR methods presented in this paper are sim "fied version of CR that might reduce significantly the computi onal cost of the blind channel estimation especially for large systems.

8. REFERENCES

1] H. Liu, G. Xu, and L. Tong, "A deterministic approach to blind entification of multichannel FIR systems," *in Proc. ICASSP*, ol. 4, pp. 581–584, 1994.

[2] E. Moulines, P. Duhamel, J. Cardoso, and S. Mayrargue, "Subspace methods for the blind identification of multichannel FIR filters," *IEEE Tr. on Sig. Proc.*, pp. 516–525, Feb. 1995.

[3] Y. Hua, "Fast maximum likelihood for blind identification of multiple FIR channels," *IEEE Tr. on Sig. Proc.*, Mar. 1996.

[4] Y. Hua, K. Abed-Meraim and M. Wax, "Blind system identification using minimum noise subspace," *8th IEEE Workshop on SSAP, Corfu, Greece*, June 1996.

[5] G. Xu, H. Liu, L. Tong, and T. Kailath, "A least-squares approach to blind channel identification." *IEEE Trans. Signal Processing*, vol. 43, pp. 2982–2993, Dec. 1995.

[6] K. Abed-Meraim, W. Qiu, and Y. Hua, "Blind system identification," *IEEE Tr. on Sig. Proc*, vol. 85, No 8, Aug 1997.

[7] L. Tong and S. Perreau, "Multichannel blind identification: from subspace to maximum likelihood methods", *Proceedings of the IEEE*, Volume: 86 Issue: 10 , pp. 1951–1968, Oct. 1998

9. APPENDIX

Proof of Theorem 2: Let consider the Z-transform of the $(q-1)$ cross-relations of (5) in the noiseless case. That leads to :

$$\widetilde{\mathbf{H}}(z)^T \mathbf{h}'(z) = 0$$

where $\mathbf{h}'(z) = [h'_1(z), \ldots, h'_q(z)]^T$ and $\widetilde{\mathbf{H}}(z)$ is a $q \times (q-1)$ polynomial matrix which k-th column is the zero valued vector except for the k_1-th and k_2-th entries that are equal to $h_{k_2}(z)$ and $-h_{k_1}(z)$ respectively (with $p_k = (k_1, k_2)$).

According to [4] and thanks to the tree structure, the columns of $\widetilde{\mathbf{H}}(z)$ form a basis of the rational subspace Range$\{\mathbf{h}(z)\}^{\perp}$, i.e., the orthogonal rational subspace to Range$\{\mathbf{h}(z)\}$. As a consequence $\mathbf{h}'(z)$ belongs to Range$\{\mathbf{h}(z)\}$ and since it is a polynomial vector with degree M equal to that of $\mathbf{h}(z)$, we have

$$\mathbf{h}'(z) = \alpha \mathbf{h}(z)$$

for a given scalar constant α.

Proof of Theorem 3: In presence of noise, matrix $\overline{\mathbf{Y}}$ in (10) constructed from the set of pairs in (11) becomes :

$$\overline{\mathbf{Y}} = \overline{\mathbf{X}} + \overline{\mathbf{W}}$$

where $\overline{\mathbf{W}}$ represents the additive noise term. Under the spatial and temporal white noise assumption and the independence of the noise and signal term, we have:

$$E(\overline{\mathbf{Y}}^H \overline{\mathbf{Y}}) = E(\overline{\mathbf{X}}^H \overline{\mathbf{X}}) + E(\overline{\mathbf{X}}^H \overline{\mathbf{W}}) + E(\overline{\mathbf{W}}^H \overline{\mathbf{X}}) + E(\overline{\mathbf{W}}^H \overline{\mathbf{W}})$$

the second and third term in the right side of above equation is equal to zero because of the independence of noise and signal terms. The last term is equal to :

$$E(\overline{\mathbf{W}}^H \overline{\mathbf{W}}) = 2\sigma^2 (T - M) \mathbf{I}_{q(M+1)}$$

where σ^2 is the noise power, T the sample size and M the channel polynomial degree. Consequently, the channel estimate given by the least eigenvector of $E(\overline{\mathbf{Y}}^H \overline{\mathbf{Y}})$ coincide with that of the noiseless covariance matrix $E(\overline{\mathbf{X}}^H \overline{\mathbf{X}})$.

Annexe 5 : Rapport de l'analyse temporelle

```
--------------------------------------------------------------------------------
Release 4.2i - Timing Analyzer E.35
Copyright (c) 1995-2001 Xilinx, Inc.  All rights reserved.

Design file:              algo_cr.ncd
Physical constraint file: algo_cr.pcf
Device,speed:             xc2v8000,-5 (ADVANCED 1.96 2002-01-02)
Report level:             verbose report
--------------------------------------------------------------------------------


================================================================================
Timing constraint: PATH "PATHFILTERS" = FROM TIMEGRP "SOURCES" TO TIMEGRP
"DESTINATIONS" ;

 246160745 items analyzed, 0 timing errors detected.
 Maximum delay is  36.788ns.
--------------------------------------------------------------------------------
Delay:                  36.788ns (data path - negative clock skew)
  Source:               U2/U2/z_reg<2>
  Destination:          U2/U2/A0_reg<4><14>
  Data Path Delay:      36.788ns (Levels of Logic = 8)
  Negative Clock Skew:  0.000ns
  Source Clock:         U2/clk rising
  Destination Clock:    U2/clk rising

  Data Path: U2/U2/z_reg<2> to U2/U2/A0_reg<4><14>
    Delay type         Delay(ns)  Logical Resource(s)
    -------------------------  --------------------
    Tcko                  0.493   U2/U2/z_reg<2>
    net (fanout=108)     16.147   U2/U2/z<2>
    Tilo                  0.382   U2/U2/C1477
    net (fanout=37)       8.564   U2/U2/syn11563
    Tilo                  0.382   U2/U2/C1465
    net (fanout=16)       3.769   U2/U2/cell945
    Tilo                  0.382   U2/U2/C1417
    net (fanout=1)        2.731   U2/U2/syn41076
    Tilo                  0.382   U2/U2/C1416
    net (fanout=1)        0.002   U2/U2/syn41090
    Tilo                  0.382   U2/U2/C1415
    net (fanout=1)        2.467   U2/U2/syn41097
    Tilo                  0.382   U2/U2/C1399
    net (fanout=1)        0.001   U2/U2/member1338<14>
    Tdxck                 0.322   U2/U2/A0_reg<4><14>
    -------------------------  ---------------------------
    Total                36.788ns (3.107ns logic, 33.681ns route)
                                  (8.4% logic, 91.6% route)

--------------------------------------------------------------------------------
Delay:                  36.392ns (data path - negative clock skew)
  Source:               U2/U2/z_reg<2>
  Destination:          U2/U2/A0_reg<5><1>
  Data Path Delay:      36.392ns (Levels of Logic = 8)
  Negative Clock Skew:  0.000ns
  Source Clock:         U2/clk rising
  Destination Clock:    U2/clk rising

  Data Path: U2/U2/z_reg<2> to U2/U2/A0_reg<5><1>
    Delay type         Delay(ns)  Logical Resource(s)
    -------------------------  --------------------
    Tcko                  0.493   U2/U2/z_reg<2>
    net (fanout=108)     16.147   U2/U2/z<2>
    Tilo                  0.382   U2/U2/C1477
    net (fanout=37)       8.993   U2/U2/syn11563
    Tilo                  0.382   U2/U2/C1087
```

```
    net (fanout=8)        1.775   U2/U2/syn11477
    Tilo                  0.382   U2/U2/C1077
    net (fanout=16)       4.194   U2/U2/cell520
    Tilo                  0.382   U2/U2/C764
    net (fanout=1)        0.002   U2/U2/syn42963
    Tilo                  0.382   U2/U2/C763
    net (fanout=1)        2.173   U2/U2/syn42981
    Tilo                  0.382   U2/U2/C752
    net (fanout=1)        0.001   U2/U2/member1736<1>
    Tdxck                 0.322   U2/U2/A0_reg<5><1>
    --------------------------  ------------------------------
    Total                36.392ns (3.107ns logic, 33.285ns route)
                                  (8.5% logic, 91.5% route)

--------------------------------------------------------------------------------
Delay:                  36.149ns (data path - negative clock skew)
  Source:               U2/U2/z_reg<2>
  Destination:          U2/U2/A0_reg<5><2>
  Data Path Delay:      36.149ns (Levels of Logic = 8)
  Negative Clock Skew:  0.000ns
  Source Clock:         U2/clk rising
  Destination Clock:    U2/clk rising

  Data Path: U2/U2/z_reg<2> to U2/U2/A0_reg<5><2>
    Delay type         Delay(ns)  Logical Resource(s)
    -------------------------  --------------------
    Tcko                  0.493   U2/U2/z_reg<2>
    net (fanout=108)     16.147   U2/U2/z<2>
    Tilo                  0.382   U2/U2/C1477
    net (fanout=37)       8.993   U2/U2/syn11563
    Tilo                  0.382   U2/U2/C1087
    net (fanout=8)        1.775   U2/U2/syn11477
    Tilo                  0.382   U2/U2/C1077
    net (fanout=16)       4.243   U2/U2/cell520
    Tilo                  0.382   U2/U2/C785
    net (fanout=1)        0.002   U2/U2/syn42900
    Tilo                  0.382   U2/U2/C784
    net (fanout=1)        1.881   U2/U2/syn42918
    Tilo                  0.382   U2/U2/C773
    net (fanout=1)        0.001   U2/U2/member1736<2>
    Tdxck                 0.322   U2/U2/A0_reg<5><2>
    --------------------------  ------------------------------
    Total                36.149ns (3.107ns logic, 33.042ns route)
                                  (8.6% logic, 91.4% route)

--------------------------------------------------------------------------------

All constraints were met.

Data Sheet report:
------------------
All values displayed in nanoseconds (ns)

Setup/Hold to clock clk
---------------+------------+------------+
               |  Setup to  |  Hold to   |
Source Pad     | clk (edge) | clk (edge) |
---------------+------------+------------+
Y1<0>          |   -4.995(R)|   10.632(R)|
Y1<10>         |   -3.990(R)|   10.465(R)|
Y1<11>         |   -2.633(R)|   10.122(R)|
Y1<12>         |   -3.716(R)|   10.616(R)|
Y1<13>         |   -3.229(R)|   12.668(R)|
Y1<14>         |   -1.765(R)|   12.435(R)|
Y1<15>         |   -4.144(R)|   10.263(R)|
Y1<1>          |   -4.173(R)|   12.113(R)|
Y1<2>          |   -3.626(R)|   12.588(R)|
Y1<3>          |   -4.104(R)|   12.624(R)|
Y1<4>          |   -3.957(R)|   11.870(R)|
Y1<5>          |   -4.069(R)|   12.032(R)|
```

Y1<6>	-4.886(R)	9.507(R)
Y1<7>	-5.499(R)	9.314(R)
Y1<8>	-3.679(R)	12.313(R)
Y1<9>	-5.237(R)	8.891(R)
Y2<0>	-7.564(R)	14.446(R)
Y2<10>	-9.486(R)	15.252(R)
Y2<11>	-8.475(R)	14.630(R)
Y2<12>	-9.677(R)	15.517(R)
Y2<13>	-5.272(R)	11.227(R)
Y2<14>	-7.524(R)	15.270(R)
Y2<15>	-5.459(R)	12.867(R)
Y2<1>	-8.858(R)	14.357(R)
Y2<2>	-9.026(R)	14.528(R)
Y2<3>	-8.641(R)	14.471(R)
Y2<4>	-8.934(R)	14.707(R)
Y2<5>	-8.967(R)	14.261(R)
Y2<6>	-8.796(R)	14.394(R)
Y2<7>	-7.822(R)	14.881(R)
Y2<8>	-10.478(R)	15.455(R)
Y2<9>	-10.281(R)	14.808(R)
start	8.464(R)	14.133(R)

Clock clk to Pad

Destination Pad	clk (edge) to PAD
R<0>	8.073(R)
R<10>	8.494(R)
R<11>	7.346(R)
R<12>	8.382(R)
R<13>	7.816(R)
R<14>	7.445(R)
R<15>	8.333(R)
R<16>	8.379(R)
R<17>	8.340(R)
R<18>	8.061(R)
R<19>	7.788(R)
R<1>	8.619(R)
R<20>	7.691(R)
R<21>	8.337(R)
R<22>	8.302(R)
R<23>	8.426(R)
R<24>	8.113(R)
R<25>	7.124(R)
R<26>	8.701(R)
R<27>	8.058(R)
R<28>	8.323(R)
R<29>	8.078(R)
R<2>	8.573(R)
R<30>	8.458(R)
R<31>	8.093(R)
R<3>	9.982(R)
R<4>	10.510(R)
R<5>	8.179(R)
R<6>	8.063(R)
R<7>	8.571(R)
R<8>	8.176(R)
R<9>	7.980(R)

Clock to Setup on destination clock clk

Source Clock	Src:Rise|Dest:Rise	Src:Fall|Dest:Rise	Src:Rise|Dest:Fall	Src:Fall|Dest:Fall
clk	37.594			

```
WARNING:Timing:2554 - Clock nets using non-dedicated resources were found in
   this design. Clock skew on these resources will not be automatically
   addressed during path analysis. To create a timing report that analyzes
   clock skew for these paths, run trce with the '-skew' option.

   The following clock nets use non-dedicated resources:
      N_clk


Timing summary:
---------------

Timing errors: 0  Score: 0

Constraints cover 246160752 paths, 0 nets, and 112416 connections (100.0% coverage)


Analysis completed Wed Jun 11 11:14:49 2003
--------------------------------------------------------------------------------

Timing Analyzer Settings:
-------------------------
OpenPCF algo_cr.pcf
Speed -5
DoHoldRaceCheck False
IncludeNets
ExcludeNets
SelectFailingTimingConstraint False
IncludeNoTimingConstraint False
Report Normal
MaxPathsPerTimingConstraint 3
DefineEndpoints ToAll
DefineEndpoints FromAll
OmitUserConstraints False
DropTimingConstraint
SetForce Off
```

Bibliographie

Bibliographie :

Partie 1:

[1] : Gardner W.A. "A new method of channel identification." IEEE Tr. On Comm., 39:813-817 , August 1991.

[2] : Tong L. and Xu G. and Kailath T. "A new approch to blind identification and equalisation of multipath channels." In Proc. Of the 25th Asilomar conference, CA , pages 856-860 , 1991.

[3] : Jean-Marc Brossier. "Signal et communication numérique: égalisation et synchronisation". Collection traitement du signal.

[4] : Hua Y and Wax M. "Strict identifiability of multiple FIR channels driven by unknown arbitrary sequence." IEEE Tr. On Sig. Proc.,vol.9 , March 1996.

[5] : Y. Hua. "Fast Maximum likelihood for blind identification of multiple FIR channels." IEEE Tr. On Sig. Proc. March 1996.

[6] : Moulines E. and Duhamel P. and Cardoso J.F. and Mayrargue S. "Subspace methods for blind identification of multichannel FIR filters." In IEEE Trans. On Sig. Proc , Feb. 1995.

[7] : K. Abed-Meraim, W. Qin, and Y. Hua. "Blind System Identification." Proc. IEEE, Aus 1997.

[8] : Abed-Meraim K. and Hua Y. and Ikram M.Z. and Duhamel P. "A fast algorithm for conditional maximum likelihood blind identification of SIMO/MIMO FIR systems." in Sig. Proc.EUSIPCO, Sep. 2000.

[9] : Abed-Meraim K. and Hua Y. "New TSML algorithms for multichannel blind identification." in Sig. Proc. SYSID 97 , Japan, Feb. 1997.

[10] : Abed-Meraim K., Loubaton P. and Moulines E. "A subspace algorithm for certain blind identification problems." in IEEE Tr. On Inf. Theory, March 1997.

[11] : H. Liu, G. Xu, and Tong L. "A deterministic approch to blind identification of multichannel FIR systems." in Sig. Proc. ICASSP, vol. 4, pp. 581-584, 1994.

[12] : Van der Veen A.J, Tolwar S. and Paulraj A. "Blind identification of FIR channels carrying multiple finite alphabet signals". Proc. ICASSP, pages 1213-1216 , 1995.

[13] : Van der Veen A.J, "Analytical method for blind dinary signal separation". IEEE Tr. on Sig. Proc. Vol. 45.pp. 1078-1082, Ap.1997.

[14] : H. Liu and G. Xu, "Closed-Form blind symbol estimation in digital communications." Proc. ICASSP, pages 2714-2723,Nov.1995.

[15] : H. Liu, G. Xu, Tong L and Kailath T. "A least squares approach to nlind channel identification". ". IEEE Tr. on Sig. Proc. Dec. 1995.

[16] : Gesbert D. and Duhamel P . and Mayrague S. "Subspace based adaptative algorithms for the blind identification of multichannel FIR filters". In Proc. EUSIPCO, Edinourgh, Scotland,1994.

[17] : Giannakis G.B and Tepedelenlioglu C. "Direct blind equalization of multiple FIR channels : A deterministic approach". In Proc. Of the 30th Asilomar Conference, CA,1996.

[18] : Giannakis G., Duhamel P., and Mayrargue S. "On-line blind multichannel equalisation based on mutually referenced filters". ". IEEE Tr. on Sig. Proc , Sep.1997.

[19] : Slock D. "Blind fractionally-spaced equalization, perfect reconstruction filterbanks, and multilinear prediction". Proc.ICASSP, Apr. 1994.

[20] : Abed-Meraim K, Duhamel P, Gesbert D, Mayrargue S. Moulines E. and Slock D. "Prediction error methods for time-domain blind identification of multichannel FIR filters". Proc. ICASSP , 3 : 1968-1971, 1995.

[21] : Abed-Meraim K, Moulines E and Loubaton P. "Prediction error method for second order blind identification: Algorithms and statistical performance". IEEE Tr. on Sig. Proc. March 1997.

[22] : Golub & Van Loan. "Matrix computation". 3rd Edition. The Johns Hopkins University Press.

[23] : O. Shalvi and all. "Adaptive equalization." Proc. IEEE, 73:1349-1387, Sep 1985.

[24] : S. Bellini. "Blind equalization",. Altra Frequenza, LVII-N.7:445-450, Sep 1988.

[25] : O. Macchi and E.R. Raghuveer. "Bispectrum estimation: A digital signal processing framework.", Volume 75, pages 869-891, July 1988.

[26] : D. Hatzikos. "Multiple time series.", John Wiley, 1970.

[27] : R. Godfrey and F. Rocca. "Zero-memory non-linear deconvolution". Geophysical Prospecting , 1981.

[28] : C.L. Nikias. "ARMA bispectrum approach to nonminimum phase system Identification".volume 36, apr. 1988.

[29] : J.K. Tugnait. "Identification of non-minimum phase linear stochastic systems". .Automatica, 1986.

[30] : K.S . Lii and M. Rosenblatt. "Deconvolution and estimation of transfer function phase and coefficients for non-gaussian linear processes." Annuals of Statistics, 1982.

[31] : G. Giannakis and J.M. Mendel." Identification of non-minimum phase systems using higher-order statistics". IEEE Tr. On Acoust. Speech and Sig. Proc., 1989.

Partie 2:

[1] : Virtex™-II Platform FPGAs: Introduction and Overview.
XILINX. « *http :www.xilinx.com /legal.htm* » Virtex-II .1.5.FPGA DS 031-1 (v1.9) September 26, 2002 . Advance Product Specification.

[2] : Virtex™-II Platform FPGAs: Detailed Description.
XILINX. « *http :www.xilinx.com /legal.htm* » Virtex-II .1.5.FPGA DS 031-2 (v2.1.1) December 6, 2002 . Advance Product Specification.

[3] : Xcell journal. Issue 43. Summer 2002. XILINX. "IBM ASICs Will Incorporate Xilinx FPGAs".

[4] : XILINX .2001 Inc. " Virtex-II Architecture I" .

[5] : XILINX. « *http :www.xilinx.com /legal.htm* » .

[6] : R. Airiau. J .M. Berge. V .Olive. "Circuit Synthesis With VHDL", Kluwer Academie Publishers. 1994.

[7] : XILINX. "Synthesis and Simulation design guide" . Xilinx Inc. USA .1998.

[8] : ISE Fondation User Manuel.

[9] : Lucazeau. B.Trézéguet. H. ASIC : "Circuits intégrés pour applications spécifiques" Technique de l'ingénieur . Traité électronique.Vol.2. pp. E 2418-1 à E2418-4.

[10] : C. Pignet. and Strauffer. Bordas. "Synthése de circuits ASIC". edition Dunod.

[11] : Michael John Sebastian Smith. "Apllication Integrated Circuits".

[12] : Bernard, Jean-Michel and Breteuil. H. " Les Circuits programmables", 1983 Collection Système d'exploitation.

[13] : T. Leighton, "Introduction to Parallel Algorithm and Architectures: Arrays, Trees, Hypercubes," San Mateo, Calif., Morgan Kaufmann, 1992.

[14] : A.K. Oudjida, Z. Sahraoui, A. Bellaouar and A. Akkouche. Proc "Design and Analysis of a High Performance Multiplier and its Generator", of the international Conference on Microelectronics ICM'90, pp 2-32-1 : 2-32-14, October 13-17 1990.

[15] : A.K. Oudjida."High Speed and Very Compact Two's Complement Serial/Parallel Multipliers Using Xilinx's FPGA." Proc of the 7th International Conference on Signal Processing Application and Technology ICSPAT'96, vol 1 pp 924-928.

[16] : A.K. Oudjida, S. Titri and M. Hamralain "N latency, 2N I/O-bandwidth 2D array matrix multiplication algorithm", the international journal for computation and mathematics in electrical and electronics engineering COMPEL, vol 21 issue 3, 2002 UK.

[17] : A.K. Oudjida, S. Titri and M. Hamralain "Mapping full-systolic array for matrix product in xilinx's XC4000(E,EX) FPGAs", the international journal for computation and mathematics in electrical and electronics engineering COMPEL, vol 21 issue 3, 2002 UK.