Table des matières

Liste des tableaux

Liste des figures

Liste des symboles

AG	Algorithme Génétique
A/D	Analog to Digital
ANN	Artificial Neural Network
ARMA	Auto Regressive Moving Average
ASIC	Application-Specific Integrated Circuits
AWGN	Additive White Gaussian Noise
BER	Bit Error Rate
CDFRNN	Complex Decision Feedback Recurrent Neural Network
CRTRL	Complex Real Time Recurrent Learning
CSD	Canonic Signed Digit
D/A	Digital to Analog
EKF	Extended Kalman Filter
FIR	Finite Impulse Response
FPGA	Field Programmable Gate Array
FXLMS	Filtred-X Least Mean Square
HPC	High-Performance Computing

ISI	Intersymbol Interference
LE	Linear Equalizer
LMS	Least Mean Square
LUT	Look Up Table
MLP	Multi-Layer Perceptron
MSE	Mean Squared Error
NLMS	Normalized Least Mean Square
QAM	Quadrature Amplitude Modulation
RLS	Recursive Least Square
RNN	Recurrent Neural Network
SNR	Signal to Noise Ratio
SPT	Signed Power of Two
SQNR	Signal to Quantization Noise Ratio
UKF	Unscented Kalman Filter
VDFGA	Volterra Decision Feedback Genetic Algorithm
VHDL	VHSIC Hardware Description Language
VHSIC	Very High Speed Integrated Circuit
VLSI	Very Large Scale Integration
B	Taille de la fenêtre de l'AG

β	Constante positive du RLS
$c(n)$	Reponse impulsionnelle du canal
$C(n)$	Matrice de covariance du RLS
C_v	Coefficient de variation
D	Délai
$e(n)$	Signal d'erreur
$f(n)$	Fonction de fitness
g	Génération actuelle
G	Nombre de générations totales
$k(n)$	Gain du RLS
L	Longueur du canal
M	Nombre de coefficients du filtre
N	Nombre de neurones
P	Population
P_c	Probabilité de recombinaison
P_m	Probabilité de mutation
P_{XXij}^k	Terme de sensibilité
R	Nombre de répétitions

$s(n)$	Symbole transmit
$\hat{s}(n)$	Symbole estimé
$u(n)$	Signal du bruit
$\hat{w}(n)$	Coefficients du filtre estimés
$w(n)$	Poids du filtre
W_{ij}	Poids qui lie le neurone i avec l'entrée j
W_L	Longueur du mot binaire
$x(n)$	Signal d'entrée
$y(n)$	Signal de sortie
$\hat{y}(n)$	Signal estimé
μ	Pas d'adaptation du LMS
λ	Facteur d'oubli
$\eta(n)$	Bruit blanc gaussien additif
δ_{ik}	Fonction de Kronecker
σ^2	Variance du bruit

Chapitre 1 - Introduction

Au fil des dernières décennies, beaucoup de travaux de recherche et de développement dans le domaine des technologies des communications ont vu le jour. Ce mouvement a permis de mettre au point des techniques qui sont beaucoup plus performantes que celles qui existaient auparavant, mais de plus en plus difficile à réaliser pratiquement. Ce besoin de nouvelles technologies d'intégration a été satisfait avec l'apparition du procédé de fabrication de circuits intégrés à très grande échelle d'intégration (VLSI- Very Large Scale Integration) permettant de réaliser des circuits qui peuvent fonctionner à des très grandes vitesses de calcul, tout en nécessitant une faible surface d'intégration. Ce procédé a été très vite utilisé dans des travaux de recherche qui sont orientés à traiter des problèmes dans d'autres domaines technologiques comme la mesure, le contrôle électronique ainsi que le biomédical.

Le traitement numérique des signaux avec ses différentes techniques et algorithmes est omniprésent dans plusieurs domaines technologiques. Parmi ces algorithmes, on retrouve les méthodes de filtrage adaptatif numérique [1]. Ces techniques permettent de mettre à jours les coefficients du filtre de manière itérative. Elles sont utilisées dans de nombreuses applications comme l'identification de systèmes [2, 3], l'égalisation de canaux de transmission [4, 5] et l'annulation de bruit [6]. On retrouve principalement ces applications dans les systèmes de télécommunication [7-9], les systèmes de contrôle [10-14] et le biomédical [15, 16].

Les algorithmes de filtrage adaptatif linéaires les plus populaires sont le LMS et le RLS (*Recursive Least Square*). Le LMS fait partie de la classe des algorithmes du gradient stochastique alors que le RLS fait partie de celle des algorithmes des moindres carrées. Le RLS est beaucoup plus performant que le LMS du point de vue vitesse de convergence, erreur asymptotique et insensibilité aux paramètres propres du système adressé. Cependant, ces avantages se traduisent par une plus grande complexité de calcul [17]. Ce point très important a permis de favoriser l'utilisation du LMS grâce à sa simplicité et sa facilité d'implémentation [18], ainsi que l'apparition de plusieurs variantes comme le NLMS (*Normalized LMS*) [19] et le FXLMS (*Filtred-X LMS*) [20].

L'inconvénient majeur de ces algorithmes est la dégradation de leurs performances lorsqu'il s'agit d'adresser des problèmes non linéaires. Ces non-linéarités sont très présentes dans la pratique, elles peuvent être intrinsèques au système ou causées par des circuits impliqués dans la mise en forme et le traitement du signal (capteurs, amplificateurs, convertisseurs A/D (*Analog to Digital*) et D/A (*Digital to Analog*), processus de modulation/démodulation...etc.). Ce désavantage provient de l'incapacité de ces algorithmes linéaires à modéliser des systèmes non linéaires. Afin d'apporter une solution à cette limitation, des nouvelles techniques de filtrage adaptatif basées sur différents concepts comme les ANNs (*Artificial Neural Network*) [21]et les séries de Volterra [22], ont vu le jour.

Lorsqu'il s'agit d'implémentation d'algorithmes en technologie VLSI, l'utilisation d'un calcul arithmétique en virgule fixe au lieu d'un calcul en virgule flottante permit de réduire considérablement la complexité d'implémentation. Ce qui se traduit par une faible surface d'intégration, une faible consommation électrique et une plus grande vitesse de calcul.

Cependant, le choix de la longueur des mots binaires nécessaire est crucial pour maintenir un bon niveau de performances. Le nombre et la nature des opérations arithmétiques déterminent la complexité de l'algorithme. Un simple algorithme comme le LMS peut avoir une robustesse considérable face aux effets de quantification introduits par la troncature des mots binaires, contrairement à un ANN ou un RLS qui nécessitent des longueurs binaires beaucoup plus importantes pour maintenir les performances observées dans le calcul en virgule flottante [23-25].

1.1 Problématique

Une implémentation en virgule fixe présente énormément d'avantages par rapport à une implémentation en virgule flottante. Parmi ces avantages on retrouve la réduction des coûts de fabrication, la réduction de la consommation électrique et l'augmentation de la vitesse de traitement. Ces caractéristiques devient déterminants lorsqu'il s'agit d'implémentation sur dispositifs cibles à ressources matérielles limitées comme les FPGAs (*Field Programmable Gate Array*) et les ASICs *(Application-Specific Integrated Circuit).*

Le passage d'un calcul en virgule flottante à un calcul en virgule fixe à faible longueur binaire provoque une dégradation de performances à cause des erreurs de quantification causées par la troncature des mots binaires d'une part, et la récurrence dans les équations de mise à jour des coefficients du filtre d'autre part. Cette dégradation peut être tolérable quand des algorithmes de filtrage adaptatif linéaires à faible complexité de calcul comme le LMS sont visés. Cependant, adresser des systèmes non-linéaires ou solliciter des performances élevées implique l'utilisation de techniques plus complexes. L'implémentation de ces méthodes nécessite beaucoup de ressources matérielles à cause du

grand nombre d'opérations arithmétiques utilisées, et la forte longueur des mots binaires requise.

Dans ce mémoire, une technique basée sur les AGs est utilisée pour traiter le problème d'identification de systèmes linéaires et non-linéaires, ainsi que l'égalisation de canaux de transmission non-linéaires, tout en considérant un environnement de calcul en virgule fixe à très faible longueur binaire, dont la valeur est comprise dans un intervalle qui s'étale de 16 bits à 4 bits. Cette application est rendue possible grâce au mécanisme de sélection naturelle des AGs qui ne nécessite pas de calcul arithmétique pour mettre à jours les coefficients du filtre, mais une stratégie de remplacement complet des coefficients. Les AGs offrent aussi une très grande vitesse de convergence lorsqu'ils sont adéquatement paramétrés, ce qui est très important pour quelques applications comme l'égalisation. Des opérateurs génétiques adaptés pour le filtrage adaptatif sont proposés pour améliorer la capacité de recherche des coefficients optimaux et la vitesse de convergence du filtre. Une architecture matérielle de l'algorithme introduit est proposée, ainsi qu'une implémentation sur FPGA ciblant le cas d'identification de systèmes.

1.2 Objectifs

L'objectif principal de ce mémoire est de concevoir et implémenter sur FPGA un AG fonctionnant avec une arithmétique à très faible longueur binaire, pour adresser des applications liées au filtrage adaptatif. Les opérateurs génétiques sont spécialement conçus pour offrir une robustesse aux effets de quantifications et améliorer les performances de traitement de l'algorithme. La technique est appliquée aux problèmes d'identification de

systèmes linéaires et non-linéaires, ainsi que l'égalisation de canaux de transmission non-linéaires. Les sous objectifs sont définis comme suit :

- Proposition d'une méthode adaptative basée sur les AG en comparaison au filtrage par LMS dans le cas d'application d'identification des paramètres d'un modèle ARMA linéaire et non-linéaire.
- Proposition d'un filtre adaptatif non linéaire de Volterra basé sur les AG appliqué à l'égalisation de canaux de communication non-linéaire. Comparaison des résultats à un égaliseur à base d'un réseau de neurone (CDFRNN) en considérant une modulation complexe de type 4-QAM (*Quadrature Amplitude Modulation*).
- Proposition d'une architecture et de son implémentation sur un FPGA cible d'un filtre adaptatif à base d'un AG.

1.3 Méthodologie

Après avoir fait l'étude bibliographique nécessaire sur le filtrage adaptatif, en particulier sur les algorithmes génétiques et les difficultés d'implémentation en technologie VLSI, une vision plus claire sur la problématique et la solution potentielle au problème viennent à l'esprit. Les informations reçues sur les techniques de filtrage adaptatif permettent de comprendre la complexité de calcul nécessaire pour le déroulement de la technique en question et ses faiblesses lorsqu'un environnement de calcul en virgule fixe à faible longueur binaire est considéré. Les recherches bibliographiques sur les AG permettent de bien choisir les sous-opérations de la technique comme : le codage des chromosomes, la recombinaison et la mutation pour améliorer la vitesse de convergence de l'algorithme.

Les applications du filtrage adaptatif ciblées sont : l'identification de système et l'égalisation de canaux de transmission. Dans l'identification de système, l'AG sera utilisé pour identifier les paramètres d'un modèle ARMA linéaire et non-linéaire. Les résultats seront comparés avec les performances du LMS et les résultats théoriques. Les critères de performances utilisés sont le SQNR (*Signal to Quantization Noise Ratio*) et le C_v (*Coefficient of Variation*). Dans le cas de l'égalisation, la technique développée à base d'AG, nommée VDFGA (*Volterra Decision Feedback Genetic Algorithm*) sera comparée avec le CDFRNN, dans le contexte d'un canal non-linéaire à forte ISI (*Intersymbol Interference*). Les critères de performances considérés sont le MSE (*Minimum Squared Error*) et le BER (*Bit Error Rate*). Toutes les simulations nécessaires pour obtenir les résultats recherchés seront faites avec l'outil de simulation MATLAB. Les simulations en virgule fixe nécessitent des très grands temps de calcul. Pour répondre à ce besoin, une plateforme de calcul haute performance (HPC – High-Performance Computing) sera utilisée, avec un total de 50 cœurs. Avec cette configuration, un gain en temps de simulation théorique de 50 est atteint, comparé à une machine ordinaire mono-cœur. Des simulations de plusieurs jours sont réduites à quelques heures seulement. Ces ressources sont fournies par l'organisme Calcul Canada.

La description matérielle de l'architecture de l'AG proposée sera effectuée avec le langage VHDL (*VHSIC Hardware Description Language*), en utilisant l'outil ModelSim de Mentor Graphics. La vérification du fonctionnement comportemental de l'architecture proposée et le rapport d'utilisation des ressources seront faites respectivement avec le simulateur ISim et l'outil de synthèse XST de Xilinx. L'implémentation sera faite sur un dispositif de la famille Spartan 6 de Xilinx.

1.4 Travaux relatifs au filtrage à base d'algorithmes génétiques

L'utilisation des AGs dans le filtrage ne date pas d'aujourd'hui. La plupart des travaux trouvés dans la littérature adressent le problème de conception de topologies et gabarits pour les filtres [26-30]. L'optimisation à base d'AG est utilisée pour rechercher les paramètres optimaux du filtre qui permettent de satisfaire des contraintes désirées comme les fréquences de coupures, les niveaux d'atténuations, facteur de qualité et la variation de la phase. Tous ses paramètres sont optimisé hors-ligne à l'aide des AGs.

Dans le cas des filtres analogiques, l'AG est généralement sollicité pour trouver les valeurs optimales des composants passifs (résistances, capacités, etc.), afin de garantir les meilleures performances. L'objectif de la méthode est d'arriver à une configuration où l'erreur globale introduite par l'utilisation de composants normalisés, soit minimale [26]. Dans le filtrage numérique, l'AG est utilisé pour trouver les coefficients du filtre optimaux qui permettent de reproduire la fonction de transfert désirée avec un maximum d'exactitude [29]. Encore une fois, ces optimisations des paramètres se réalisent hors-ligne.

D'autres travaux s'intéressent à la réduction de la complexité d'implémentation des filtres numériques. Les coefficients du filtre sont représentés sous forme SPT *(Signed Power of Two)* ou CSD *(Canonic Signed Digit)* pour remplacer les opérations de multiplication par des simples opérations d'additions et de décalages binaires, conduisant ainsi à une réduction de complexité lorsque le nombre de termes employés dans la représentation est assez faible [31, 32]. Dans ce contexte, l'AG est utilisé pour trouver le nombre minimum de termes à prendre, tout en garantissant un niveau de performance acceptable. La recherche des solutions minimisant les ressources dans un contexte d'implémentation en arithmétique entière est réalisée hors-ligne. Suite à la solution

optimale recherchée, des coefficients respectant la forme SPT, CSD ou autre sont implémentés.

Dans ce travail, le cas du filtrage adaptatif exigeant une mise à jour en ligne des coefficients d'un filtre sera réalisé à base d'AG. Plusieurs travaux de recherches peuvent être trouvés dans la littérature [33-35]. Ces motivations sont justifiées par la faculté des AGs à trouver la solution globale au problème d'optimisation, tout en offrant la possibilité d'adresser des problèmes linéaires et non-linéaires. L'objectif est d'exploiter la capacité de recherche des algorithmes génétiques pour trouver les coefficients du filtre qui permettent d'approcher la solution optimale de Wiener. Cependant, tous ces travaux considèrent un environnement de calcul en virgule flottante.

Peu de travaux ont investigué les effets du calcul en virgule fixe sur les performances des AGs dans le filtrage adaptatif en arithmétique entière. Dans [24], les auteurs ont étudié les effets de quantifications sur les performances d'un AG, en considérant le problème d'identifications de paramètres d'un modèle ARMA linéaire et non-linéaire. Les résultats de simulation ont montré la robustesse de l'AG aux effets de quantification, même lorsque des mots binaires d'une longueur de seulement 6-bits sont considérés. Cependant, une contrainte limitant la recherche dans le voisinage immédiat des paramètres recherchés a été appliquée. En pratique, cette action est possible seulement si des informations préalables sur les coefficients recherchés sont disponibles.

Dans le cadre de ce travail, nous ne comptons pas limiter la plage des recherches et accepter toutes la plage dynamique que procurent les longueurs binaires utilisées.

1.5 Organisation du mémoire

Dans le chapitre 2, le principe du filtrage adaptatif et les facteurs de sélection d'un algorithme adaptatif sont donnés. Puis, un survol sur les différentes techniques linéaires et non-linéaires de référence est effectué. Ensuite, le fonctionnement de base d'un algorithme génétique est présenté. Dans la dernière partie du chapitre, le calcul en virgule fixe dans les AGs est abordé.

Le développement de l'AG proposé dans ce mémoire est couvert dans le chapitre 3. Les étapes nécessaires à l'exécution de l'algorithme génétique sont présentées, ainsi que la manière de coder les chromosomes. Les opérateurs génétiques proposés sont décortiqués est présentés en détail. Ensuite, le problème d'identification des paramètres d'un modèle ARMA linéaire et non-linéaire est abordé. À la fin, les performances de l'AG proposé sont évaluées, en faisant une comparaison avec le LMS et les performances théoriques.

Dans le chapitre 4, une technique à base d'AG est utilisée pour égaliser un canal de communication non-linéaire. Le problème d'égalisation est d'abord décrit, suivi par une introduction de la technique de référence CDFRNN. Après, la méthode proposée nommée VDFGA (*Volterra Decision Feedback Genetic Algorithm*) est exposée. Dans la dernière partie, la description mathématique du canal est donnée. Par la suite, les résultats de comparaison entre le CDFRNN et le VDFGA sont fournis et discutés.

Le chapitre 5 est dédié à l'implémentation sur FPGA de l'AG détaillé dans le chapitre 3, considérant l'application d'identification de système. L'architecture matérielle de chaque bloc est d'abord donnée et expliquée, suivie par une description du problème d'identification des paramètres du model ARMA. Un rapport d'implémentation indiquant les ressources utilisées ainsi que les performances achevées est donné. À la fin, une

validation du bon fonctionnement de l'implémentation est effectuée, en vérifiant la capacité du system à identifier correctement les paramètres du model considéré, dans le cas linéaire et non-linéaire.

Le dernier chapitre de ce mémoire donnera une conclusion générale de l'étude, suivie de la bibliographie.

Chapitre 2 - Filtrage adaptatif et algorithmes génétiques

2.1 Filtrage adaptatif

Les développements modernes dans le domaine du filtrage adaptatif ont commencé dans les années 30 et 40 avec les travaux de Kolmogorov, Wiener et Levinson pour résoudre des problèmes d'estimation linéaire [36]. Le filtre connu sous le nom de Wiener-Kolmogorov est la solution optimale dans le sens de l'erreur quadratique. Cependant, ce filtre nécessite des informations préalables sur les statistiques des données à traiter, ce qui n'est pas généralement possible en pratique. L'utilisation de techniques de filtrage adaptatif comme le LMS et le RLS résout ce problème. Le choix de l'algorithme adéquat repose sur les facteurs suivants [1]:

- **Vitesse de convergence** : représente le nombre d'itérations requis par un algorithme pour atteindre la solution optimale de Wiener en considérant des entrées stationnaires. Une vitesse de convergence rapide permet à l'algorithme de s'adapter rapidement à l'environnement stationnaire des statistiques inconnues.
- **Désajustement** : fournit une mesure quantitative sur la déviation entre le minimum du MSE produit par le filtre de Wiener et la valeur finale du MSE de l'algorithme en question, après avoir calculer la moyenne sur un ensemble de filtres adaptatifs.
- **Poursuite** : est présente lorsqu'un algorithme de filtrage adaptatif opère dans un environnement non stationnaire, l'algorithme doit être capable de

poursuivre les variations statistiques dans l'environnement. Les performances de poursuite sont influencées par deux caractéristiques contradictoires : (a) vitesse de convergence, et (b) les fluctuations de l'état stationnaire causées par le bruit de l'algorithme.

- **Robustesse** : pour qu'un filtre adaptatif soit considéré robuste, les petites perturbations ne doivent causer que des petites erreurs d'estimation. Les perturbations peuvent avoir une origine interne ou externe au filtre.
- **Complexité de calcul** : ça comprend (a) le nombre d'opérations (multiplications, divisions et additions/soustractions) requis pour faire une itération complète de l'algorithme, (b) la taille de mémoire nécessaire pour stocker les données et le programme et (c) l'investissement requit pour programmer l'algorithme sur un ordinateur.
- **Structure** : ça représente la structure de la circulation de l'information dans l'algorithme, ce qui détermine la manière dont l'algorithme est implémenté sur le matériel. Par exemple, un algorithme qui a une structure qui permet une grande modularité, parallélisme ou concurrence est favorable pour une implémentation en technologie VLSI.
- **Propriétés numériques** : quand un algorithme est implémenté numériquement, les problèmes de précision sont causés par les erreurs de quantification. Ces erreurs sont dues à la conversion analogique-numérique et la représentation numérique des calculs internes. Habituellement, c'est la deuxième source d'erreurs de quantification qui cause de sérieux problèmes de conception. Particulièrement, on s'intéresse beaucoup plus à la stabilité

numérique et la précision numérique. La stabilité numérique est une caractéristique propre à l'algorithme de filtrage adaptatif. Quant à la précision numérique, elle est déterminée par le nombre de bits utilisé dans la représentation numérique d'échantillons de donnée et des coefficients du filtre. Un algorithme de filtrage adaptatif est dit robuste numériquement quand il insensible à la variation de la longueur de mots utilisée dans l'implémentation.

Les filtres adaptatifs peuvent être classifiés comme étant linéaires ou non linéaires. Le choix du type de filtre à prendre dépend de la nature du système adressé.

2.2 Filtres adaptatifs linéaires

Dans la Figure 2-1, un schéma bloc d'un filtre adaptatif linéaire est donné. Une séquence de données d'entrée $x(n)$ passe d'abord par un filtre transversal. Ensuite, en se basant sur le signal de l'erreur $e(n)$ entre la sortie du filtre $\hat{y}(n)$ et le signal désiré $y(n)$, on vient ajuster les coefficients $W(n)$ avec un algorithme de mise à jour des coefficients [1].

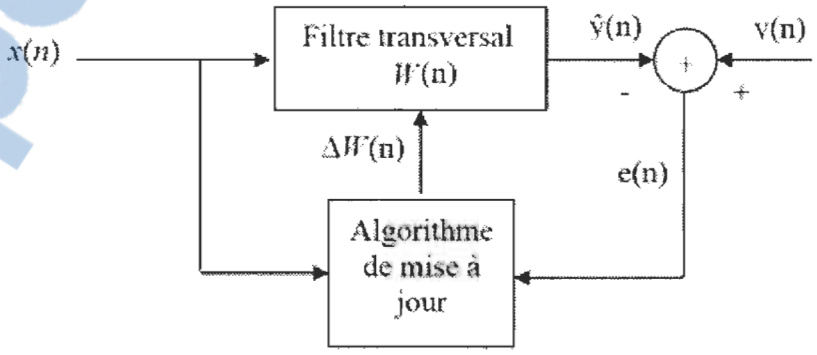


Figure 2-1 Filtre adaptatif linéaire

2.2.1 Méthode du gradient stochastique (LMS)

Le LMS est la méthode de filtrage adaptatif la plus populaire grâce à sa simplicité, sa faible complexité de calcul et son faible nombre de paramètres à ajuster.

Le LMS se déroule selon deux étapes principales :

1. *Passage par un filtre* : ce processus implique le passage des données d'entrée par un filtre considérant des poids avec une certaine valeur initiale

$$\hat{y}(n) = \hat{w}(n)x(n) \tag{2.1}$$

Ensuite, calculer l'erreur instantanée en faisant la différence entre le signal de sortir du filtre et la valeur désirée.

$$e(n) = y(n) - \hat{y}(n) \tag{2.2}$$

2. *Mise à jour des poids* : cette étape consiste à mettre à jour les poids du filtre en se basant sur l'erreur instantanée $e(n)$

$$\hat{w}(n+1) = \hat{w}(n) + \mu x^T(n)e(n) \tag{2.3}$$

Suivant le même principe de fonctionnement de la Figure 2-1, une séquence du vecteur d'entrée $x(n)$ est multipliée par un vecteur colonne dont les éléments sont les poids du filtre présenté comme suit : $\mathbf{w}(n) = \left[w_1(n) \;\; w_2(n) \;\; \cdots \;\; w_m(n)\right]^T$. La sortie résultante représente la valeur estimée du l'entrée désirée. Après, l'erreur qui résulte de la différence entre la sortie estimée $\hat{y}(n)$ et l'entrée désirée $y(n)$ est utilisée pour mettre à jour les coefficients du filtre de manière à minimiser l'erreur quadratique moyenne. Après un certain nombre d'itérations, l'erreur tombe plus ou moins à la valeur nulle. Le paramètre utilisé dans

l'équation de mise à jour du filtre μ est nommé pas d'adaptation du LMS. Le choix de ce paramètre représente un compromis entre erreur asymptotique, vitesse de convergence et stabilité. Sa valeur doit être choisie selon le contexte de l'application. Néanmoins, pour assurer la convergence du premier ordre, la valeur de μ doit vérifier l'équation suivante :

$$0 < \mu < \frac{2}{\lambda} \tag{2.4}$$

où λ représente l'auto corrélation des observations.

2.2.2 Méthode récursive des moindres carres (RLS)

La complexité du RLS dépasse de loin celle du LMS. Cependant, ses performances en matière de vitesse de convergence, erreur asymptotique et sensibilité aux valeurs propres du système sont bien supérieures comparé au LMS. À la différence du LMS, la mise à jour des coefficients du filtre utilise un gain $k(n)$ en fonction de l'itération instantanée :

$$\hat{\mathbf{w}}(n+1) = \hat{\mathbf{w}}(n) + k(n)e(n) \tag{2.5}$$

avec la formule de $k(n)$ donnée par :

$$k(n) = \frac{x^T(n)C(n)\lambda^{-1}}{1+\lambda^{-1}x(n)C(n)x^T(n)} \tag{2.6}$$

où $C(n)$ est la matrice de covariance donnée par :

$$C(n) = C(n)\lambda^{-1} - \lambda^{-1}k(n)x(n)C(n) \tag{2.7}$$

Le paramètre λ est appelé facteur d'oubli dont la valeur est comprise entre 0 et 1. La valeur initiale de la matrice de covariance est donnée par :

$$C(0) = \beta^{-1}I \tag{2.8}$$

avec I, représentant la matrice identité et β, une constante positive très faible.

2.3 Filtres adaptatifs non-linéaires

Dans un environnement non-linéaire, les non-linéarités peuvent êtres causées par divers effets. En plus de la nature intrinsèque du système, les circuits et les techniques impliquées dans le traitement et la mise en forme du signal (comme les capteurs, amplificateurs, convertisseurs A/D et D/A, processus de modulation/démodulation…etc.) représentent des sources potentielles de distorsions non-linéaires. Dans la suite du document nous présenterons deux techniques non-linéaires fondamentales.

2.3.1 Filtres adaptatifs non-linéaires à base de Volterra

Le développement en séries de Volterra d'une fonction est une généralisation des séries de Taylor avec mémoire [1]. Cette représentation est largement utilisée pour modéliser la relation entre l'entrée et la sortie d'un système non-linéaire. L'expansion en séries de Volterra tronquée à l'ordre P est donnée par [37]:

$$y[n] = h_0 + \sum_{m_1=0}^{N-1} h_1[m_1]x[n-m_1] + \sum_{m_1=0}^{N-1}\sum_{m_2=0}^{N-1} h_2[m_1,m_2]x[n-m_1]x[n-m_2] + \ldots + \sum_{m_1=0}^{N-1}\sum_{m_2=0}^{N-1}\ldots\sum_{m_p=0}^{N-1} h_p[m_1,m_2,\ldots,m_p]x[n-m_1]x[n-m_2]\ldots x[n-m_p] \quad (2.9)$$

où les paramètres h_P sont les coefficients de Volterra, généralement appelés Volterra kernels.

Le schéma bloc d'un filtre non-linéaire à base de Volterra est donné dans la Figure 2-2. Le système d'expansion de Volterra permet de réaliser les différentes combinaisons entre les

instances du signal d'entrée x qui compose le vecteur U. Un filtre adaptatif linéaire à réponse impulsionnelle finie (FIR – Finit Impulse Response) vient ensuite pour estimer les bons coefficients de Volterra en se basant sur le signal d'erreur entre la sortie du filtre et le signal désiré.

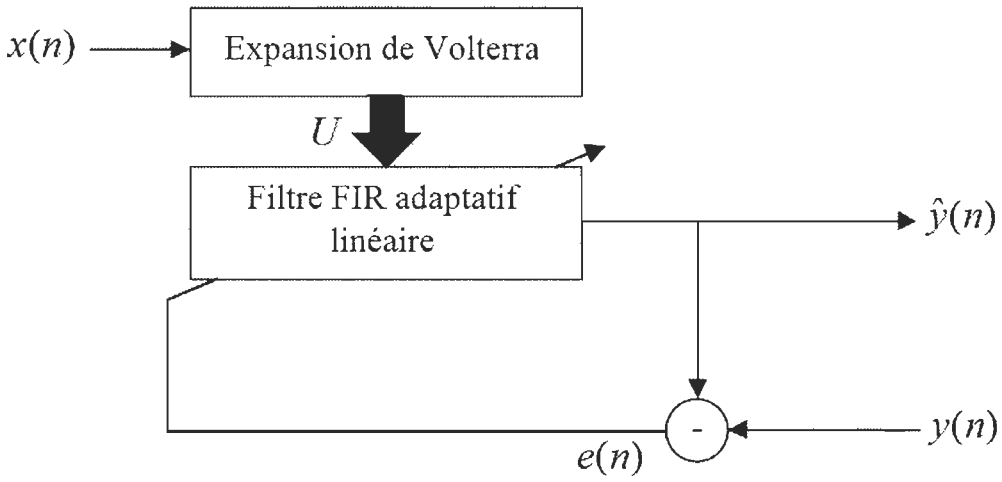


Figure 2-2 Filtre adaptatif non-linéaire de Volterra

2.3.2 Réseaux de neurones artificiels (ANNs)

Un réseau de neurones artificiel est un ensemble d'unités de traitement non-linéaires interconnectées, appelées neurones. Cette configuration permet de distribuer la non-linéarité à travers tout le réseau. Le développement de cette technique était inspiré du fonctionnement du cerveau humain, d'où son nom. Figure 2-3 présente les différents composants qui forment un neurone. Dans la première partie de l'opération, un circuit de somation calcule la somme pondérée résultante des produits entre les entrées du neurone et les poids synaptiques. Dans la deuxième partie, le résultat obtenu dans l'opération précédente est passé par une fonction d'activation pour générer le signal de sortie du

neurone. Les fonctions d'activation les plus utilisées dans la littérature ont la forme d'une sigmoïde ou une tangente hyperbolique.

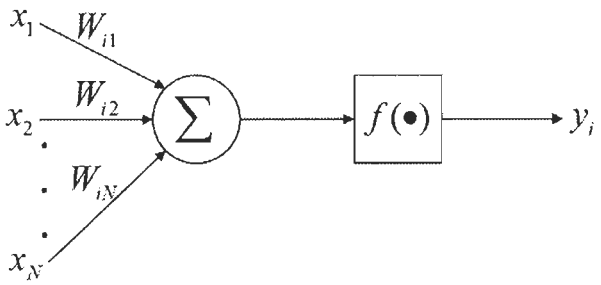


Figure 2-3 Composants d'un neurone

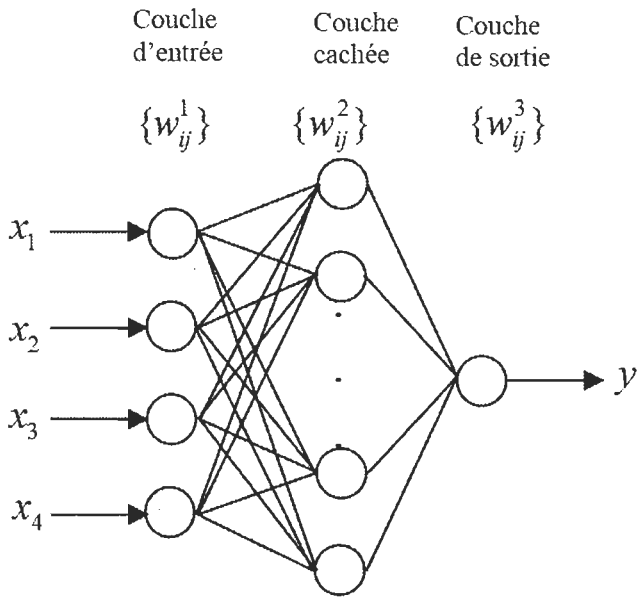


Figure 2-4 Réseau de neurones type MLP

La Figure 2-4 montre un exemple de réseau de neurones type perceptron multicouches (MLP – Multi-Layer Perceptron), qui est très populaire et utilisé dans les travaux de recherche. Le réseau est composé de trois couches : une couche d'entrée, une couche cachée et une couche de sortie. La difficulté consiste à trouver les valeurs optimales des

paramètres du réseau, qui sont nombreux (nombre de couches dans le réseau, nombre de neurones par couche, algorithme d'apprentissage, pas d'apprentissage, forme de la fonction d'activation...etc.).

2.4 Algorithmes génétiques

Les algorithmes génétiques sont des techniques d'optimisation utilisées pour résoudre des problèmes dans plusieurs disciplines. Leur fonctionnement est inspiré de la biologie, ou plus exactement du principe d'évolution des espèces. Autrement dit, de la théorie de Darwin. Les algorithmes génétiques sont devenus populaires à travers les travaux de John Holland dans les débuts des années 70 [38]. Des travaux sur l'optimisation de ces algorithmes se poursuivent jusqu'à présent.

Comme le filtrage adaptatif est un problème d'optimisation dont la solution est de trouver les coefficients optimaux du filtre, appliquer les algorithmes génétiques pour traiter ce problème peut avoir des avantages (complexité, stabilité, vitesse de convergence, non-linéarité...etc.) par rapport aux autres techniques conventionnelles.

2.4.1 Fonctionnement d'un algorithme génétique

Les opérations qui définissent le déroulement d'un algorithme génétique sont résumées dans l'organigramme suivant (Figure 2-5) :

- **Initialisation de la population** : la population est un ensemble d'individus appelés chromosomes. Ces chromosomes représentent les solutions possibles au problème adressé. Les chromosomes peuvent être codés en décimal, en binaire ou en toute autre représentation. Dans le cas du filtrage adaptatif, un chromosome n'est rien d'autre que les coefficients à optimiser.

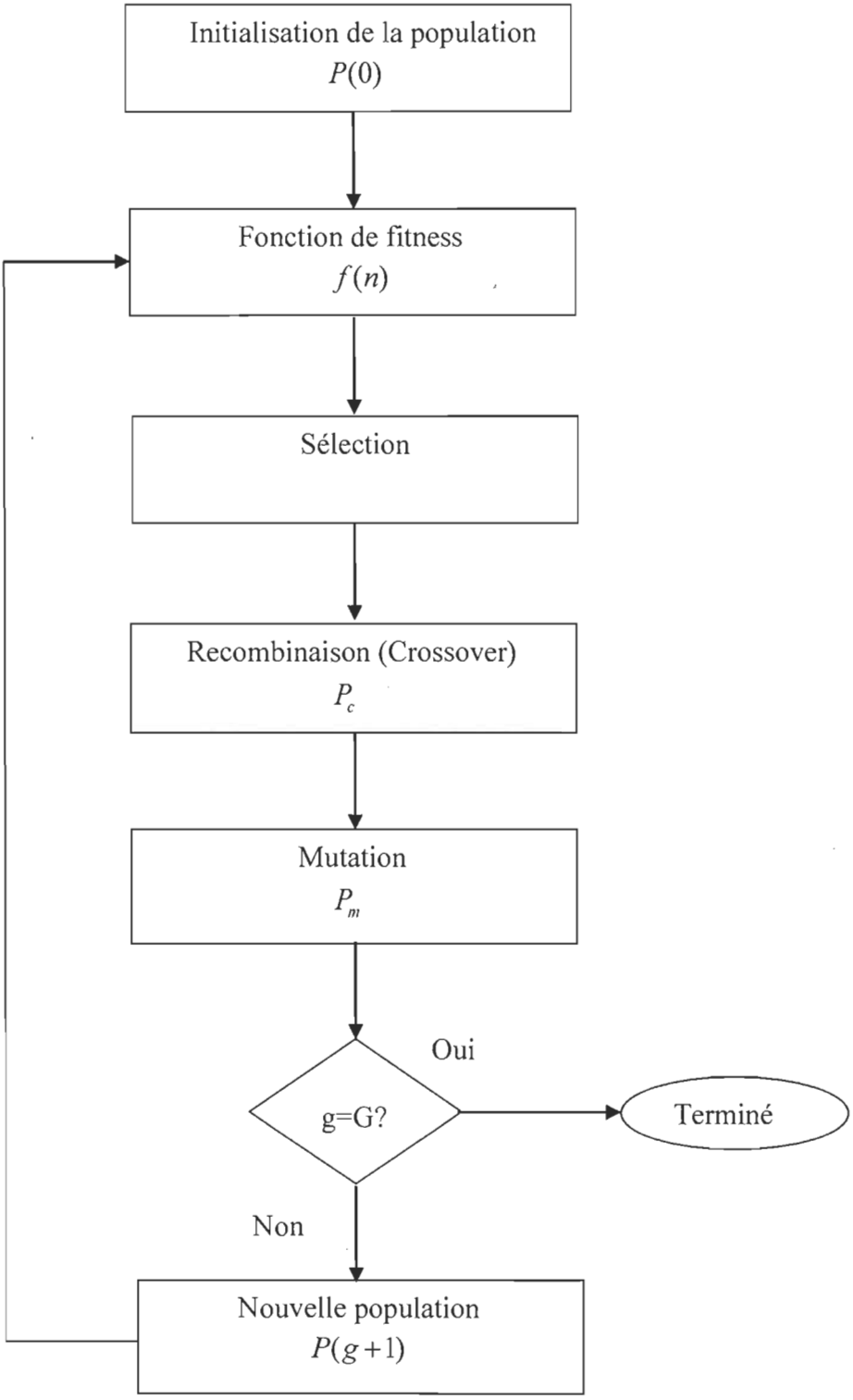
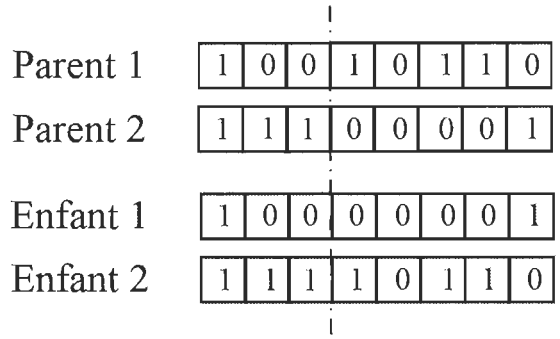


Figure 2-5 Algorithme génétique

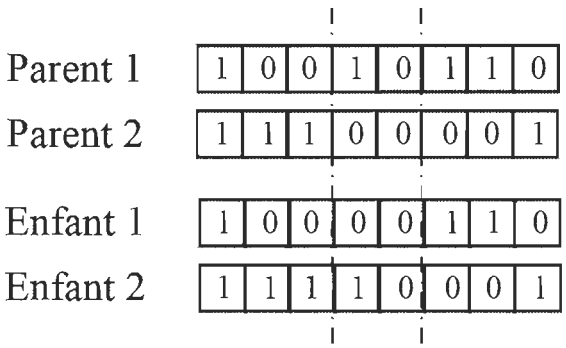
Au de Au début du processus, la population est initialisée d'une manière complètement aléatoire.

- **Application de la fonction de fitness** : aussi appelée fonction de coût, sa formule dépend essentiellement du problème à traiter. La fonction de fitness produit ce qu'on appelle un score de fitness qui nous informe sur la qualité d'un chromosome, ce paramètre nous permettra de classer les chromosomes du bon au mauvais. Cette fonction est la plus couteuse en matière de ressources dans tout l'algorithme.
- **Sélection** : l'opération de sélection permet le choix des chromosomes (parents) qui vont participer à l'opération de recombinaison. Les méthodes de sélection les plus utilisés dans la littérature sont :
  - La sélection par la roulette russe.
  - La sélection par troncature.
  - La sélection par échantillonnage stochastique universel.
- **Recombinaison** : l'opération de recombinaison consiste à faire une mixture entre deux chromosomes (parents) sélectionnés par l'opération précédente pour former des nouveaux chromosomes appelés « enfants ». Le nombre de chromosomes concernés par cette opération est déterminé par la probabilité de recombinaison P_c. La Figure 2-6 illustre deux types de recombinaison très utilisés. Pour la recombinaison à un seul point de croisement, le point est choisi de manière aléatoire, ce qui divise les chromosomes en deux segments. Ensuite, un des segments est échangé pour former deux nouveaux chromosomes (enfants). Pour la recombinaison à deux points de croisement, deux points

aléatoires sont utilisés. Les segments entre les deux points sont échangés pour former les chromosomes enfants.



(a)



(b)

Figure 2-6 Opération de recombinaison : (a) un seul point de croisement et (b) deux points de croisement

- **Mutation** : l'opération de mutation joue un rôle très important pour échapper au problème des minimums locaux. La mutation ne touche généralement qu'une petite portion du chromosome. Le nombre de chromosomes affectés par la mutation est déterminé par la probabilité de mutation P_m. La Figure 2-7 montre

un exemple d'opération de mutation. Un bit est choisi de manière aléatoire, puis inversé avec la fonction logique NOT. Le résultat forme le chromosome enfant.

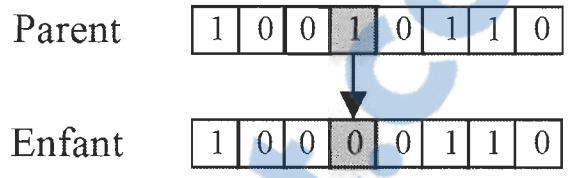


Figure 2-7 Opération de mutation

- **Terminaison** : plusieurs critères de terminaison de l'algorithme génétique peuvent être adoptés, comme le nombre de générations à faire ou une condition sur le score de fitness à vérifier.

2.4.2 Algorithmes génétiques et le calcul en virgule fixe

Avec tous les avantages que le calcul en virgule fixe apporte en matière d'implémentation sur matériel (surface d'intégration, vitesse de calcul, consommation d'énergie), étudier les performances d'un algorithme dans ce mode de calcul est primordiale pour trouver le bon compromis entre longueur de mots binaires employée et performances recherchées. Les AGs ont un avantage considérable dans cet environnement de calcul, grâce à leur possibilité de représenter et de manipuler de manière binaire les chromosomes.

Plusieurs propositions d'implémentations sur matériel (FPGA, ASIC) peuvent être trouvées dans la littérature. Des travaux proposent des architectures spécifiques à un problème donné [39], d'autres restent plus général en offrant des solutions reconfigurables et paramétrables (taille de la population, fonction de fitness, probabilité de

recombinaison...etc.) pour pouvoir adresser plusieurs problèmes d'optimisation avec une seule architecture [40, 41]. Ces propositions ont été utilisées pour résoudre différents problèmes comme le problème du voyageur de commerce et des équations mathématiques complexes.

En ce qui concerne le filtrage adaptatif en virgule fixe, très peu de travaux ont adressé cette problématique. Dans notre référence principale [24], les auteurs ont effectué une étude comparative entre un AG avec des opérateurs génétiques conventionnels, le LMS et le RLS. Différentes longueurs binaires ont été investiguées. Les résultats obtenus ont montré la capacité de l'AG à performer en utilisant des mots binaires avec une longueur de seulement 6-bits, là où les autres algorithmes ont échoué. Cependant, une contrainte limitant la recherche dans le voisinage immédiat des paramètres recherchés a été appliquée, ce qui n'est pas pratique.

Les travaux qui visent à implémenter un filtre adaptatif à base d'AG fonctionnant en temps réel sur du matériel parallèle (comme les FPGAs) sont quasi inexistants.

Chapitre 3 - Algorithmes génétiques à faible longueur binaire pour les applications du filtrage adaptatif: *identification de système*

3.1 Sommaire

Les algorithmes génétiques ont démontré leur efficacité à traiter des applications du filtrage adaptatif à travers des résultats issues de plusieurs travaux de recherche. L'avantage majeur de ces algorithmes par rapport aux algorithmes de filtrage adaptatif conventionnels, est la capacité d'adresser des problèmes non-linéaires. Un inconvénient majeur de cet achèvement est la grande quantité de calcul requise, ce qui peut être très contraignant quand des dispositifs à ressources matérielles limitées sont considérés.

Dans ce travail, nous proposons un algorithme génétique à faible complexité de calcul pour les applications du filtrage adaptatif. Des opérateurs génétiques sont spécialement conçus pour améliorer les performances de l'algorithme, tout en assurant une grande robustesse à l'environnement de calcul à très faible longueur binaire considéré. Cette pratique permet de réduire la surface d'intégration de l'algorithme.

L'algorithme développé est comparé avec le LMS et les performances théoriques, dans le contexte d'identification de paramètres d'un modèle ARMA linéaire et non-linéaire. Un environnement de calcul en virgule fixe est considéré, où les mots binaires ont une longueur de seulement 6 bits. Les critères de performances évalués sont le SQNR et le C_v (coefficient de variation). Dans la première partie de l'étude, les paramètres à identifier sont considérés fixes pendant les simulations. Pour le cas linéaire, l'AG présente une meilleure stabilité et un gain en SQNR de 2.5 dB dans les dernières itérations par rapport au LMS, et

0.5 dB moins que le SQNR théorique. Une faible dégradation dans le cas non-linéaire est observée, là où le LMS cesse de fonctionner. Dans la deuxième partie de l'étude, les paramètres à identifier varient aléatoirement d'une répétition à une autre. Dans le cas linéaire, l'AG maintient son fonctionnement avec une légère dégradation de la stabilité, et une baisse de 4.5 dB par rapport au SQNR théorique. Les résultats obtenus dans le cas non-linéaire sont identiques à ceux obtenus dans la première partie de l'étude.

Paper 1 **H. Merabti and D. Massicotte, " Towards Hardware Implementation of Genetic Algorithms for Adaptive Filtering Applications," in *IEEE Canadian Conference on Electrical and Computer Engineering (CCECE)*, Toronto, Canada, 4-7 May 2014.**

Towards Hardware Implementation of Genetic Algorithms for Adaptive Filtering Applications

Hocine MERABTI and Daniel MASSICOTTE

Université du Québec à Trois-Rivières, Department of Electrical and Computer Engineering

Laboratoire des Signaux et Systèmes Intégrés

E-mail: { Hocine.Merabti, Daniel.Massicotte }@uqtr.ca

3.2 Abstract

Genetic algorithms have been successfully applied to resolve adaptive filtering problems. The main advantage of using such algorithms over conventional adaptive filtering techniques, is their ability to deal with nonlinear systems. However, intensive computations are needed to achieve proper performances, which can be very critical when limited resources devices are considered for hardware implementation. This work proposes a low computation load genetic algorithm for adaptive filtering applications. Very low bit-wordlength fixed-point arithmetic is used in all operations to minimize the algorithm footprint. We compare the proposed method with the LMS and theoretical performances in identifying the parameters of an ARMA system. Simulation results show the high performances of the algorithm to deal with linear and nonlinear environments where only 6-bit wordlength is used.

3.3 Introduction

Genetic algorithms (GA) have been widely studied to address many adaptive filtering problems [1]-[4]. This motivation arises principally from the ability of genetic algorithms to find the optimal solution to the problem considering both linear and nonlinear environments. This represents a huge advantage compared to the popular adaptive filtering

algorithms like the least mean square (LMS) and the recursive least mean squares (RLS), which are limited to deal with linear systems only [5].

When it comes to hardware implementation, fixed-point arithmetic can have tremendous benefits over floating-point arithmetic (low power consumption, small area of integration and higher processing speed). This can be decisive when it comes to limited resources reprogrammable devices like FPGAs [6]. However, the computation precision should be carefully chosen, since low bit-wordlength can significantly decrease the signal processing characteristics (convergence, accuracy, asymptotic error and stability) due to high quantization errors.

In [7], the authors used a floating-point genetic algorithm to identify the parameters of a linear and a non-linear auto regressive moving average (ARMA) model. Interesting performances were achieved at the expense of extensive computation complexity due to the large chromosome population (500) and smoothing window size (200) employed. Always considering the same identification application, but in the fixed-point context [8], great results were obtained using very low wordlength computation. However, the genetic optimization was performed considering a small portion of the global search space. This could not be possible without the use of an additional technique that can provide a first estimation of the coefficients to identify, which causes additional overhead.

This paper deals with a specifically designed low computation load genetic algorithm for adaptive filtering applications, considering very low bit-wordlength processing environment. The main goal of the algorithm is to achieve high signal processing performances in term of convergence rate, asymptotic error and stability assuring the

convergence, while keeping the computation load low. This is realized by optimising genetic operators in order to reduce the population size and the smoothing window, without performances degradation. The studied application is parameters identification of an ARMA model in its linear and nonlinear forms. The GA is initially used to identify well defined system parameters. After that, a more general study covering the ability of the algorithm to identify random parameters is investigated. The performances of the proposed technique are compared to LMS and theoretical results.

In this paper, we expose the following points; a description of the proposed GA method is done in Section 2. Section 3 details the ARMA model parameters identification problem. In Section 4, simulation results for the LMS and the proposed GA are presented. Finally, we expose our conclusion in Section 5.

3.4 Adaptive Filter Based on GA

The genetic algorithm is basically a combination of five main operations: population initialization, fitness calculation, selection, crossover and mutation (Figure 3-1). Individuals forming the population are called chromosomes, each one represents the concatenated binary coded parameters, which in our case are the potential tap coefficients. The chromosome coding scheme case of tap coefficients wordlength, Q-bit, for 6-bit is as following:

w_1	w_2	$\cdots$	w_M
010010	111010	$\cdots$	010011

As a result, every parameter ranges in the interval [-1, $1 - 2^{-(Q-1)}$] (1 sign bit, and 5 fractional bits), leading to a chromosome searching space of

2^{QM} possibilities. The chromosomes change through successive iterations (sample n), called generations.

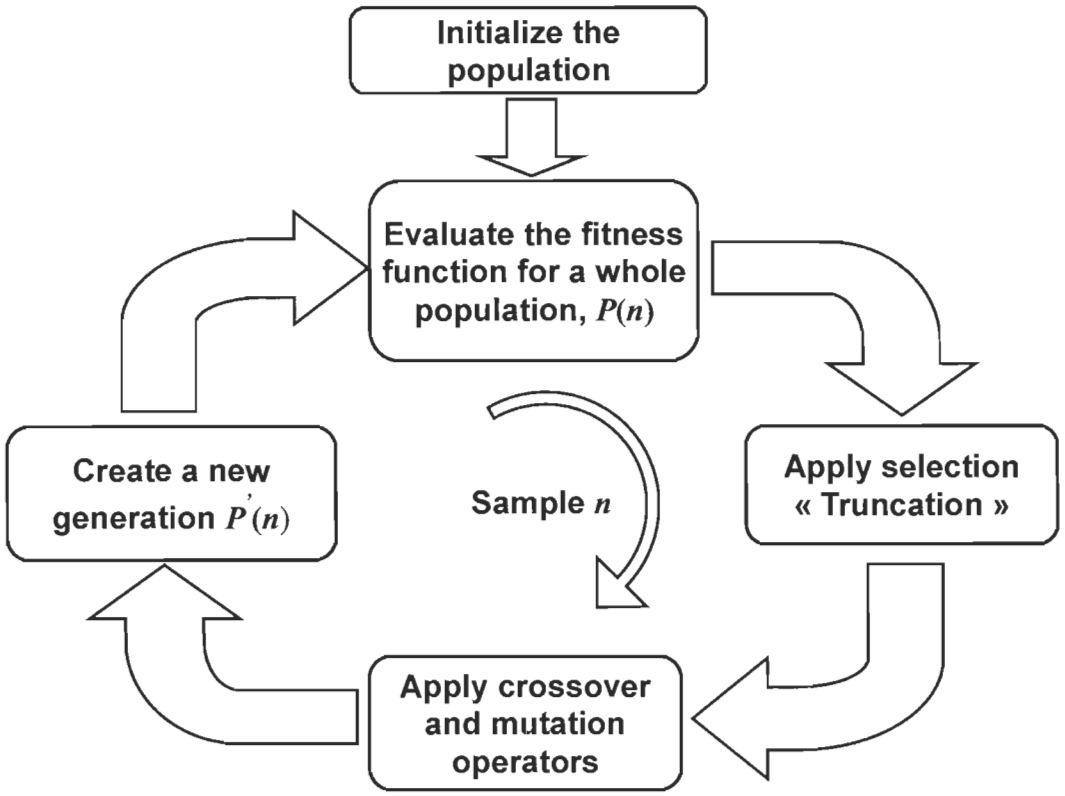


Figure 3-1 Main steps of genetic algorithm

In the beginning, every individual N_p of the population $P(n)$ is initialized in a random manner. Then, a fitness function $f(n)$ is applied to each chromosome, and a fitness score is thereby given to every individual. This function is composed of one or multiple criteria depending on the problem to optimize. The algorithm computation load is tightly related to the fitness function computation complexity, since it has to be applied to every

chromosome. After applying $f(n)$ on all chromosomes, a selection mechanism selects the fitter individuals among the population according to their fitness score.

The selected chromosomes will serve as a first pool of parents to the crossover operation, while the remaining chromosomes of the population form a second pool of parents. The selection method used in this work is based on the truncation selection technique. Its principle is to select a small portion of chromosomes with the highest fitness scores to form the first pool of parents. The choice of this method lies in its simplicity and ease of VLSI implementation. Next, one chromosome is picked up randomly from each pool. These parents are the chosen chromosomes for the crossing over with a probability P_c.

Adaptive crossover probability based on the fitness score can be used to enhance the convergence speed of the GA [1]. The idea behind this technique is to adjust automatically some parameters of the algorithm for best overall performances.

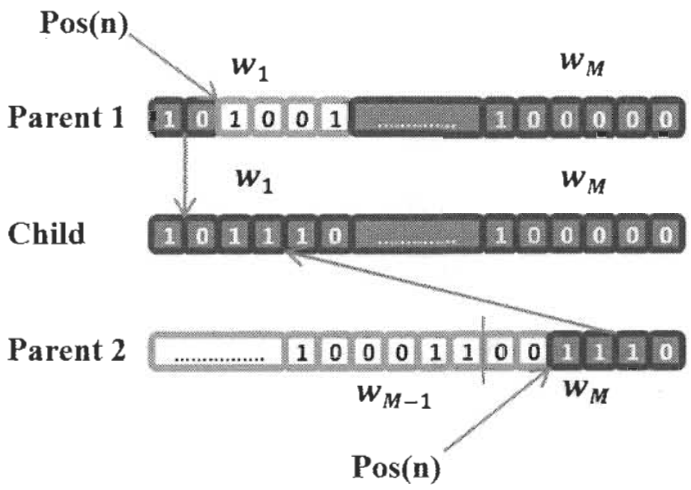


Figure 3-2 Proposed crossover operation

A new crossover approach is proposed in this paper, it is shown in Figure 3-2. First, the picked up parents are uniformly divided into M segments, representing the filter's

coefficients. Then, a single segment is selected randomly from each parent. After that, one point recombination is performed at the segment level. The position of the recombination point $Pos(n)$ is given in function of the best fitness score of the current population. The higher the fitness score is, the higher position of the crossover point is at the segment level. The function mapping fitness scores to crossover positions is totally linear. This move permits a guided research process instead of a totally random one. Improving results have been observed when compared to conventional crossover methods [7].

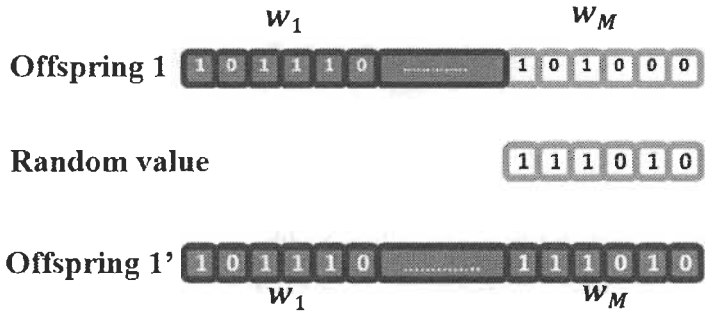


Figure 3-3 Mutation operation

The next step is the mutation operation, it is applied on chromosomes resulting from the previous operation (Offspring). Variable mutation probability $P_m(n)$ based on the fitness score criterion is used in this operation. The process is shown in Figure 3-3. One segment is selected randomly from an offspring, in the figure, segment corresponding to w_M is chosen. The selected segment is thereby replaced by a random generated bit value with equal bit wordlength as w_M.

Finally, a single cycle is then completed, and a new generation of population, $P'(n)$, is created.

The process (fitness calculation, selection, crossover and mutation) is repeated each generation n until the specified number of total generations, G is reached.

3.5 System Identification –ARMA Model

The performance study is done for an identification system described by the following ARMA model, Figure 3-4 [8]:

$$y_L(n) = \sum_{k=1}^{N_a} a_k y_L(n-k) + \sum_{k=0}^{N_b} b_k x(n-k) + \sum_{k=0}^{N_c} c_k u(n-k) \tag{3.1}$$

for linear system, and

$$y_{NL}(n) = tanh\left(v_{NL} y_L(n)\right) \tag{3.2}$$

for nonlinear system, where $y_L(n)$ is the output of the linear system; $y_{NL}(n)$ and v_{NL} are the output of the nonlinear system and the variance of the nonlinearity, respectively ; $x(n)$ is the input, $y(n)$ is the output and $u(n)$ is a noise signal. We suppose no correlation between $u(n)$ and the output signal. The input signal $x(n)$ is a random signal here, uncorrelated with $u(n)$. $u(n)$ is a white Gaussian noise with a zero mean and is 0.2 variance.

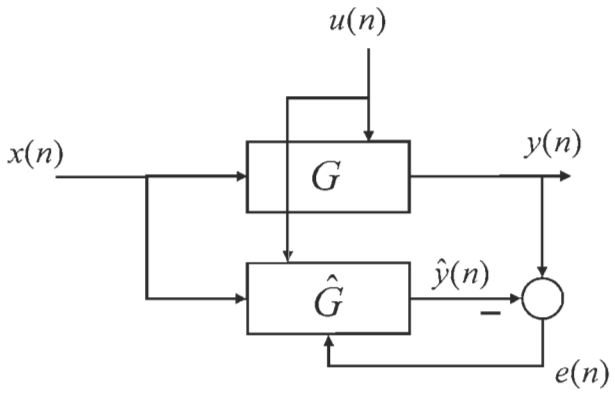


Figure 3-4Block diagram for the system identification

The problem consists to identify the system coefficients $\{a_k\}, \{b_k\}$ et $\{c_k\}$ using the system input and output signals. The model ARMA has non zero coefficients for $a_1, a_2, b_5, b_6, c_0, c_1, \text{and}, c_2$.

To identify the ARMA parameters, we defined the input vector

$\mathbf{v}(n) = \left[y(n-1), y(n-2), x(n-5), x(n-6), u(n), u(n-1), u(n-2)\right]^T$, $\dim(\mathbf{v}) = M \times 1$ with the following vector used to estimate ARMA parameters $\mathbf{w}(n) = \left[w_1(n) \;\; w_2(n) \;\; \cdots \;\; w_M(n)\right]^T$ where M defines the number or ARMA parameters to identify (M=7).

The LMS filter is defined as following:

$$\hat{y}(n) = \hat{\mathbf{w}}^T(n)\mathbf{v}(n) \quad \text{with } \hat{\mathbf{w}}(0) = \mathbf{0} \tag{3.3}$$

$$e(n) = y(n) - \hat{y}(n) \tag{3.4}$$

$$\hat{\mathbf{w}}(n+1) = \hat{\mathbf{w}}(n) + \mu\mathbf{v}(n)e(n) \tag{3.5}$$

For the GA, described in Section 2, we defined the fitness function method as:

$$f(n) = \sum_{m=0}^{B-1} \left| y(n-m) - \hat{y}(n-m) \right| \tag{3.6}$$

where, B is the block size of data over which we evaluate the filter and calculate the absolute error. Very interesting performances were achieved, as presented in the next section.

3.6 Simulation Results

Initially, we consider the system identification problem where the ARMA model has well defined coefficients corresponding to $a_1 = 1.5, a_2 = -0.7, b_5 = 1, b_6 = 0.5,$ $c_0 = 1, c_1 = -1, c_2 = 0.2$ [8]. In the non-linear case, a strong nonlinearity variance $v_{NL} = 2$ is used. The performances of fixed-point LMS are presented for different wordlengths. The step-size parameter (μ) is selected for best tradeoff between steady-state error and convergence speed. For all simulations, the crossover probability P_c was fixed at 100% during all the iterations. Small population size of $N_P = 16$ and a fitness window of B=16 are used. For both adaptive methods, all operations and signal are quantized to Q-bit wordlength in a normalized fixed-point arithmetic:1-bit was reserved for the sign and the remaining bits for the fractional part. The simulation results were obtained by averaging 100 independent realizations (repetitions).

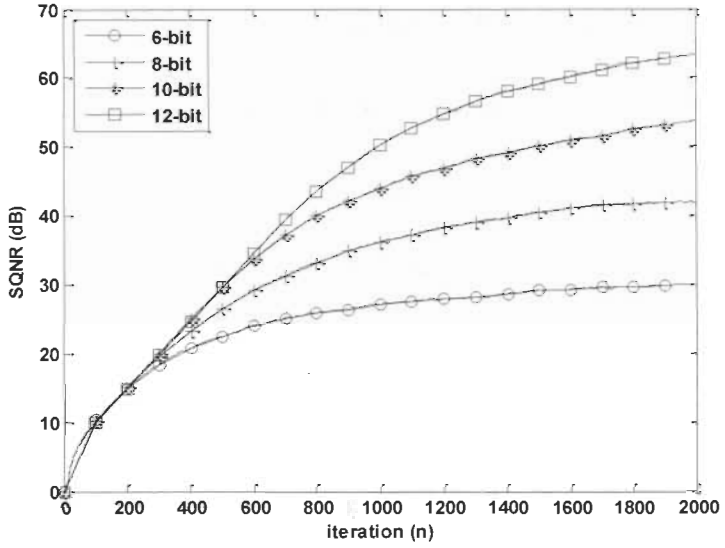
The comparative study is done based on the Signal to Quantization Noise Ratio (SQNR) for accuracy metric

$$SQNR(n) = 10\log\left(\sum_{i=1}^{M} w_i^2 \Big/ \sum_{i=1}^{M}\left(w_i - \hat{w}_i(n)\right)^2\right)(\text{dB}) \tag{3.7}$$

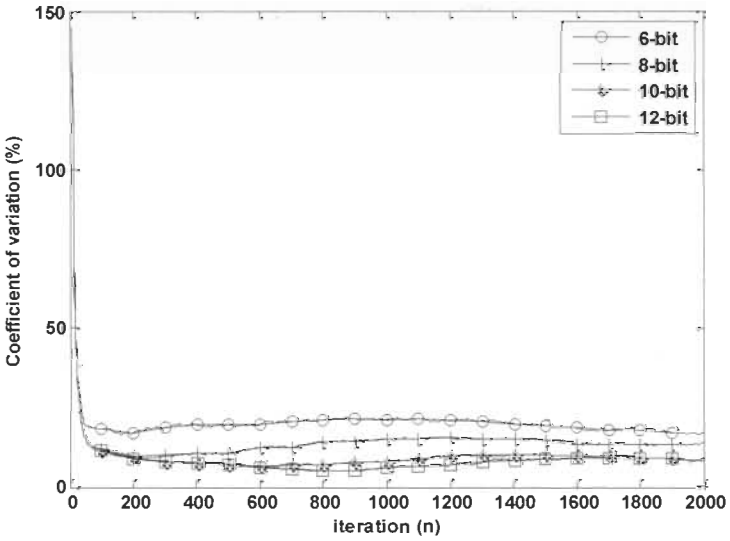
where w_i is the exact coefficient and $\hat{w}_i$ is the estimated coefficient. To measure the convergence stability, the repeatability of the adaptive methods using the coefficient of variation C_v defined as

$$C_v(n) = \frac{\sqrt{\frac{1}{R-1}\sum_{i=1}^{R}\left(SQNR(n) - \frac{1}{R}\sum_{i=1}^{R} SQNR(n)\right)^2}}{\frac{1}{R}\sum_{i=1}^{R} SQNR(n)} \times 100\,(\%) \tag{3.8}$$

where R is the number of repetition.



a)



b)

Figure 3-5 Linear system identification for different wordlenghts using LMS: a) SQNR and b) C_v

Figure 3-5 shows the simulation results of the LMS algorithm for the linear system identification problem. Curves corresponding to Q5, Q7, Q9 and Q11 formats are

presented. As expected from the LMS, the algorithm is performing well in identifying the linear system, the SQNR goes up from 30 dB in the last iterations with a corresponding C_v of 17% , when a Q5 scheme is adopted, to 63 dB with a C_v of 9% for the Q11 format. As provided, the evolution of the SQNR after convergence in function of wordlength is quasilinear.

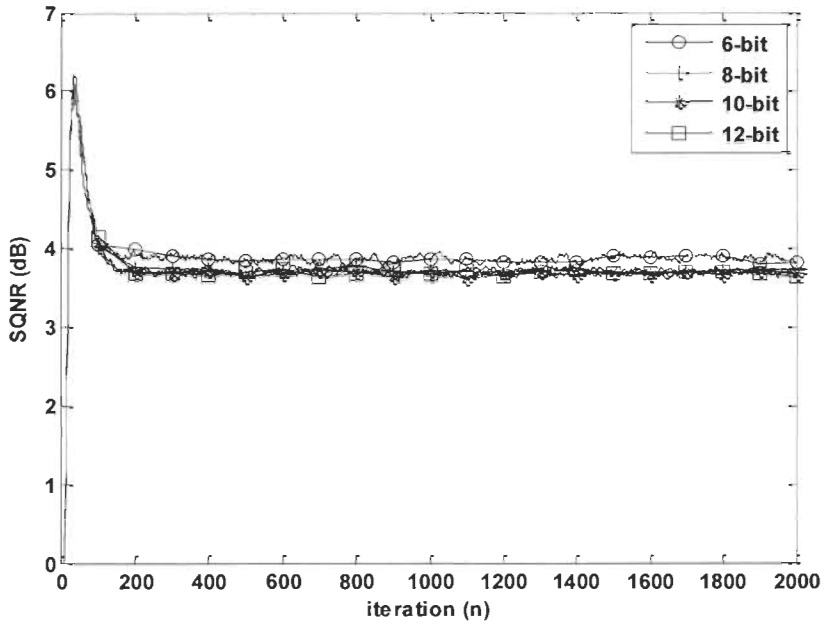
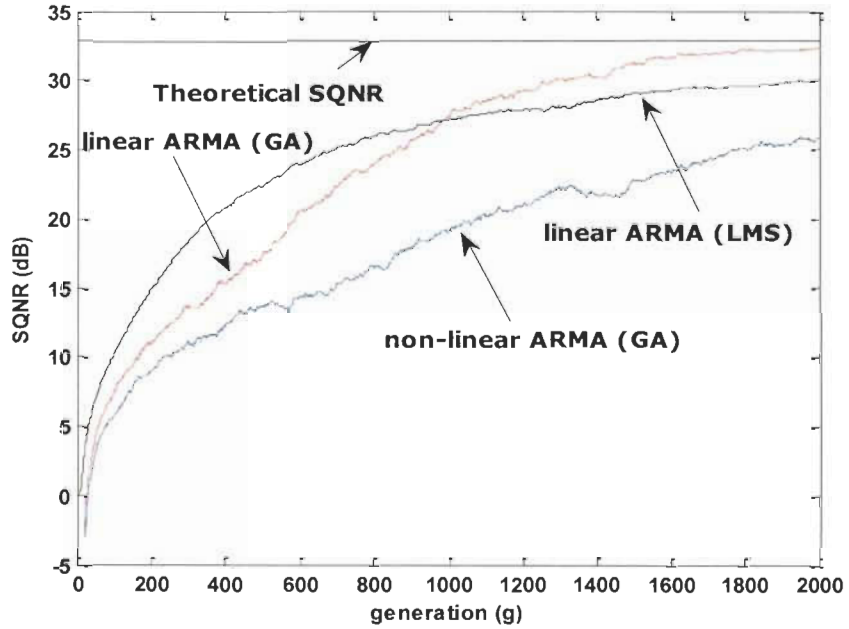
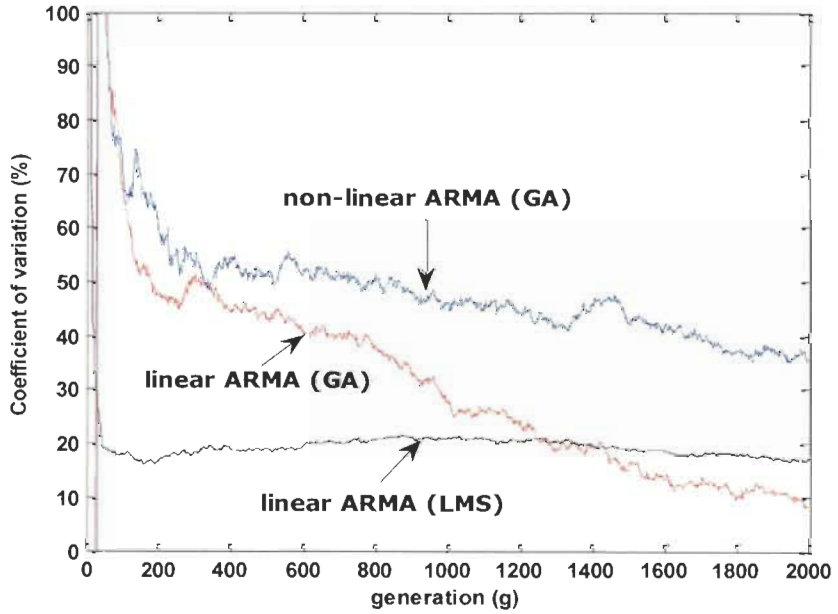


Figure 3-6 SQNR of nonlinear system identification for different wordlengths using LMS

Figure 3-6 demonstrates the inability of the LMS algorithm to deal with non-linear systems. Very poor results with no sign of convergence are observed for all the different wordlenghts.



a)



b)

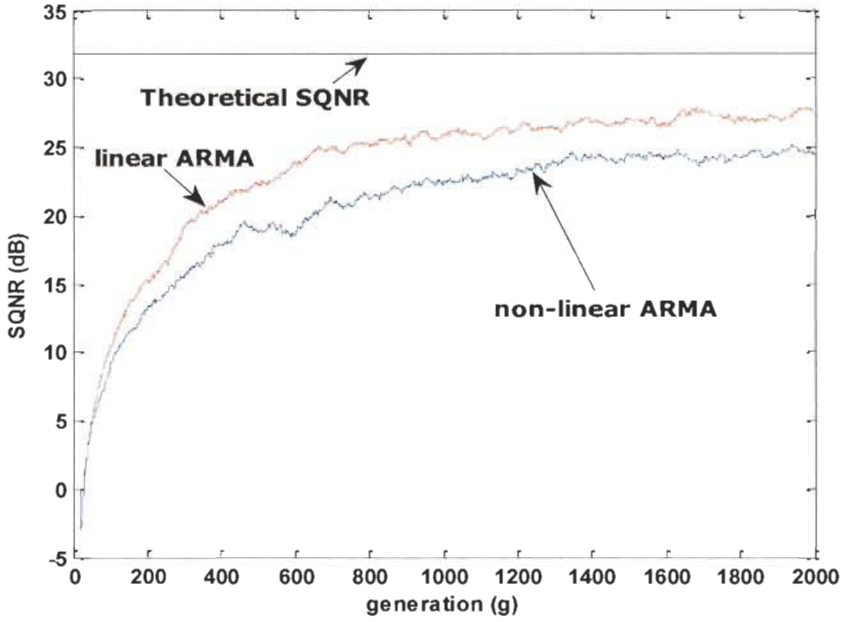
Figure 3-7 6-bit wordlength system identification using the GA: a) SQNR and b) C_v

The performances obtained by the GA presented in Section 2 are shown in Figure 3-7 when using the Q5 format. The theoretical SQNR is calculated using the perfect coefficients on Q-bit quantization. It has a value of 33 dB. The GA achieved an SQNR of 32.5 dB in the case of the linear system identification problem. This is 2.5 dB more than LMS and 0.5 dB close to the theoretical SQNR. A very low coefficient of variation reaching 9% in the last generations was observed. This demonstrates high stability considering the random nature of GA.

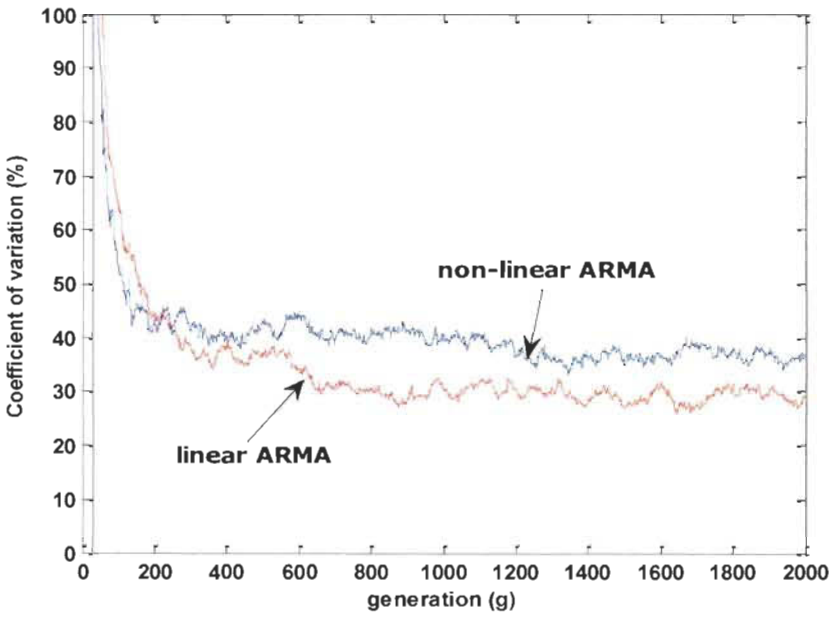
The ability of the GA to deal with the nonlinear system is also presented in Figure 3-7 b). We can observe an SQNR of 26 dB at 2000 generations, with a corresponding coefficient of variation of 36%. This is pretty good considering the strong nonlinearity assumed in this study.

The next step is to study a more realistic scenario. The coefficients to be identified are generated randomly at each realization in the interval [0,1[for a_1, b_5, b_6, c_0, c_2, and [-1,0] for a_2, c_1. This will test the ability of the GA to perform regardless the values of the coefficients to identify.

Figure 3-8 shows the simulation results for the Q5 format. The average of theoretical SQNR is 32 dB. In linear system identification, the GA presents an SQNR of 27.5 dB and a coefficient of variation of 29% in the last iterations. This represents 4.5 dB less than the theoretical SQNR. In the nonlinear case, an SQNR around 25 dB and a coefficient of variation of 38% were observed. Compared to the first scenario where well defined coefficients have to be identified, a degradation of 4 dB is observed in the linear identification study, while no significant change in the nonlinear case. This is very interesting, considering random coefficients to identify.



a)



b)

Figure 3-8 6-bit wordlength random coefficients identification using the GA: a) SQNR and b) C_v

3.7 Conclusion

Low computation load genetic algorithm considering very low bit-wordlength fixed-point arithmetic environnemt has been proposed. We have studied the performances of the algorithm in identifying well defined and random parameters of an ARMA model in both linear and non-linear cases. The proposed GA demostrated a better SQNR and coefficient of variation than the conventional LMS in the linear case. In nonlinear identification, tha GA kept performing with a slight degradation, while the LMS did not show any sign of convergence. The futur work will focus on invistigating different adaptive filtering applications and FPGA implementation.

3.8 Acknowledgement

The authors wish to thank the National Sciences and Engineering Research Council of Canada (NSERC) for financial support and Compute Canada for parallel equipment support.

3.9 References

[1] Cheng-Yuan, C. and C. Deng-Rui, "Active Noise Cancellation Without Secondary Path Identification by Using an Adaptive Genetic Algorithm," *IEEE Trans. on Instrumentation and Measurement*, 59(9), 2010, pp. 2315-2327.

[2] Amiri, N. K. and S. M. Fakhraie, "Digital Network Echo Cancellation Using Genetic Algorithm and Combined GA-LMS Method," *IEEE Pacific Conference on Circuits and Systems*, APCCAS 2006.

[3] S.C. Ng, S.H. Leung, C.Y. Chung, A. Luk, W.H. Lau, "The Genetic Search Approach A New Learning Algorithm For Adaptive IIR Filtering", *IEEE Signal Processing Magazine*, vol. 13, no. 6, 1996, pp. 38-46.

[4] F. Russo and G. L. Sicuranza, "Accuracy and Performance Evaluation in the Genetic Optimization of Nonlinear Systems for Active Noise Control," *IEEE Trans. on Instrumentation and Measurement*, 56(4), 2007, pp. 1443-1450.

[5] S. Haykin, Adaptive Filter Theory, Prentice-Hall, 1996.

[6] Jihong, L. and L. Deqin (2005). "A Survey of FPGA-Based Hardware Implementation of ANNs," *Int. Conf. on Neural Networks and Brain (ICNNB)*, 2005.

[7] V. Duong and A. R. Stubberud, "System identification by genetic algorithm," in *IEEE Aerospace Conference Proceedings*, 2002, pp. 5-2331-*5-2337 vol.5.*

[8] D. Massicotte, D. Eke, "High Robustness to Quantification Effect of an Adaptive Filter based on Genetic Algorithm," *IEEE Int. Conference on Circuit and System (NEWCAS),* Montreal, 2007, pp. 373-376.

Chapitre 4 - Algorithmes génétiques à faible longueur binaire pour les applications du filtrage adaptatif: *égalisation de canaux de communication non-linéaires*

4.1 Sommaire

L'égalisation de canaux de communication non-linéaires a été au centre de plusieurs travaux de recherche ces dernières années. Les sources de ces non-linéarités sont diverses, elles peuvent provenir du canal lui-même, ou encore causées par les amplificateurs, convertisseur, diaphonie et processus de modulation/démodulation. Ce type de canaux est rencontré dans plusieurs systèmes de communication dans la pratique, notamment dans les transmissions via liaisons satellitaires.

Les ANNs figurent parmi les solutions potentielles pour traiter ce problème, grâce à leur capacité de modéliser efficacement des systèmes non-linéaires. L'égaliseur à base de CDFRNN a fait l'objet de plusieurs études. Lorsque ce réseau de neurones est entrainé par un algorithme suiveur de gradient comme le CRTRL (*Complex Real Time Recurrent Learning*), l'équaliseur présente une faible vitesse de convergence, nécessitant une longue séquence de symboles d'entrainement pour atteindre des performances satisfaisantes.

Dans ce travail nous proposons une nouvelle approche pour égaliser un canal non-linéaire en utilisant les AGs. Le VDFGA proposé utilise une modélisation en séries de Volterra dans une configuration à retour de décision, afin de faire face aux non-linéarités ainsi qu'aux fortes ISI du canal considérées dans l'étude. Le but de la technique élaborée est d'exploiter la vitesse de recherche élevée de l'AG pour estimer les bons coefficients du

filtre, avec un minimum de symboles pilotes. Une représentation en virgule fixe à très faible longueur binaire est adoptée pour les coefficients.

La méthode proposée est comparée avec un égaliseur à base de CDFRNN. Ce dernier est entrainé avec le CRTRL, qui est totalement implémenté en virgule flottante, tandis que les coefficients du filtre à base d'AG sont codés sur 8-bits. La technique proposée offre une très grande vitesse de convergence avec un MSE qui atteint le régime stationnaire en moins de 300 itérations, comparée à la méthode de référence qui continue de converger lentement même pendant les dernières itérations observées. Cette vitesse de convergence permit d'utiliser peu de symboles d'entrainement. Le BER obtenu avec la méthode développée atteint 10^{-2} pour un SNR de 24 dB, tandis que l'algorithme de référence atteint un BER de 3.5×10^{-2}.

Paper 2 **H. Merabti and D. Massicotte, " Nonlinear Adaptive Channel Equalization using Genetic Algorithms," in *IEEE International New Circuits and Systems (NEWCAS)*, Trois-Rivieres, Canada, 22-25 June 2014.** (Best Student Paper Award nominee)

Nonlinear Adaptive Channel Equalization using Genetic Algorithms

Hocine MERABTI and Daniel MASSICOTTE

Université du Québec à Trois-Rivières, Department of Electrical and Computer Engineering

Laboratoire des Signaux et Systèmes Intégrés

E-mail: { Hocine.Merabti, Daniel.Massicotte }@uqtr.ca

4.2 Abstract

Nonlinear adaptive channel equalization is a well-documented problem. Equalizers based on the complex decision feedback recurrent neural network (CDFRNN) have been intensively studied to address this problem. However, when trained with conventional training algorithms like the real time recurrent learning (RTRL) technique, the equalizer suffers from low convergence speed, requiring very long training sequence to achieve proper performance. In this work, we propose a new approach to equalize nonlinear channels using genetic algorithms. The proposed Volterra decision feedback genetic algorithm (VDFGA) uses a genetic optimization strategy to estimate Volterra kernels in order to model the inverse of the channel response. Simulation results show very high convergence speed, which allowed to achieve interesting bit error rate (BER) using relatively short training symbols, when considering only 8-bits long coded weights.

4.3 Introduction

Nonlinearities in communication systems can be the result of several effects. Besides the communication channel nature itself, unwanted distortions coming from amplifiers and converters, crosstalk interference and modulation/demodulation process are common causes for nonlinearities. Equalizing communication channels with nonlinear distortions is

a well-known problem, it has been the subject of many research works in the last decades [1]-[3]. The problem becomes more difficult to treat when the intersymbol interference (ISI) gets severe. In addition, using complex modulation schemes like QAM makes the equalization tougher, due to their sensitivity to nonlinearities [4].

In adaptive equalization, artificial neural networks (ANNs) represent a potential solution to this problem, they have been in the center of many studies due to their ability to effectively model nonlinear systems, which permits high equalization performances compared to linear equalizers (LEs) that use conventional adaptive filtering algorithms like the least mean square (LMS) and the recursive least square (RLS) [5]. It was demonstrated that a recurrent neural network (RNN) can achieve better results than a multilayer perceptron (MLP) neural network with less neurons [1]. The most used algorithm for training RNNs is the RTRL algorithm [6], it was introduced by Williams and Zipser [7]. However, the main drawback of the RTRL algorithm is its slow convergence speed, requiring the use of very long training sequences to achieve proper equalization, which can be very critical in some applications. To overcome this limitation, and to offer high tracking capability for time varying channels, the authors in [8] used the extended Kalman (EKF) and the unscented Kalman (UKF) to train the RNN. Very interesting performances were achieved at the expense of extensive computation load.

Recently, many studies have investigated the use of genetic algorithms (GAs) to address nonlinear adaptive filtering applications. They have been successfully applied to system identification [9] and active noise control [10]. The interest in this class of algorithms is driven by their ability to treat nonlinear problems and provide high convergence speed. They can also operate using very low bit wordlength in fixed point

arithmetic, thanks to the ability to adapt their weights using natural updating mechanism, which does not require arithmetic operations [11], making it suitable for implementations.

In this work, we present a new approach to equalize nonlinear communication channels. An original adaptive algorithm named Volterra decision feedback genetic algorithm (VDFGA) is proposed. The algorithm uses Volterra series expansion of the filter input along with decision feedback to provide robustness to channels with high nonlinearities and large ISI. A GA is then used to estimate the Volterra kernels and the feedback weights by minimizing a cost function. The technique is compared to the well-known RTRL trained CDFRNN algorithm [4], in the sight of equalizing a nonlinear communication channel considering 4- QAM modulation. High convergence speed and stability of the proposed approach are observed using fixed point representation of the weights, which is an inner feature of the GA.

In this paper, we expose the following points: the channel equalization problem and the reference technique are presented in Section 2. Section 3 discloses the GA based proposed algorithm. In Section 4, simulation results and discussions are given. Finally, we expose our conclusion in Section 5.

4.4 Channel Equalization Problem and the CDFRNN

4.4.1 Adaptive channel equalization

A typical block diagram of an adaptive equalizer for discrete nonlinear dispersive channels is presented in Figure 4-1 [4].

Assume 4-QAM modulation scheme, $s(n) = I(n) + jQ(n)$ denotes the transmitted symbol at the nth instant, I and Q are considered uniformly distributed random variables taking values over $\{+1, -1\}$.

$c(n)$ is the channel impulse response, and $\eta(n)$ is an additive white Gaussian noise (AWGN) with a zero mean and variance σ^2. The equalizer input signal $x(n)$ is expressed as:

$$x(n) = g\left(\sum_{i=0}^{L-1} s(n-i)c(i)\right) + \eta(n) \tag{4.1}$$

where L is the channel length and $g(\bullet)$ is a nonlinear function. The equalizer operates by minimizing the error signal $e(n)$ between its output and the desired transmitted symbol $s(n-D)$, by adaptively altering the filter tap weights $W(n)$. D is a delay used to compensate for the time needed by symbols to pass through the system.

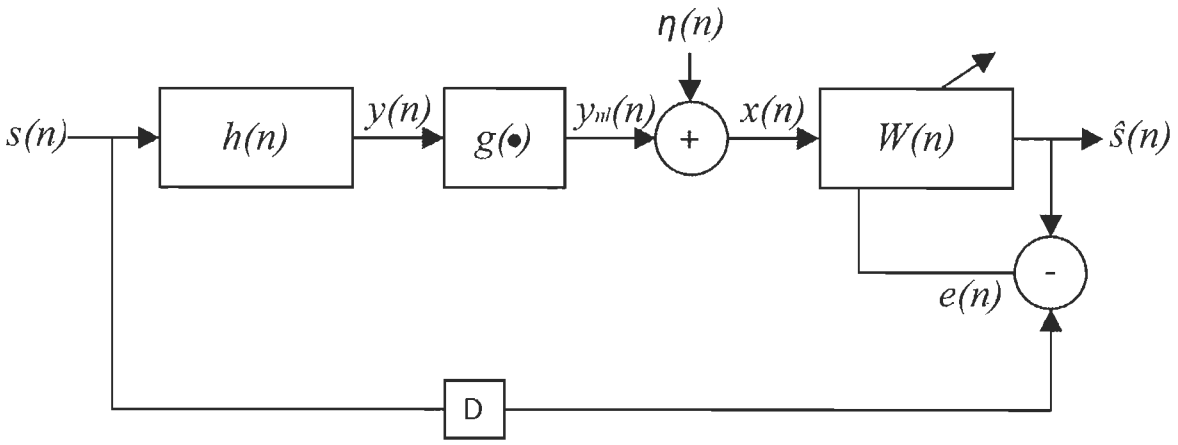


Figure 4-1 Adaptive channel equalization block diagram

4.4.2 Neural network training

The architecture of a typical fully interconnected CDFRNN is presented in Figure 4-2. The structure is composed of N neurons, M external inputs, and N feedback signals. The output signals are passed through a decision device and a delay circuit before getting injected into the network. Two phases are needed to train the neural network using the Complex RTRL (CRTRL) algorithm, they are described by the following equations [6]

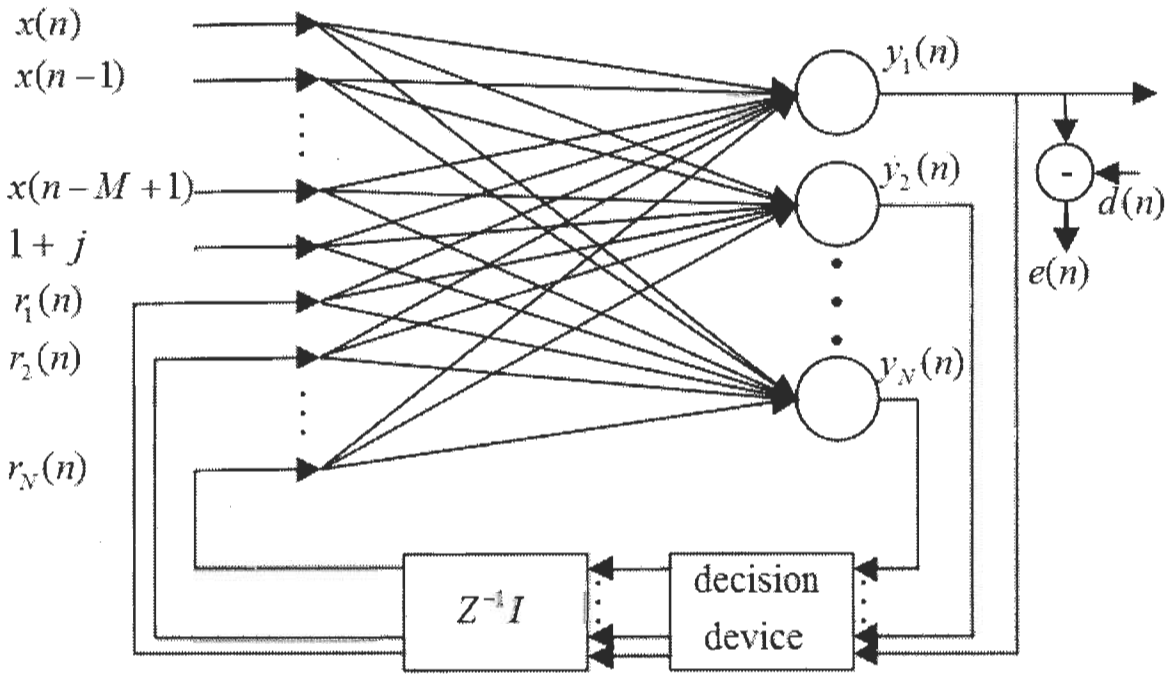


Figure 4-2 Architecture of the CDFRNN

4.4.2.1 Feedforward phase

Consider the input vector $V(n)$ described as

$$V(n) = \left[x(n), \ldots, x(n-M+1), 1+j, r_1(n), \ldots, r_N(n)\right] \tag{4.2}$$

The neurons output are given by

$$u_k(n) = u_k^R(n) + ju_k^I = \sum_{j=1}^{N+M+1} V_i(n) W_{ij}(n) \tag{4.3}$$

$$y_k(n) = f(u_k^R(n)) + jf(u_k^I) \tag{4.4}$$

where W_{ij} is the weight on the connexion of the ith neuron and the jth input , and $f(\bullet)$ is a nonlinear activation function, usually a hyperbolic tangent function.

4.4.2.2 Learning phase

The recurrent expression for the sensitivity terms P^k_{RRij}, P^k_{RIij}, P^k_{IRij}, P^k_{IIij} are given below. Indexes range over $\{k = 1, ..., N; i = 1, ..., N; j = 1, ..., N + M + 1\}$.

$$\begin{pmatrix} P^k_{RRij} & P^k_{RIij} \\ P^k_{IRij} & P^k_{IIij} \end{pmatrix}(n) = \begin{pmatrix} f'(u_k^R) & 0 \\ 0 & f'(u_k^I) \end{pmatrix}(n) \times \left(\sum_{l=1}^{N} \begin{pmatrix} W^R_{kl} & -W^I_{kl} \\ W^I_{kl} & W^R_{kl} \end{pmatrix} \begin{pmatrix} P^l_{RRij} & P^l_{RIij} \\ P^l_{IRij} & P^l_{IIij} \end{pmatrix}(n) + \delta_{ik} \begin{pmatrix} v_j^R & -v_j^I \\ v_j^I & v_j^R \end{pmatrix} \right)(n) \tag{4.5}$$

where $f'(\bullet)$ is the derivative of the activation function, and δ_{ik} is the Kronecker delta.

The weight updating equation is expressed as

$$W_{ij}(n+1) = W_{ij}(n) + \mu \sum_{k=1}^{N} \left(\begin{bmatrix} e_k^R & e_k^I \end{bmatrix} \begin{pmatrix} P^k_{RRij} & P^k_{RIij} \\ P^k_{IRij} & P^k_{IIij} \end{pmatrix} \begin{bmatrix} 1 \\ j \end{bmatrix}(n) \right) \tag{4.6}$$

μ is the learning rate with strictly positive value. The sensitivity matrix is initialized with all-zeros elements, and the weights matrix (W_{ij}) with very small random values.

4.5 Proposed VDFGA Equalizer

The block diagram of the proposed Volterra decision feedback genetic algorithm equalizer is presented in Figure 4-3. It combines three different techniques:

1. a block that performs truncated Volterra series expansion of the input signal to allow reliable approximation of the inverse of the channel response,
2. a decision feedback structure that improves the equalizer robustness to large ISI, and
3. a genetic algorithm to adaptively estimate Volterra kernels and the weights associated with the feedback signals.

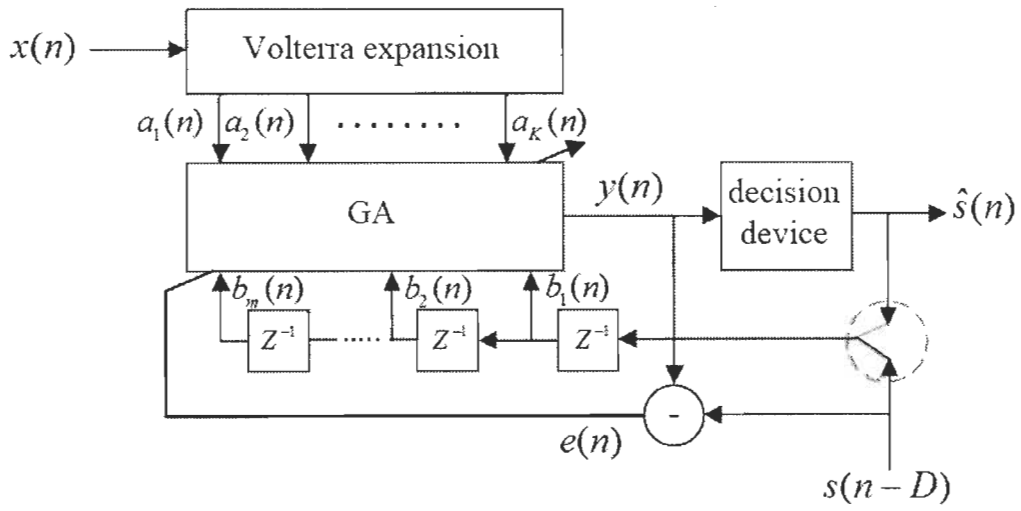


Figure 4-3 Block diagram of the VDFGA

A second order truncated Volterra series expansion has the following form

$$y_V(n) = h_0 + \sum_{u_1=0}^{N-1} h_1[u_1]x[n-u_1] + \sum_{u_1=0}^{N-1}\sum_{u_2=0}^{N-1} h_2[u_1,u_2]x[n-u_1]x[n-u_2] \tag{4.7}$$

After eliminating the offset component and duplicated terms, the expression can be reduced to

$$y_V(n) = A(n)H_V^T \tag{4.8}$$

where A and H_V are the input and the weight vectors, respectively, which are defined as

$$A(n) = [x(n), x(n-1), x(n-2), x^2(n), x(n)x(n-1), x(n)x(n-2), x^2(n-1), x(n-1)x(n-2), x^2(n-2)] \tag{4.9}$$

$$H_V = [h_1[0], h_1[1], h_1[2], h_2[0,0], h_2[0,1], h_2[0,2], h_2[1,1], h_2[1,2], h_2[2,2]\ ] \tag{4.10}$$

The decision feedback component signal is represented by the following equations

$$y_{DF}(n) = B(n)H_{DF}^T(n) \tag{4.11}$$

where B and H_{DF} are expressed as

$$B(n) = [s(n-D-1), s(n-D-2), ..., s(n-D-m)] \tag{4.12}$$

during the training phase,

$$B(n) = [\hat{s}(n-1), \hat{s}(n-2), ..., \hat{s}(n-m)] \tag{4.13}$$

during the estimation phase, and

$$H_{DF}(n) = [h_1(n), h_2(n), ..., h_m(n)] \tag{4.14}$$

It is sufficient to choose a feedback order m following [8]

$$m = L + K - D - 2 \tag{4.15}$$

where K represents the size of vector H_V

The total output signal y is given by

$$y(n) = y_V(n) + y_{DF}(n) \tag{4.16}$$

The error signal used to update the filter weights is expressed as

$$e(n) = y(n) - s(n - D) \tag{4.17}$$

The operation diagram of the genetic algorithm used to estimate H_V and H_{DF} is presented in Figure 4-4.

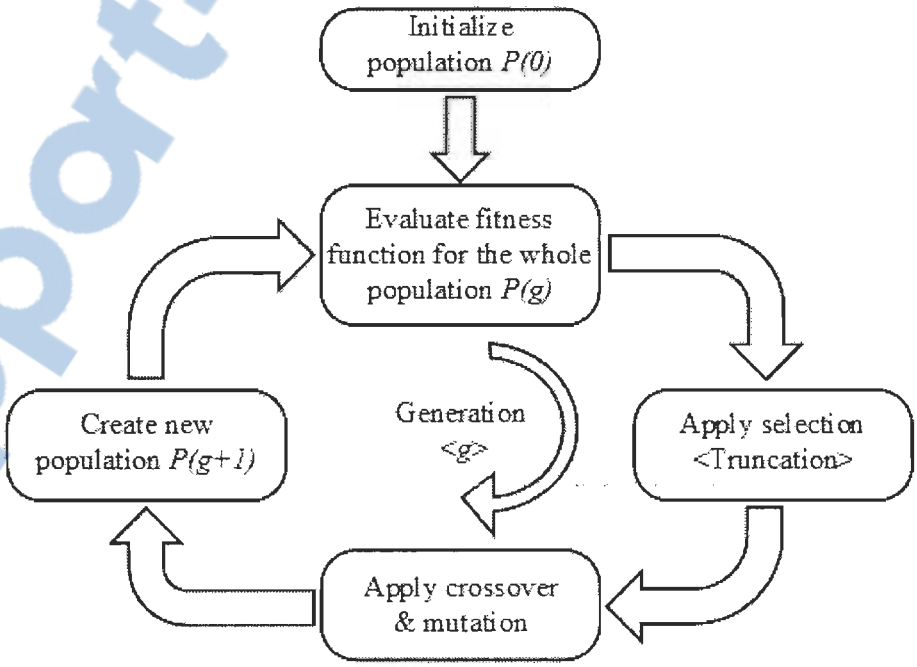


Figure 4-4 Operation diagram of genetic algorithm

It is basically a combination of five main operations: population initialization, fitness calculation, selection, crossover and mutation. Individuals forming the population are called chromosomes, each one represents the concatenated binary coded parameters, which in our case are the potential complex value weight vectors. The chromosome coding scheme case of complex tap coefficients wordlength, Q-bit, for 8-bit is as following:

w_1		.....	w_M	
Re	Im		Re	Im
11010010	00111010	.....	01110100	11101101

Initially, the population P with N_P chromosomes is generated in a random manner. Then, the fitness function block evaluates the fitness score during the training phase for every chromosome of the population using the following fitness function

$$f(n)=\sum_{m=0}^{\beta-1}\left|y(n-m)-s(n-D-m)\right| \tag{4.18}$$

Where β represents the size of the smoothing window.

The process of selecting potential parent chromosomes that will participate in the crossover and mutation operations is based on the truncation selection mechanism. This method gives more chances for the fittest chromosomes to be selected, forming then the first parents pool. The remaining chromosomes of the population form the second parents pool.

The crossover operation is performed on the selected parents with probability P_c as shown in Figure 4-5. One-point crossover is performed at the gene level (real or imaginary part). The crossover position $Pos(g)$ is adaptive, it varies according to the best fitness score

of the population. This approach allows large part of the randomly selected gene to be affected when high fitness score is observed. Alternatively, small part of the gene is affected when low fitness score is registered. This aided crossover helps increasing the convergence speed. The resulting child chromosome is passed to the mutation block.

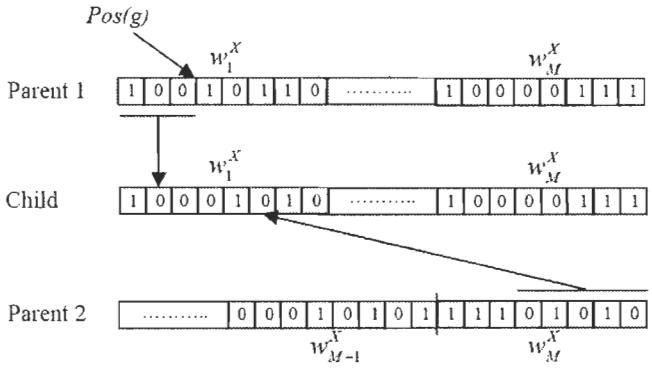


Figure 4-5 Crossover operation

The mutation operation is presented in Figure 4-6. A bitwise XOR operation is performed between a random part selected from the incoming chromosome and a random value. The resulting affected weight replaces the original. The variable mutation rate P_m is inversely proportional to the best fitness score of the population, which helps creating diversity within the population during all operation, helping to explore more search spaces. The resulting chromosomes form the new population $P(g+1)$.

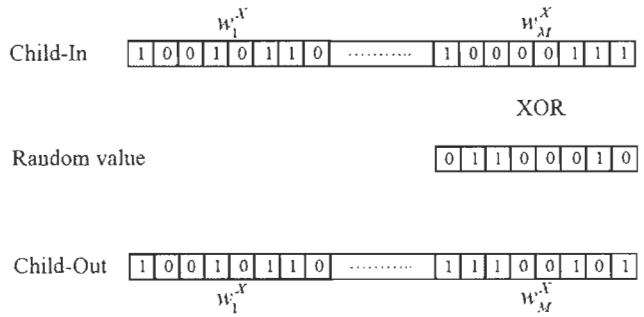


Figure 4-6 Mutation operation

4.6 Simulation Results

A highly dispersive non-linear channel is considered in this study, it is frequently encountered in communication systems [4]. The channel behavior is described by the following equations

$$y(n) = 0.3482s(n) + 0.8704s(n-1) + 0.3482s(n-2) \tag{4.19}$$

$$y_{NL}(n) = y(n) + 0.2y^2(n) \tag{4.20}$$

The equalization performance of the proposed VDFGA is compared to the CRTRL trained CDFRNN equalizer [4] considering the model presented in Figure 4-1. The neural network uses $N = 4$ hidden neurons, $M = 5$ external inputs and 4 feedback signals, and a transmission delay $D = 3$. A learning rate of $\mu = 0.01$ was carefully chosen in the stability limits to offer maximum convergence speed.

The genetic algorithm optimization uses population size of $N_P = 32$ chromosomes and a smoothing window of $\beta = 64$. A crossover probability of $P_c = 1$ and a variable mutation rate P_m ranging from 0.02 to 0.4. The weights were coded using the $Q7$ format (1 sign bit, 7 fractional bits) for the real and imaginary part value of taps. A feedback order of $m = 8$ was considered.

The study was done for the mean squared error (MSE) and the bit error rate (BER). The results are averaged over 50 independent trials and 10^4 training symbols. For all method evaluated, the weights corresponding to 500 training symbols were used to evaluation the BER using the next 10^4 symbols.

The MSE performance is presented in Figure 4-7 for a signal to noise ratio, $SNR = 16dB$. The VDFGA demonstrates very high convergence speed and less fluctuations compared to the CDFRNN. The VDFGA needed only 300 training data (or generations) to reach the steady state regime, while the CDFRNN continues its convergence even in the last iterations. This gain of performance is justified by the random natural evolution feature of the GA, in contrast to the CRTRL which is a gradient-following algorithm.

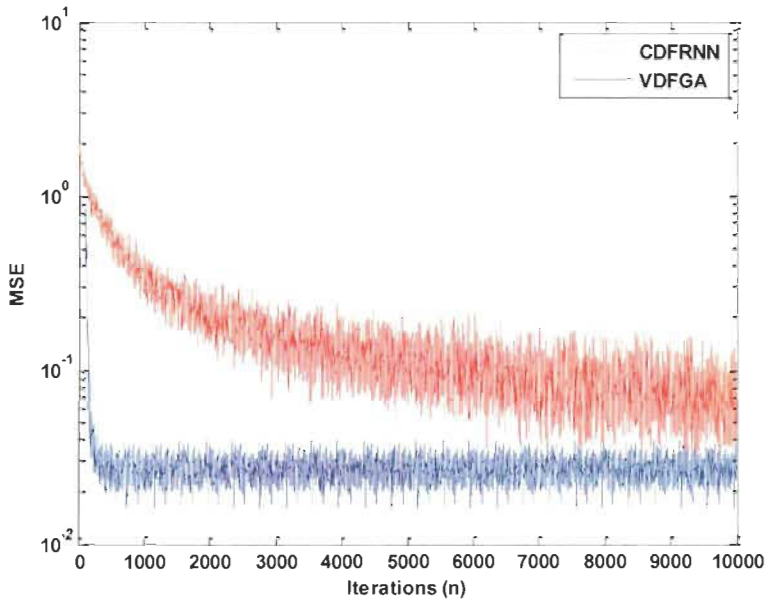


Figure 4-7 MSE performance for SNR=16 dB

Figure 4-8 shows the BER evaluation curves for SNR range of 4 to 24 dB. These results are a direct consequence of the fast convergence speed observed earlier, the VDFGA achieves a BER of about 10^{-2} for a $SNR = 24$, when the CDFRNN reaches a BER of 3.5×10^{-2} for the same SNR.

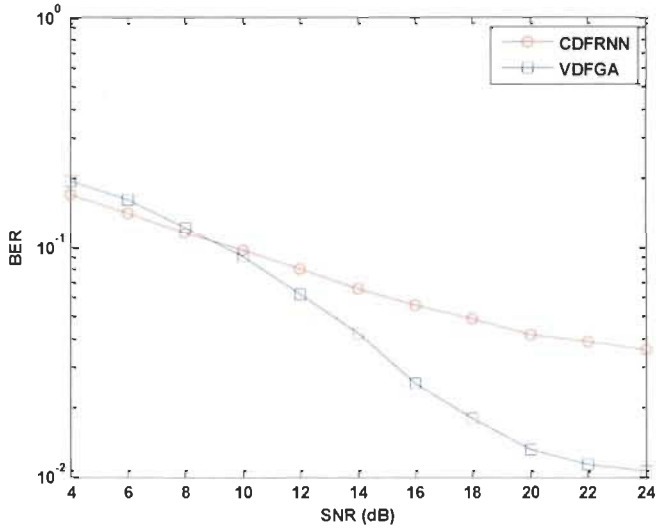


Figure 4-8 BER performance evaluation

4.7 Conclusion

In this work we presented a Volterra with GA based approach for equalizing nonlinear communication channels with 4-QAM links. The performance of the proposed technique was compared to a well known neural network based nonlinear equalization technique. The approach outperforms the reference method in term of convergence speed which allowed to achieve better BER with relatively short training symbols. The technique also used only 8 bits wordlength for fixed-point weights, compared to the floating point representation considered for the reference method.

4.8 Acknowledgement

The authors wish to thank the National Sciences and Engineering Research Council of Canada (NSERC) for financial support, Compute Canada for parallel equipment support, and the ReSMiQ.

4.9 References

[1] G. Kechriotis, E. Zervas, E.S. Manolakos, "Using recurrent neural networks for adaptive communication channel equalization," *IEEE Trans. Neural Networks*, vol. 5, pp. 267-278, 1994.

[2] T. Ogunfunmi and T. Drullinger, "Equalization of non-linear channels using a Volterra-based non-linear adaptive filter," *IEEE 54th Int. Midwest Symposium on Circuits and Systems (MWSCAS),* 2011, pp. 1-4.

[3] T. F. B. de Sousa and M. A. C. Fernandes, "Multilayer perceptron equalizer for optical communication systems," *IEEE MTT-S Int. Microwave & Optoelectronics Conference (IMOC)*, 2013, pp. 1-5.

[4] H. Q. Zhao, X. P. Zeng, Z. Y. He, W. D. Jin, and T. R. Li, "Complex-valued pipelined decision feedback recurrent neural network for non-linear channel equalisation," *IET Communications*, vol. 6, pp. 1082-1096, 2012.

[5] S. Ong, C. Sooyong, Y. Cheolwoo, and H. Daesik, "A complex version of a decision feedback recurrent neural equalizer as an infinite impulse response filter," *IEEE Global Communications Conference (GLOBECOM)*, 1997, pp. 57-61.

[6] G. Kechriotis, E.S. Manolakos, "Training fully recurrent neural networks with complex weights," *IEEE Trans. on Circuits and Systems II: Analog and DSP*, vol. 41, pp. 235-238, 1994.

[7] R. J. Williams and D. Zipser, "Experimental Analysis of the Real-time Recurrent Learning Algorithm," *Connection Science*, vol. 1, pp. 87-111, 1989.

[8] C. Jongsoo, A. C. C. Lima, and S. Haykin, "Kalman filter-trained recurrent neural equalizers for time-varying channels," *IEEE Trans. Comm*, vol. 53, pp. 472-480, 2005.

[9] H. M. Abbas and M. M. Bayoumi, "Volterra-system identification using adaptive real-coded genetic algorithm," *IEEE Trans. on Systems, Man and Cybernetics, Part A: Systems and Humans*, vol. 36, pp. 671-684, 2006.

[10] Cheng-Yuan, C. and C. Deng-Rui, "Active Noise Cancellation Without Secondary Path Identification by Using an Adaptive Genetic Algorithm," *IEEE Trans. on Instrumentation & Measur*, 59(9), 2010, pp. 2315-2327.

[11] D. Massicotte and D. Eke, "High robustness to quantification effect of an adaptive filter based on genetic algorithm," *IEEE NEW Circuits and Systems (NEWCAS)*, 2007, pp. 373-376.

Chapitre 5 - Algorithmes génétiques à faible longueur binaire pour les applications du filtrage adaptatif: *implémentation sur FPGA*

5.1 Sommaire

Dans la littérature, les travaux qui traitent les problèmes de filtrage adaptatif à base d'algorithmes génétiques sont principalement implémentés sur des machines fonctionnant avec des processeurs séquentiels. Ce type d'implémentation présente beaucoup d'inconvénients, comme la faible vitesse d'exécution et la consommation considérable d'énergie. Ces faiblesses sont critiques pour les applications qui nécessitent un fonctionnement en temps réel.

Dans ce travail nous proposons une implémentation sur FPGA de l'AG proposé dans le chapitre 3. La faible charge de calcul et la robustesse au calcul arithmétique à faible longueur binaire offertes par la méthode proposé sont favorables pour les implémentations sur des dispositifs à ressources matérielles limitées. Une architecture matérielle est proposée pour chaque bloc d'opération de l'AG.

Les résultats d'implémentation sont donnés pour le cas d'étude impliquant le problème d'identification abordée dans le chapitre 3. Des simulations comportementales sont faites pour vérifier le bon fonctionnement de l'architecture. L'étude du SQNR est donnée pour valider le fonctionnement global de l'implémentation. Les résultats d'implémentation sur matériel donnent une occupation de 498 slices et une vitesse de traitement de 320K échantillons/seconde.

Paper 3 **H. Merabti and D. Massicotte, " Hardware Implementation of a Real-time Genetic Algorithm for Adaptive Filtering Applications," in *IEEE Canadian Conference on Electrical and Computer Engineering (CCECE)*, Toronto, Canada, 4-7 May 2014.**

Paper 4 **H. Merabti and D. Massicotte, " FPGA Based Implementation of a Genetic Algorithm for ARMA Model Parameters Identification," in *ACM Great Lakes Symposium on VLSI (GLSVLSI)*, Houston, Texas, USA, 21-23 May 2014.** (Voir Annexe A)

Hardware Implementation of a Real-time Genetic Algorithm for Adaptive Filtering Applications

Hocine MERABTI and Daniel MASSICOTTE

Université du Québec à Trois-Rivières, Department of Electrical and Computer Engineering

Laboratoire des Signaux et Systèmes Intégrés

E-mail: { Hocine.Merabti, Daniel.Massicotte }@uqtr.ca

5.2 Abstract

Genetic algorithms are increasingly being used to address adaptive filtering problems. The interest lies in their ability to find the global solutions for linear and nonlinear problems. However, all the work available in the literature use software implementations running on sequential processors. This work proposes a hardware architecture of a real-time genetic algorithm for adaptive filtering applications. Specifically designed genetic operators are proposed to improve processing performance and robustness to the quantization effect, making low bit-wordlength fixed-point arithmetic implementation possible, which permit hardware cost saving. The proposed architecture is modeled in VHDL and implemented in FPGA using 6-bits wordlength, addressing linear and nonlinear ARMA model parameters identification problem. The implementation experiments show high signal processing performance and low footprint.

5.3 Introduction

Adaptive filtering is present in many signal processing applications such as identification [1], channel equalization [2] and noise cancellation [3]. The principle of adaptive filtering is to adjust the filter weights through successive iterations in the goal to reach an optimum solution called the Wiener solution [4]. The most popular linear adaptive

filtering algorithms are the recursive least square (RLS) and the least mean square (LMS). The RLS demonstrates high performance in terms of convergence speed and asymptotic error compared to LMS. However, the latest is the most used in practice due to its implementation simplicity [5]. When it comes to non-linear systems, artificial neural networks (ANNs) represent a great choice to consider. From the hardware implementation point of view, most of the work in the literature regarding ANNs is done without supporting the algorithm learning phase. The network is trained offline using software to estimate the weights. After that, weights are implemented along with the feedforward architecture. Still, few implementations supporting fully operational ANNs are proposed. However, they suffer from performances degradation compared to software due to limited data precision imposed by hardware resources constraints [6].

Recently, several research works brought up the use of genetic algorithms (GAs) to address adaptive filtering problems [7-8]. This interest is driven by the ability of these algorithms to search and find the global solution for the optimization problem whether the system is linear or non-linear. Another benefit of using GAs is the robustness to the quantization effect, thanks to their natural weights updating mechanism which does not require arithmetic operations [9]. However, all the work proposed in the literature consider sequential processor based software implementations only.

This paper presents a hardware architecture of a genetic algorithm for adaptive filtering applications. Genetic operators are specifically designed to improve the algorithm performance and speed, and robustness to quantization effect in low wordlength fixed-point processing environment. The proposed GA is modeled in very high speed integrated circuits hardware description language (VHDL) and implemented in field programmable

gate array (FPGA) using 6-bits wordlength arithmetic. The addressed adaptive filtering application is linear and nonlinear auto regressive moving average (ARMA) model parameters identification.

In this paper, we expose the following points; a description of the GA based adaptive filtering is done in Section 2. Section 3 details the hardware architecture of the proposed GA. Section 4 present the ARMA model parameters identification problem. In Section 5, implementation results and experiments are presented. Finally, we expose our conclusion in Section 6.

5.4 GA Based Adaptive Filtering

A general block diagram of a GA based adaptive filter is presented in Figure 5-1. The filter input vector is defined as $V(n) = [x(n), x(n-1)....., x(n-M+1)]^T$, the weights victor is $W(n) = [w_1(n), w_2(n) ..., w_M(n)]$, where M is the filter length. The output is expressed as: $y(n) = W(n).V(n)$.

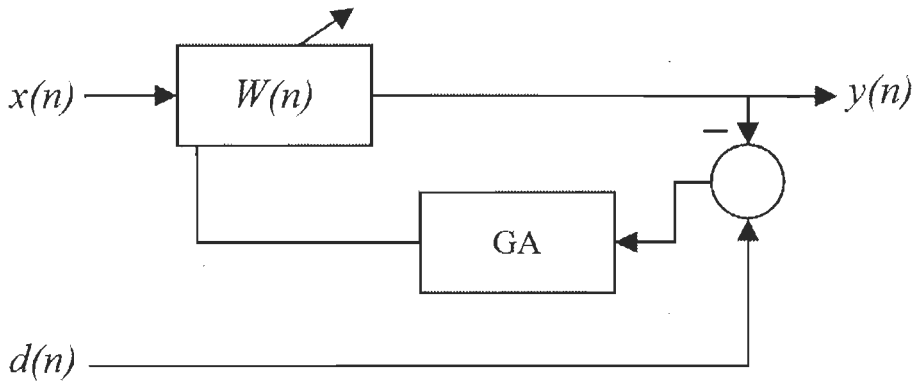


Figure 5-1 GA based adaptive filter

GA deals with the adaptive filtering process as an optimization problem where the goal is to minimize or maximize a cost function, which is usually called the fitness function. The

most common practice is to minimize the mean-squared error between the desired reference signal $d(n)$ and the filter output.

A typical fitness function has the following form [7]:

$$f(n)=\frac{1}{B}\sum_{m=0}^{B-1}\left(d(n-m)-y(n-m)\right)^2 \tag{5.1}$$

where B is the number of sampled data points. This smoothing operation is used to reduce the fluctuations effect caused by the adaptation process. Therefore, achieving a more stable and reliable fitness score.

The optimization process is performed by altering the filter weights using the natural updating mechanism of GAs.

The genetic algorithm is basically a combination of five main operations: population initialization, fitness calculation, selection, crossover and mutation (Figure 5-2). Individuals forming the population are called chromosomes, each one represents the concatenated binary coded parameters (genes), which in our case are the potential tap coefficients ($w_{1\ldots M}$). The coding scheme case of wordlength (wl), Q-bit, for 6-bit is as following:

w_1	w_2	$\cdots$	w_M
010010	111010	$\cdots$	010011

As a result, every parameter ranges in the interval [-1, $1-2^{-Q}$] (1 sign bit, and 5 fractional bits), leading to a chromosome searching space of 2^{QM} possibilities. The chromosomes change through successive iterations (sample n), called generations. The elitism model is adopted here, where the best individual of the current

population is passed to the next population. The process stops when a predefined number of generations (G) is reached.

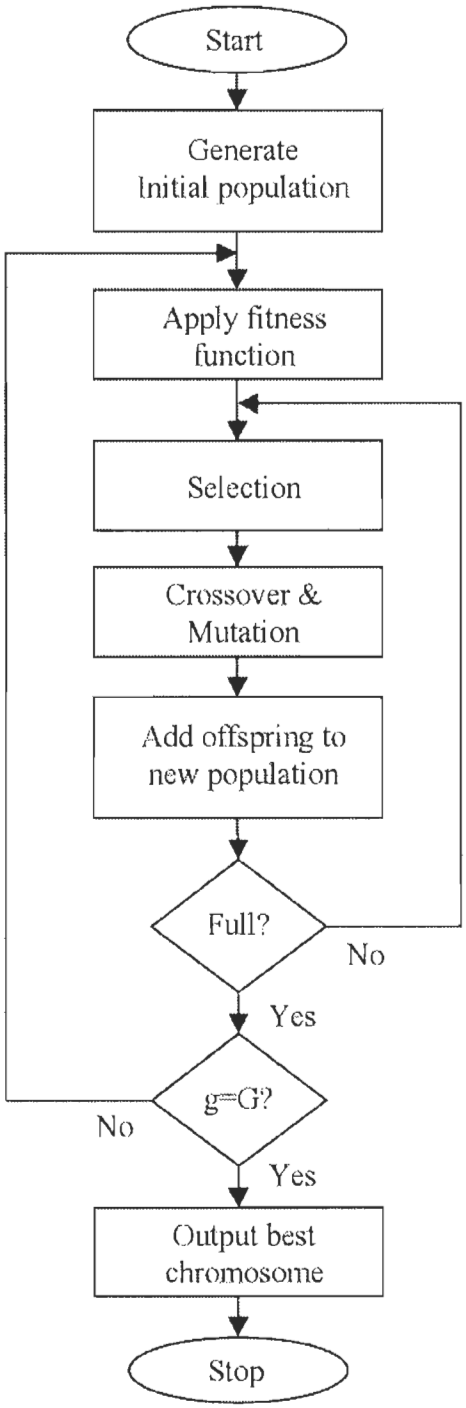


Figure 5-2 GA flowchart

5.5 Proposed Hardware Architecture

Hardware architectures and behavioral operation of the main genetic algorithm operations (Figure 5-2) are presented in this section. A finite state machine (FSM) based control unit is used to execute the operation process sequences.

5.5.1 Random Number Generator

The random number generator (RNG) is implemented using the popular linear feedback shift register (LFSR). It is used to generate the initial population, and random variables for crossover and mutation. The choice of using LFSR RNG is justified by its ability to provide long period pseudo-random sequences, using simple registers and few logic gates. Figure 5-3 shows the architecture of a 42-bits long LFSR based RNG.

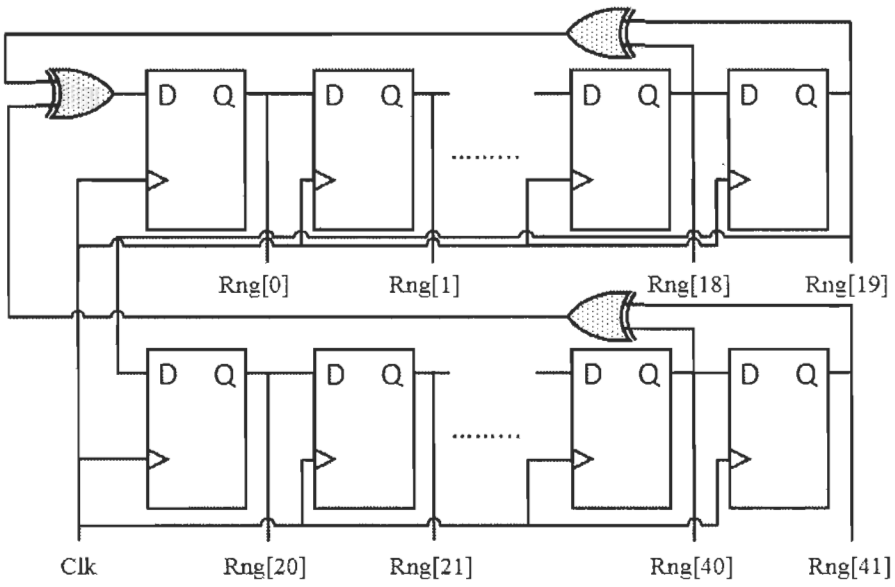


Figure 5-3 42 bits LFSR based random number generator

5.5.2 Fitness Block

The fitness block, which is shown in Figure 5-4, is used to implement the fitness function, that is defined as:

$$f(n)=\sum_{m=0}^{B-1}\left|y(n-m)-d(n-m)\right| \tag{5.2}$$

where $y(n)=W.V(n)$.

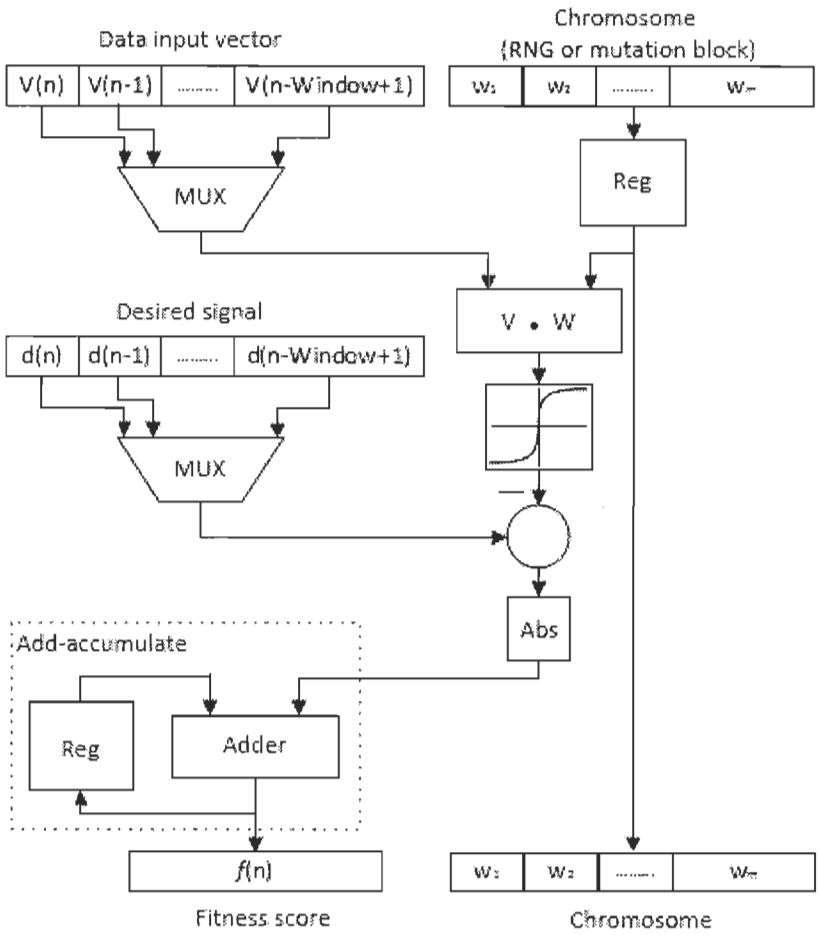


Figure 5-4 Hardware architecture of the fitness block

As it can be seen, this function is slightly different than (5.1). The square operator was replaced by the absolute value, and the division was completely removed. This helps the

reduction of the computation complexity, and increases the robustness to quantization effect in low wordlength fixed-point arithmetic. A nonlinear activation function is also added to the output of the filter in order to deal with nonlinear applications. It has a hyperbolic tangent function, the nonlinear function can be directly mapped in a lookup table (LUT).

In the beginning, the chromosome coming from the initial or the new population, is stored in a register. The fitness score is then calculated using conventional arithmetic operators. The summation is performed using an add-accumulate circuit in combination with multiplexers for data input selection. The resulting fitness score is transmitted along with the chromosome to the selection block.This process is repeated for all chromosomes of the population.

5.5.3 Selection Block

The hardware architecture of the selection block is presented in Figure 5-5, the truncation selection technique is adopted. Three main steps are needed to complete the required operation.

During the first step, chromosomes which are sequentially received from the fitness block are sorted in the ascending order based on their fitness score. This is completed by using a largest-out linear array sorter [10]. The sorted chromosomes form the first parents pool. In the second step, a simplified truncation circuit form the second pool of parents with a reproduction of the fittest chromosomes only, following predefined proportions. In the last step, candidates are randomly selected from the parent pools for the crossover operation.

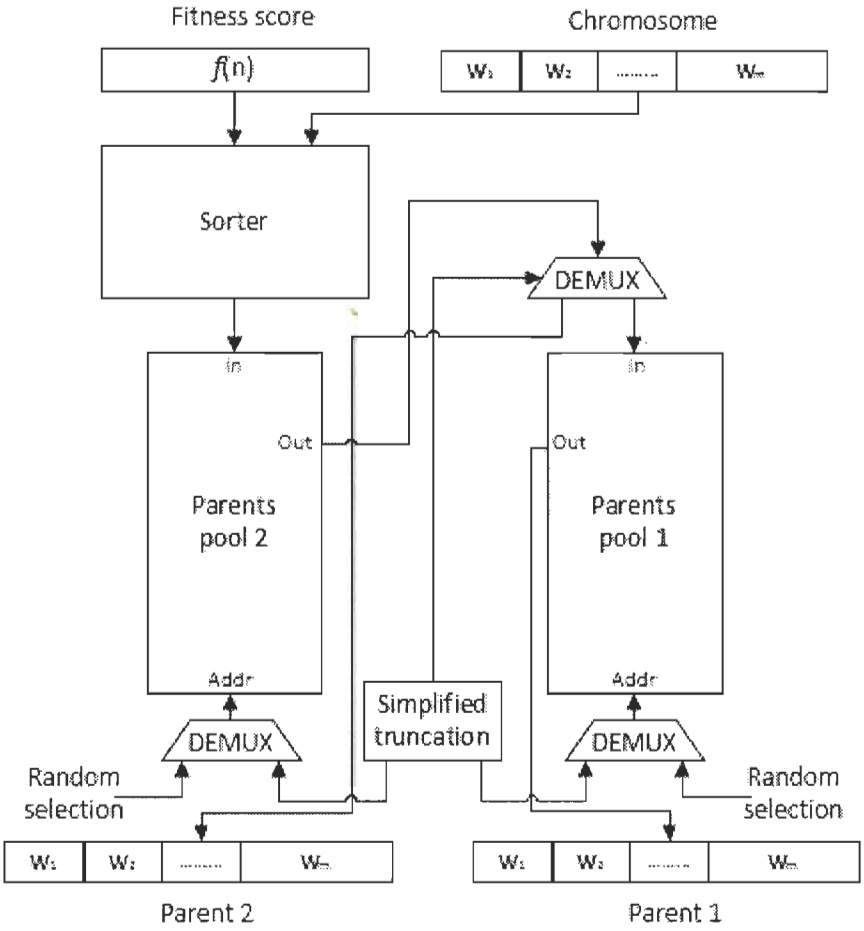


Figure 5-5 Hardware architecture of the selection block

5.5.4 Crossover Block

The hardware architecture of the crossover operator is given in Figure 5-6. A new crossover approach is adopted in the aim to accelerate the GA convergence speed. Single-point crossover is performed on the selected chromosomes at the gene level, with a crossover rate of 1. The participating genes are randomly selected. The crossover point position is set according to the fitness score of the best chromosome of the population. For high fitness scores (synonym to low fit solutions), the crossover point is close to the MSB of the weight, thereby allowing replacement of many bits of the gene. On the other hand, for low fitness scores, the crossover point is near the LSB. Subsequently, small number of

bits will be affected. The resulting weight takes place of the original weight in the first parent chromosome (selected among the fittest parents pool), forming an offspring chromosome for the mutation block.

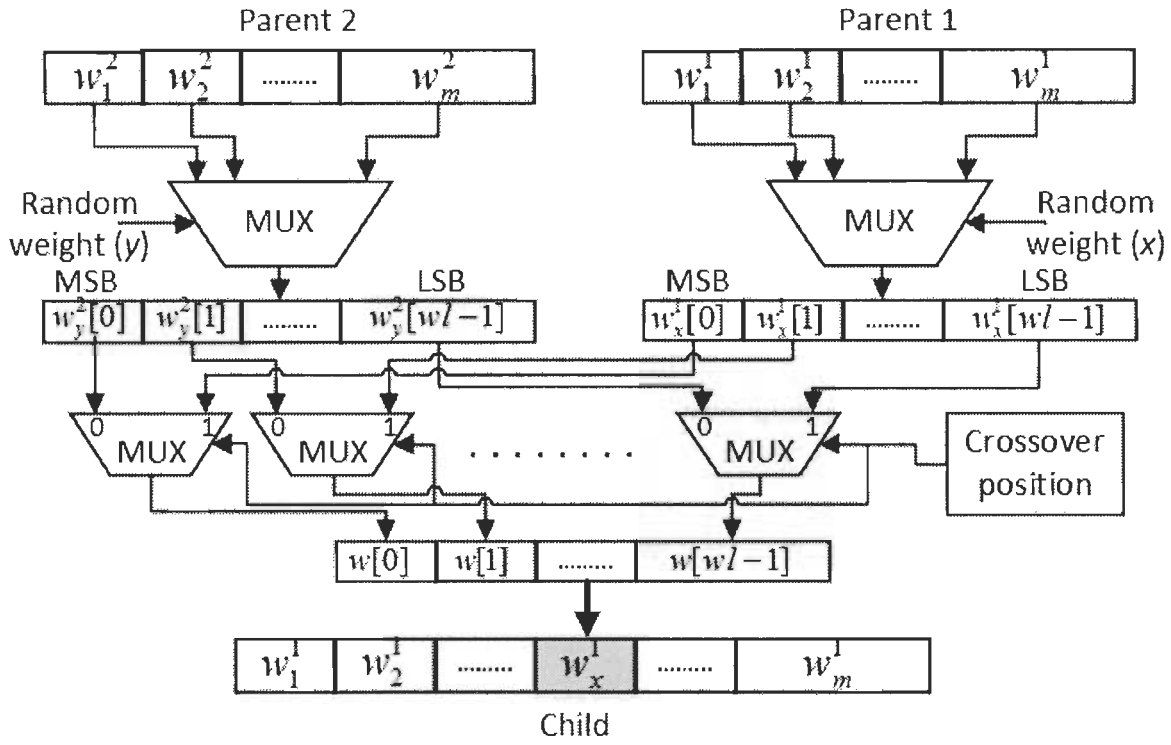


Figure 5-6 Hardware architecture of the crossover block

5.5.5 Mutation Block

Figure 5-7 shows the hardware architecture of the mutation block. The mutation is achieved by performing a XOR operation between a randomly selected weight, and a random generated binary word. The resulting chromosome will be part of the new population that will be used in the following generation. A variable mutation rate based on the best fitness score of the population is adopted here. It gradually increases from 0.1 to 0.5 when the fitness score varies from its highest to the lowest value. The motivation

behind this is to create large diversity within the population to explore more search spaces. Thereby, helping the GA to escape local minimums.

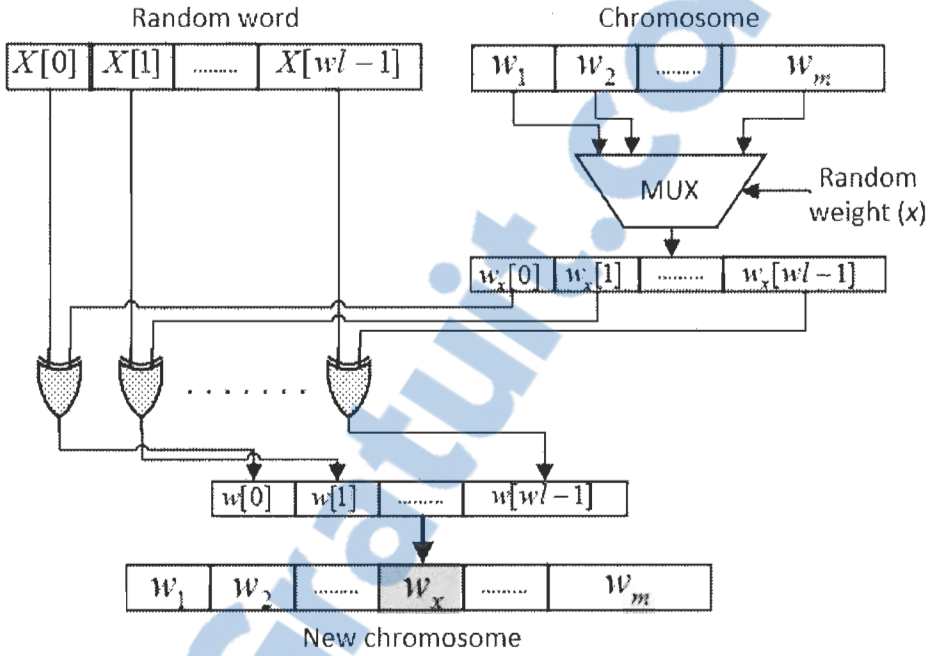


Figure 5-7 Hardware architecture of the Mutation block

5.6 Study Case: System Identification

The proposed GA presented in the previous section is used to identify the parameters of the following ARMA model, Figure 5-8 [11]:

$$y_L(n) = \sum_{k=1}^{N_a} a_k y_L(n-k) + \sum_{k=0}^{N_b} b_k x(n-k) + \sum_{k=0}^{N_c} c_k u(n-k) \tag{5.3}$$

for linear system,

$$y_{NL}(n) = tanh\left(v_{NL} y_L(n)\right) \tag{5.4}$$

for nonlinear system, where $y_L(n)$ is the output of the linear system; $y_{NL}(n)$ and v_{NL} are the output of the nonlinear system and the variance of the nonlinearity, respectively ; $x(n)$

is the input, $y(n)$ is the output and $u(n)$ is a noise signal. We suppose no correlation between $u(n)$ and the output signal. The input signal $x(n)$ is a random signal here, uncorrelated with $u(n)$. $u(n)$ is a white Gaussian noise with a zero mean and is 0.2 variance.

The problem consists to identify the system coefficients $\{a_k\}, \{b_k\}$ et $\{c_k\}$ using the system input and output signals. The model ARMA has non zero coefficients for $a_1, a_2, b_5, b_6, c_0, c_1,$ and, c_2.

To identify the ARMA parameters, we defined the input vector

$\mathbf{v}(n) = [y(n-1),\ y(n-2),\ x(n-5),\ x(n-6),\ u(n), u(n-1),\ u(n-2)]^T$, $\dim(\mathbf{v}) = M \times 1$ with the following vector used to estimate ARMA parameters $\mathbf{w}(n) = [w_1(n)\ \ w_2(n)\ \ \cdots\ \ w_M(n)]^T$ where M defines the number or ARMA parameters to identify (M=7).

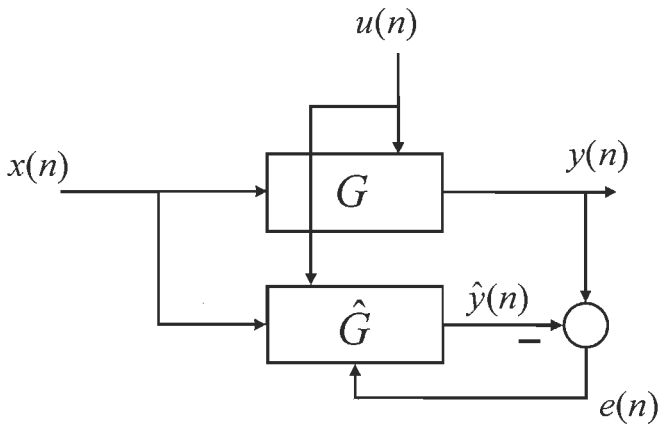


Figure 5-8 Block diagram for the system identification

5.7 Implementation Results

The GA based ARMA identification system was implemented on the XILINX Spartan-6 xc6slx4-3tqg144 FPGA. A population of 16 chromosomes and a smoothing window size of B=16 were used. To test the robustness to the quantization effect, low fixed-point wordlength arithmetic environment was considered, where all arithmetic operations were implemented using the Q5 format (1 sign bit and 5 fractional bits).

The post place and route implementation results and the timing performance achieved are presented in Table 5-1 and Table 5-2 respectively. No DSP slice or block memory have been used in the design, LUTs were used instead for faster performance. The design used 83% of the total device logic, and was able to provide a processing rate of 320 Kilo Generation/second.

Table 5-1 Post Place-and-Route implementation summary

Resource utilization	Available	Used
Slice registers	4800	889 (18%)
Slice LUTs	2400	1415 (58%)
Total Slices	498	600 (83%)

Table 5-2 Timing performance

Parameters	Rate
Maximum clock frequency	105 MHz
Maximum group rate	320 KGen/sec

The signal processing performance analysis is done on the Signal to Quantization Noise Ratio (SQNR)

$$SQNR(n) = 10\log\left(\sum_{i=1}^{M} w_i^2 \Big/ \sum_{i=1}^{M}\left(w_i - \hat{w}_i(n)\right)^2\right)(\text{dB}) \tag{5.5}$$

where w_i is the exact coefficient, and $\hat{w}_i$ is the estimated coefficient.

To investigate the ability of the GA to successfully identify the ARMA coefficients, 10 independent realizations were performed. In each realization, system signals are produced for randomly generated coefficients, ranging in the intervals [0,1[for a_1, b_5, b_6, c_0, c_2, and [-1,0] for a_2, c_1. For the nonlinear system, a strong linearity variance $v_{NL} = 2$ is used.

Figure 5-9 shows the average SQNR of all realizations for the linear and the nonlinear identification problems described by (3) and (4), respectively. A theoretical SQNR is given as a reference metric, it is calculated by using the Q5 truncation quantization of the perfect floating-point coefficients, it has a mean value of 26.9 dB.

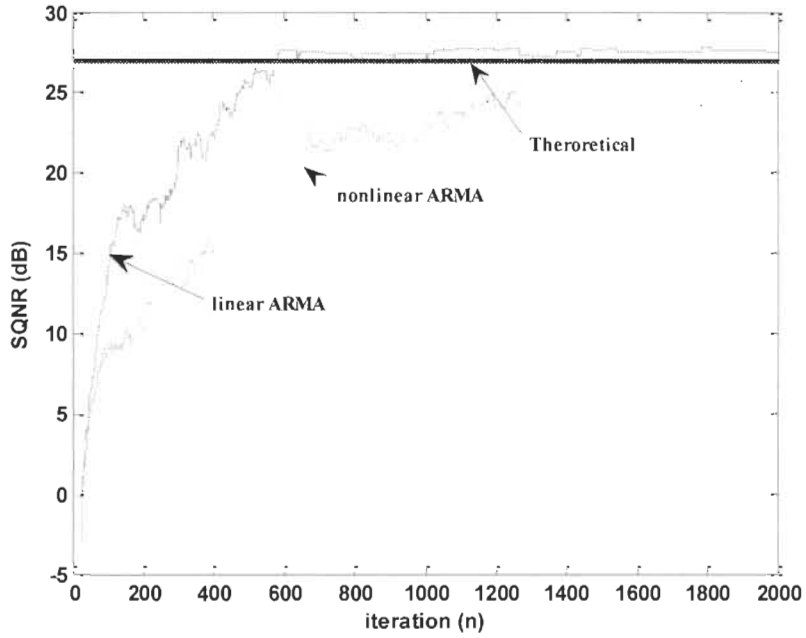


Figure 5-9 System identification performance results

The implemented GA converges to a steady state with an SQNR of 27.6 dB at generation 600 for the linear identification, it is 0.7 dB more than the theoretical SQNR. This gain of performance is justified by the ability of the GA to perform rounding by altering the coefficients LSBs through the optimization process.

For nonlinear identification, the GA still capable of estimating the system parameters with a maximum SQNR of 24.7 dB, which is 2.2 dB below the theoretical SQNR and 2.9 dB lower than the performance achieved in linear identification. This is very interesting given the strong nonlinearity considered in the study. 1200 generations were required for the GA to converge to steady state.

5.8 Conclusion

A hardware architecture of a real-time genetic algorithm for adaptive filtering applications has been proposed. The design considered low wordlength fixed-point arithmetic processing environement. An FPGA implementation was provided to verify the functionality of the proposed GA, treating an ARMA model parameters identification problem in linear and nonlinear scenarios. The implementation shows high signal processing performances, where only 6-bits worlength fixed-point arithmetic was used in all operations.

5.9 Acknowledgement

The authors wish to thank the National Sciences and Engineering Research Council of Canada (NSERC) for financial support.

5.10 References

[1] C. Antweiler, A. Telle, P. Vary, and G. Enzner, "Perfect-sweep NLMS for time-variant acoustic system identification," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP),* 2012, pp. 517-520.

[2] M. U. Otaru, A. Zerguine, and L. Cheded, "Adaptive channel equalization: A simplified approach using the quantized-LMF algorithm," in *IEEE International Symposium on Circuits and Systems (ISCAS)*, 2008, pp. 1136-1139.

[3] M. G. Morrow, C. H. G. Wright, and T. B. Welch, "Real-time DSP for adaptive filters: A teaching opportunity," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP),* 2013, pp. 4335-4338.

[4] S. Haykin, Adaptive Filter Theory, Prentice-Hall, 1996.

[5] B. Widrow, "Thinking about thinking: the discovery of the LMS algorithm,"in *IEEE Signal Processing Magazine*, vol. 22, pp. 100-106, 2005.

[6] L. Jihong and L. Deqin, "A Survey of FPGA-Based Hardware Implementation of ANNs," in *International Conference on Neural Networks and Brain, ICNN&B '05*, 2005, pp. 915-918.

[7] H. M. Abbas and M. M. Bayoumi, "Volterra-system identification using adaptive real-coded genetic algorithm," *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, vol. 36, pp. 671-684, 2006.

[8] Cheng-Yuan, C. and C. Deng-Rui, "Active Noise Cancellation Without Secondary Path Identification by Using an Adaptive Genetic Algorithm,"*IEEE Transactions on Instrumentation and Measurement*, 59(9), 2010, pp. 2315-2327.

[9] D. Massicotte and D. Eke, "High robustness to quantification effect of an adaptive filter based on genetic algorithm," in *IEEE Northeast Workshop on Circuits and Systems (NEWCAS)*, 2007, pp. 373-376.

[10] V. A. Pedroni, R. P. Jasinski, and R. U. Pedroni, "A very efficient single-iteration oldest-out data sorter," in *IEEE International Symposium on Circuits and Systems (ISCAS)*, 2011, pp. 2141-2144.

[11] D. Vu and A. R. Stubberud, "System identification by genetic algorithm," in *IEEE Aerospace Conference Proceedings*, 2002, pp. 5-2331-5-2337 vol.5.

Chapitre 6 - Conclusion

Les algorithmes du filtrage adaptatif numérique sont sollicités pour traiter plusieurs problèmes relevant du traitement numérique de signal. Parmi les applications du filtrage adaptatif, on retrouve l'identification de système et l'égalisation de canal. Le choix de l'algorithme adéquat repose sur des critères de performance comme la vitesse de convergence, l'erreur asymptotique, sensibilité aux paramètres propres du système, ainsi que sur la nature du system adressé, autrement dit, linéaire ou non-linéaire. Faire appel à un algorithme à haute performance ou une technique capable d'adresser des systèmes non-linéaires peut nécessiter une grande charge de calcul. Ce besoin de ressources est principalement causé par le nombre d'opérations arithmétiques élevé, ainsi que la longueur des mots binaires nécessaire pour garantir le niveau de performances recherché. Dans le but de proposer une technique performante, capable d'adresser des problèmes non-linéaires et implémentable avec peu de complexité, nous avons exploré le filtrage adaptatif à base d'AGs. Ces algorithmes évolutionnaires sont caractérisés par leur: capacité à trouver le minimum global dans le processus d'optimisation, faculté à traiter des problèmes non-linéaires, vitesse de convergence élevée et la robustesse aux effets de quantification. Pour améliorer ces caractéristiques, des opérateurs génétiques ont été développés. La première application utilisée pour évaluer les performances de l'AG proposé est l'identification de paramètres d'un model ARMA linéaire et non-linéaire, en faisant une comparaison avec le LMS et les résultats théoriques. L'AG a été entièrement implémenté en virgule fixe à très faible longueur binaire où des mots binaires de 6-bits ont été utilisés. Dans la première phase de l'étude, des coefficients fixes ont été considérés. Les résultats obtenus avec Matlab

pour le modèle linéaire ont présenté un gain en SQNR de 2.5 dB par rapport au LMS et une dégradation de seulement 0.5 dB par rapport aux résultats théoriques. Une vitesse de convergence similaire à celle du LMS a été observée. L'étude du coefficient de variation a permis d'évaluer la stabilité de l'AG. Un C_v aussi bon que celui du LMS a été atteint dans les dernières itérations, malgré la nature aléatoire de l'AG. La supériorité écrasante de l'AG a été démontrée dans le cas du modèle non-linéaire, l'AG a continué à fonctionner avec un SQNR de 25 dB et un C_v de 38% alors que le LMS n'a présenté aucun signe de convergence avec un SQNR saturé à 3.8 dB. Dans la deuxième phase, la capacité d'identifier des coefficients générés aléatoirement dans chaque répétition est explorée. La vitesse de convergence de l'AG a considérablement augmenté dans les deux cas du modèle, mais une petite dégradation du SQNR et du C_v a été constatée dans le modèle linéaire. La dégradation du C_v est causée par la variation du SQNR théorique d'une répétition à une autre.

Dans la deuxième partie du projet, nous nous sommes intéressés à l'application d'égalisation de canal non-linéaire. Les performances de l'égaliseur à base d'AG ont été comparées à celles d'un égaliseur à base de CDFRNN. En observant les courbes du MSE, l'AG a pu se distinguer par une impressionnante vitesse de convergence, qui lui a permis d'atteindre le régime permanant en moins de 300 itérations, tandis que le CDFRNN a continuer de converger lentement même dans les dernières itérations. Dans l'étude du BER, 500 symboles d'entrainement ont été utilisés pour entrainer les deux égaliseurs. La technique à base d'AG a achevé un BER d'environ 10^{-2} pour SNR=24 dB, alors que l'égaliseur à base de CDFRNN a atteint un BER de 3.5×10^{-2} pour le même SNR. Ce gain

a été achevé en utilisant des coefficients codés sur 8-bits seulement pour l'AG, pendant que le CDFRNN emploi une représentation en virgule flottante.

La dernière partie de ce mémoire a été consacrée à l'implémentation sur FPGA de l'AG proposé. L'application d'identification de système a été considérée pour valider le fonctionnement de l'architecture. Les résultats du SQNR obtenus avec le simulateur ISim ont démontré des performances remarquables. Pour le modèle linéaire, l'AG a pu atteindre un SQNR de 27.6 dB pour 600 générations, ce qui représente 0.7 dB plus que le SQNR théorique. Ce gain de performance se justifie par la capacité de l'AG à faire de l'arrondissement, en jouant sur les bits les moins significatifs des coefficients. Pour le cas non-linéaire, un SQNR de 24.7 dB pour 1200 générations a été atteint. Cette dégradation est causée par la sévère non-linéarité considérée dans l'étude. Les résultats de synthèse de l'implémentation sur un dispositif de la famille Spartan 5 de Xilinx, ont révélé une faible consommation de ressources avec 498 slices occupées et une vitesse de traitement maximale de 320K générations par seconde.

Bibliographie

[1] S. Haykin, "Adaptive Filter Theory," *Prentice-Hall,* 1996.

[2] Y. Rongshan, S. Ying, and S. Rahardja, "LMS in prominent system subspace for fast system identification," in *IEEE Statistical Signal Processing Workshop (SSP)*, 2012, pp. 209-212.

[3] C. Yilun, G. Yuantao, and A. O. Hero, "Sparse LMS for system identification," in *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2009*, 2009, pp. 3125-3128.

[4] C. Lan-Jian, F. Zhi-Zhong, and Y. Qing-Kun, "Improved robustness adaptive step size LMS equalization algorithm and its analysis," in *International Conference on Computational Problem-Solving (ICCP)*, 2010, pp. 141-144.

[5] F. Yangyang, C. Xue, Z. Weiqin, Z. Xian, and Z. Hai, "The Comparison of CMA and LMS Equalization Algorithms in Optical Coherent Receivers," in *6th International Conference on Wireless Communications Networking and Mobile Computing (WiCOM)*, 2010, pp. 1-4.

[6] B. Widrow, J. R. Glover, Jr., J. M. McCool, J. Kaunitz, C. S. Williams, R. H. Hearn, *et al.*, "Adaptive noise cancelling: Principles and applications," *Proceedings of the IEEE,* vol. 63, pp. 1692-1716, 1975.

[7] R. Candido, M. T. M. Silva, and V. H. Nascimento, "Affine combinations of adaptive filters," in *42nd Asilomar Conference on Signals, Systems and Computers*, 2008, pp. 236-240.

[8] M. Lazaro-Gredilla, L. A. Azpicueta-Ruiz, A. R. Figueiras-Vidal, and J. Arenas-Garcia, "Adaptively Biasing the Weights of Adaptive Filters," *IEEE Transactions on Signal Processing,* vol. 58, pp. 3890-3895, 2010.

[9] A. S. Prasad, S. Vasudevan, R. Selvalakshmi, K. S. Ram, G. Subhashini, S. Sujitha, *et al.*, "Analysis of adaptive algorithms for digital beamforming in Smart Antennas," in *International Conference on Recent Trends in Information Technology (ICRTIT)*, 2011, pp. 64-68.

[10] Y. Gengyun, G. Fengxiang, W. Changsong, and C. Xiao, "Design and simulation based on Kalman filter fuzzy adaptive PID control for mold liquid level control system," in *Chinese Conference on Control and Decision , CCDC '09*, 2009, pp. 6105-6109.

[11] K. E. Grosspietsch, "Adaptive filters for the dependable control of autonomous robot systems," in *Proceedings of the19th IEEE International Symposium on Parallel and Distributed Processing*, 2005, p. 4 pp.

[12] J. Hesselbach, H. W. Hoffmeister, and K. Loeis, "Multiple channel Filtered-X LMS-RLS vibration control in wood machining," in *IEEE International Conference on Control and Automation*, 2009, pp. 2060-2065.

[13] Q. Huang, Z. Gao, S. Gao, Y. Shao, and X. Zhu, "Comparison of LMS and RLS Algorithm for Active Vibration Control of Smart Structures," in *Third International Conference on Measuring Technology and Mechatronics Automation (ICMTMA)*, 2011, pp. 745-748.

[14] Z. Jun and Z. Xing-qun, "Study on ship manoeuvering control based on adaptive inverse control technology," in *Proceedingsof the IEEE Intelligent Transportation Systems*, 2003, pp. 1698-1703 vol.2.

[15] Y. Tu, R. Zhou, and F. Zhang, "ECG Signal Preprocessing Based on Change Step Iteration of the LMS Adaptive Filtering Algorithm," in *WRI World Congress on Computer Science and Information Engineering* 2009, pp. 155-159.

[16] W. Yuan and R. Zhou, "An Improved Self-Adaptive Filter Based on LMS Algorithm for Filtering 50Hz Interference in ECG Signals," in *8th International Conference on Electronic Measurement and Instruments, ICEMI '07*, 2007, pp. 3-874-3-878.

[17] K. Elangovan, "Comparative study on the channel estimation for OFDM system using LMS, NLMS and RLS algorithms," in *International Conference on Pattern Recognition, Informatics and Medical Engineering (PRIME)*, 2012, pp. 359-363.

[18] B. Widrow, "Thinking about thinking: the discovery of the LMS algorithm," *IEEE Signal Processing Magazine,* vol. 22, pp. 100-106, 2005.

[19] S. C. Douglas and T. H. Y. Meng, "An optimum NLMS algorithm: performance improvement over LMS," in *International Conference on Acoustics, Speech, and Signal Processing*, 1991, pp. 2125-2128 vol.3.

[20] S. M. Kuo and D. R. Morgan, "Active noise control: a tutorial review," *Proceedings of the IEEE,* vol. 87, pp. 943-973, 1999.

[21] R. E. Uhrig, "Introduction to artificial neural networks," in *Proceedings of the IEEE IECON 21st International Conference on Industrial Electronics, Control, and Instrumentation*, 1995, pp. 33-37 vol.1.

[22] A. Borys, "Nonlinear Aspects of Telecommunications: Discrete Volterra Series and Nonlinear Echo Cancellation," *CRC Press,* 2000.

[23] D. F. Eke, "Méthode d'identification en virgule fixe d'un modèle non linéaire basé sur les algorithmes génétiques," *Mémoire de maîtrise,* May 2008.

[24] D. Massicotte and D. Eke, "High robustness to quantification effect of an adaptive filter based on genetic algorithm," in *IEEE Northeast Workshop on Circuits and Systems*, 2007, pp. 373-376.

[25] L. Jihong and L. Deqin, "A Survey of FPGA-Based Hardware Implementation of ANNs," in *International Conference on Neural Networks and Brain, ICNN&B '05* 2005, pp. 915-918.

[26] R. A. Vural, T. Yildirim, T. Kadioglu, and A. Basargan, "Performance Evaluation of Evolutionary Algorithms for Optimal Filter Design," *IEEE Transactions on Evolutionary Computation,* vol. 16, pp. 135-147, 2012.

[27] A. Das and R. Vemuri, "An Automated Passive Analog Circuit Synthesis Framework using Genetic Algorithms," in *IEEE Computer Society Annual Symposium on VLSI, ISVLSI '07*, 2007, pp. 145-152.

[28] C. Goh and Y. Li, "GA automated design and synthesis of analog circuits with practical constraints," in *Proceedings of the 2001 Congress on Evolutionary Computation*, 2001, pp. 170-177 vol. 1.

[29] N. E. Mastorakis, I. F. Gonos, and M. N. S. Swamy, "Design of two-dimensional recursive filters using genetic algorithms," *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications,* vol. 50, pp. 634-639, 2003.

[30] R. S. Zebulum, M. A. Pacheco, and M. Vellasco, "Comparison of different evolutionary methodologies applied to electronic filter design," in *Proceedings of the IEEE International Conference on Evolutionary Computation*, 1998, pp. 434-439.

[31] S. Sriranganathan, D. R. Bull, and D. W. Redmill, "Low complexity two-dimensional digital filters using unconstrained SPT term allocation," in *IEEE International Symposium on Circuits and Systems, ISCAS '96, Connecting the World*, 1996, pp. 762-765 vol.2.

[32] X. Xu and B. Nowrouzian, "Local search algorithm for the design of multiplierless digital filters with CSD multiplier coefficients," in *IEEE Canadian Conference on Electrical and Computer Engineering*, 1999, pp. 811-816 vol.2.

[33] C. Cheng-Yuan and C. Deng-Rui, "Active Noise Cancellation Without Secondary Path Identification by Using an Adaptive Genetic Algorithm," *IEEE Transactions on Instrumentation and Measurement,* vol. 59, pp. 2315-2327, 2010.

[34] V. Duong and A. R. Stubberud, "System identification by genetic algorithm," in *IEEE Aerospace Conference Proceedings*, 2002, pp. 5-2331-5-2337 vol.5.

[35] F. Russo and G. L. Sicuranza, "Accuracy and Performance Evaluation in the Genetic Optimization of Nonlinear Systems for Active Noise Control," *IEEE Transactions on Instrumentation and Measurement,* vol. 56, pp. 1443-1450, 2007.

[36] S. C. Douglas, "Introduction to Adaptive Filters," *CRC Press LLC,* 1999.

[37] T. Ogunfunmi and T. Drullinger, "Equalization of non-linear channels using a Volterra-based non-linear adaptive filter," in *IEEE 54th International Midwest Symposium on Circuits and Systems (MWSCAS)*, 2011, pp. 1-4.

[38] M.Gen and R.Cheng, "Genetic Algorithms and Engineering Design," *Wiley-Interscience publication,* 1997.

[39] M. A. Vega-Rodriguez, R. Gutierrez-Gil, J. M. Avila-Roman, J. M. Sanchez-Perez, and J. A. Gomez-Pulido, "Genetic algorithms using parallelism and FPGAs: the

TSP as case study," in *International Conference Workshops on Parallel Processing*, 2005, pp. 573-579.

[40] C. Pei-Yin, C. Ren-Der, C. Yu-Pin, S. Leang-san, and H. A. Malki, "Hardware Implementation for a Genetic Algorithm," *IEEE Transactions on Instrumentation and Measurement,* vol. 57, pp. 699-705, 2008.

[41] P. R. Fernando, S. Katkoori, D. Keymeulen, R. Zebulum, and A. Stoica, "Customizable FPGA IP Core Implementation of a General-Purpose Genetic Algorithm Engine," *IEEE Transactions on Evolutionary Computation,* vol. 14, pp. 133-149, 2010.

Annexe A – FPGA Based Implementation of a Genetic Algorithm for ARMA Model Parameters Identification

Hocine MERABTI and Daniel MASSICOTTE

Université du Québec à Trois-Rivières, Department of Electrical and Computer Engineering

Laboratoire des Signaux et Systèmes Intégrés

E-mail: { Hocine.Merabti, Daniel.Massicotte }@uqtr.ca

Abstract

In this paper, we propose an FPGA implementation of a genetic algorithm (GA) for linear and nonlinear auto regressive moving average (ARMA) model parameters identification. The GA features specifically designed genetic operators for adaptive filtering applications. The design was implemented using very low bit-wordlength fixed-point representation, where only 6-bit wordlength arithmetic was used. The implementation experiments show high parameters identification capabilities and low footprint.

Categories and Subject Descriptors

C.3 [**Special-purpose and application-based systems**]: Signal Processing Systems

Keywords

Genetic algorithms, ARMA, System Identification, Adaptive filtering, Non-linear systems, Low wordlength arithmetic, FPGA

Introduction

System identification is a major issue in signal processing and various other fields [1].ARMA models have been involved in many time series analysis studies. Using adaptive

filtering techniques to identify the parameters of an ARMA model is a common practice. However, addressing nonlinear models require the use of nonlinear adaptive filtering algorithms, which are generally much more complex than linear techniques.

Recently, several research works brought up the use of GAs to address adaptive filtering problems [2-3]. This interest is driven by the ability of these algorithms to search and find the global solution for the optimization problem whether the system is linear or nonlinear. Another benefit of using GAs is the robustness to the quantization effect, thanks to their natural weights updating mechanism which does not require arithmetic operations [4]. However, all the proposed works in the literature for adaptive filtering using GA, consider sequential processor based software implementations only.

In this work, we propose a field programmable gate array (FPGA) implementation of a GA for linear and nonlinear ARMA model parameters identification. The GA with specifically designed genetic operators for adaptive filtering applications, and the ARMA identification problem were studied in [5]. This algorithm was designed with a special focus on computation load reduction and robustness increase to the quantization effect in low wordlength fixed-point processing environment, which makes it very attractive for hardware implementations. The target GA based identification system is modeled in very high speed integrated circuits hardware description language (VHDL) and implemented in FPGA using 6-bits wordlength arithmetic.

GA Operation

The flowchart of the implemented GA is shown in Figure 1. The operation begins by generating a random initial population *P(0)*. Individuals forming the population are called

chromosomes, each represents the concatenated binary coded parameters (genes), which in our case are the potential ARMA model coefficients. The fitness function is then applied to every chromosome of the population. Next, the fitness scores calculated by the previous operation are used by the selection mechanism to select the two chromosomes that will participate in the crossover operation. After that, the offspring generated by the crossover is passed through the mutation device. The resulting chromosome is used to form the new population *P(g+1)*. Once the newly formed population is complete, the process (fitness function, selection, crossover and mutation) is repeated till the user defined generation *G* is reached.

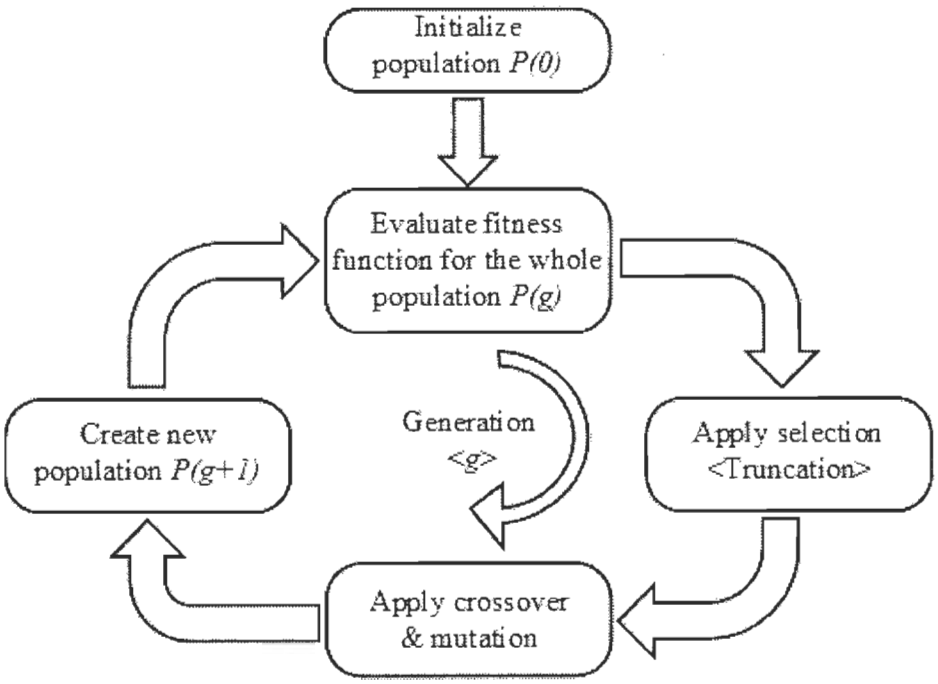


Figure 1 GA flowchart

Implementation Results

The GA based ARMA identification system [5] was implemented on the XILINX Spartan-6 xc6slx4-3tqg144 FPGA. A population of 16 chromosomes and a smoothing window size of B=16 were used. To test the robustness to the quantization effect, low fixed-point wordlength arithmetic environment was considered, where all arithmetic operations were implemented using the Q5 format (1 sign bit and 5 fractional bits).

The post place and route implementation results and the timing performance achieved are presented in Table 1 and Table 2, respectively. No DSP slice or block memory have been used in the design, LUTs were used instead for faster performance. The design used 83% of the total device logic, and was able to provide a processing rate of 320 Kilo Generation/second.

The signal processing performance analysis is done on the Signal to Quantization Noise Ratio (SQNR)

$$SQNR(n) = 10\log\left(\sum_{i=1}^{M} w_i^2 \Big/ \sum_{i=1}^{M}\left(w_i - \hat{w}_i(n)\right)^2\right)(\text{dB}) \tag{1}$$

where w_i is the exact coefficient, and $\hat{w}_i$ is the estimated coefficient. M is the number of coefficients, it has a value of $M = 7$ in this study.

To investigate the ability of the GA to successfully identify the ARMA coefficients, 10 independent realizations were performed.

In each realization, system signals are produced for randomly generated coefficients. This corresponds to the second scenario studied in [5].

Table 1 Post Place-and-Route implementation summary

Resource utilization	Available	Used
Slice registers	4800	889 (18%)
Slice LUTs	2400	1415 (58%)
Total Slices	498	600 (83%)

Table 2 Timing performance

Parameters	Rate
Maximum clock frequency	105 MHz
Maximum group rate	320 KGen/sec

Figure 2 shows the average SQNR of all realizations for the linear and the nonlinear identification problems. A theoretical SQNR is given as a reference metric, it is calculated by using the Q5 truncation quantization of the perfect floating-point coefficients, and it has a mean value of 26.9 dB.

The implemented GA converges to a steady state with an SQNR of 27.6 dB at generation 600 for the linear identification, it is 0.7 dB more than the theoretical SQNR.

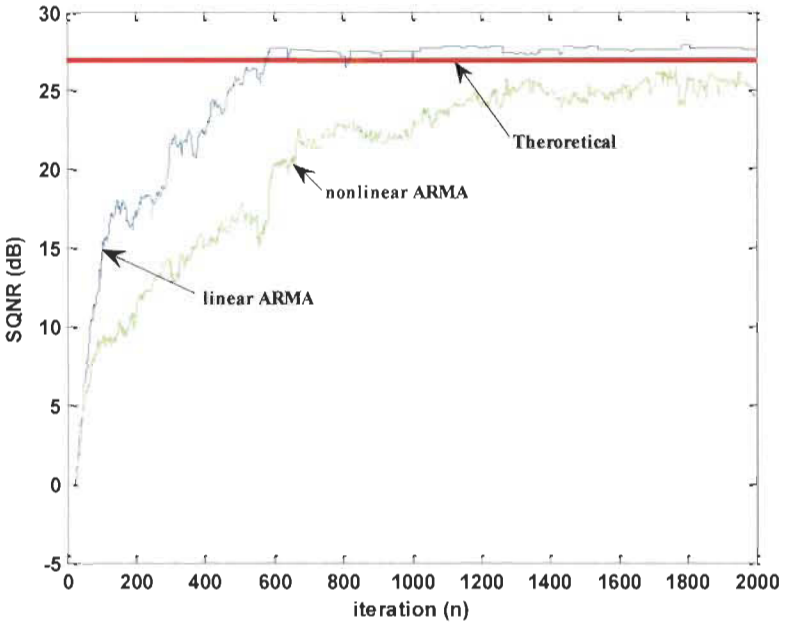


Figure 2 System identification performance results

This gain of performance is justified by the ability of the GA to perform rounding by altering the coefficients LSBs through the optimization process.

For nonlinear identification, the GA still capable of estimating the system parameters with a maximum SQNR of 24.7 dB, which is 2.2 dB below the theoretical SQNR and 2.9 dB lower than the performance achieved in linear identification. This is very interesting given the strong nonlinearity considered in the study. 1200 generations were required for the GA to converge to steady state.

Conclusion

An FPGA implementation of a GA for linear and nonlinear auto regressive moving average (ARMA) model parameters identification has been presented. The design considered low wordlength fixed-point arithmetic processing environement. The

implementation shows high signal processing performances and low resources cost, where only 6-bits worlength fixed-point arithmetic was used in all operations.

Acknowledgements

The authors wish to thank the National Sciences and Engineering Research Council of Canada (NSERC) for financial support.

References

[1] T. Cassar, K. P. Camilleri, and S. G. Fabri, "Order Estimation of Multivariate ARMA Models," *IEEE Journal of Selected Topics in Signal Processing*, vol. 4, pp. 494-503, 2010.

[2] V. Duong and A. R. Stubberud, "System identification by genetic algorithm," in *IEEE Aerospace Conference Proceedings*, 2002, pp. 5-2331-5-2337 vol.5.

[3] Cheng-Yuan, C. and C. Deng-Rui, "Active Noise Cancellation Without Secondary Path Identification by Using an Adaptive Genetic Algorithm," *IEEE Transactions on Instrumentation and Measurement*, 59(9), 2010, pp. 2315-2327.

[4] D. Massicotte and D. Eke, "High robustness to quantification effect of an adaptive filter based on genetic algorithm," in *IEEE Northeast Workshop on Circuits and Systems (NEWCAS),* 2007, pp. 373-376.

[5] H. Merabti and D. Massicotte, "Towards Hardware Implementation of Genetic Algorithms for Adaptive Filtering Applications," accepted in *IEEE Canadian Conference on Electrical and Computer Engineering (CCECE)*, Toronto, mai 2014.

Annexe B – Résultats additionnels

Abstract

Low bit-wordlength is a key solution in low power VLSI implementation. However, in adaptive filtering, the processing characteristics are lost in low fixed-point due to the recurrence in equation to update the filter coefficients. This paper presents a method based on genetic algorithm to update the filter coefficients in low bit-wordlength fixed-point implementation. Crossover and mutation techniques are proposed for fixed-point genetic algorithm. We compare the proposed method with most popular adaptive methods to identify the parameters of an ARMA system and to equalize a communication channel. Simulation results show the high robustness of the method in low fixed-point arithmetic where only 6-bit wordlength is used in all operations.

Introduction

Adaptive filtering is present in many signal processing applications such as identification, inverse problem, system modeling, or signal filtering [1]. From algorithm performance point of view, the recursive least mean square (RLS) outperforms the least mean square (LMS) adaptive filtering; however in fixed-point implementation point of view it becomes more sensitive to quantization effect in fixed-point operations and significant performance degradation is observed due principally to the covariance matrix. From the VLSI implementation point of view, if the fixed-point calculation has undeniable advantages in terms of circuit implementation (power consumption, small area of integration and speed of calculation), the signal processing characteristics (convergence, accuracy, error and asymptotic stability) are affected by the reduction of finite precision

calculations of the filter coefficients. The intrinsic recurrence of adaptive filtering methods (e.g. LMS, RLS) to update the filter coefficients requires more accuracy in fixed-point implementation and increases the bit-wordlength to reach the floating-point performance.

This paper deals with fixed-point arithmetic implementation, where the adaptive filtering is understood as an optimization problem rather than a problem of minimizing a cost function (usually based on gradient descent). As optimization technique, we considered the genetic algorithms (GA), method of the metaheuritic family [2]. Use GA as optimization technic for adaptive filtering is not new and largely shown in literature (e.g. [6]-[9]). Most of them are interested to compute the FIR, IIR filters parameters or other forms in a floating point manner, where no concern to work at low bit wordlength (ex: 6-bit).

Compared to our previous work [5], we improved the GA method (presented in Section 2) to increase the performance results in terms of convergence rate, stability and accuracy. In [5], the GA method was used considering min-max limit values applied on each coefficient to be estimated in identification system problem. In this present paper, a new crossover and mutation methods proposed eliminate this constraint in the process, and no min-max limits are applied. Thereby, using 6-bit and 8-bit as a low fixed-point filter, the identification parameters of an ARMA system and unlike our previous work [5], we demonstrated our improvement for channel equalization system too.

In this paper, we expose the following points; a description of GA method is done in Section 2. Section 3 details the applications for ARMA model and the follows by equalization problem in Section 4. Finally, we expose our conclusion in Section 5.

Adaptive filter based on GA

The genetic algorithm is basically a combination of five main operations: population initialization, fitness calculation, selection, crossover and mutation (Figure 1). Individuals forming the population are called chromosomes, each one represents the concatenated binary coded parameters, which in our case are the potential tap coefficients (w_M). The coding scheme case of wordlength, Q-bit, for 6-bit is as following:

w_1	w_2	$\cdots$	w_M
010010	111010	$\cdots$	010011

As a result, every parameter ranges in the interval [-1, $1-2^{-Q}$] (1 sign bit, and 5 fractional bits), leading to a chromosome searching space of ($2^Q M$) combinations. The chromosomes change through successive iterations (sample n), called generations.

In the beginning, every individual, N_P , of the population, $P(n)$, is initialized in a random manner. Then, a fitness function $f(n)$ is applied to each chromosome, and a fitness score is thereby given to every individual. This function is composed of one or multiple criteria depending of the problem to optimize. After to have applied $f(n)$ on all chromosomes, a selection mechanism selects the fitter individuals among the population according to their fitness score.

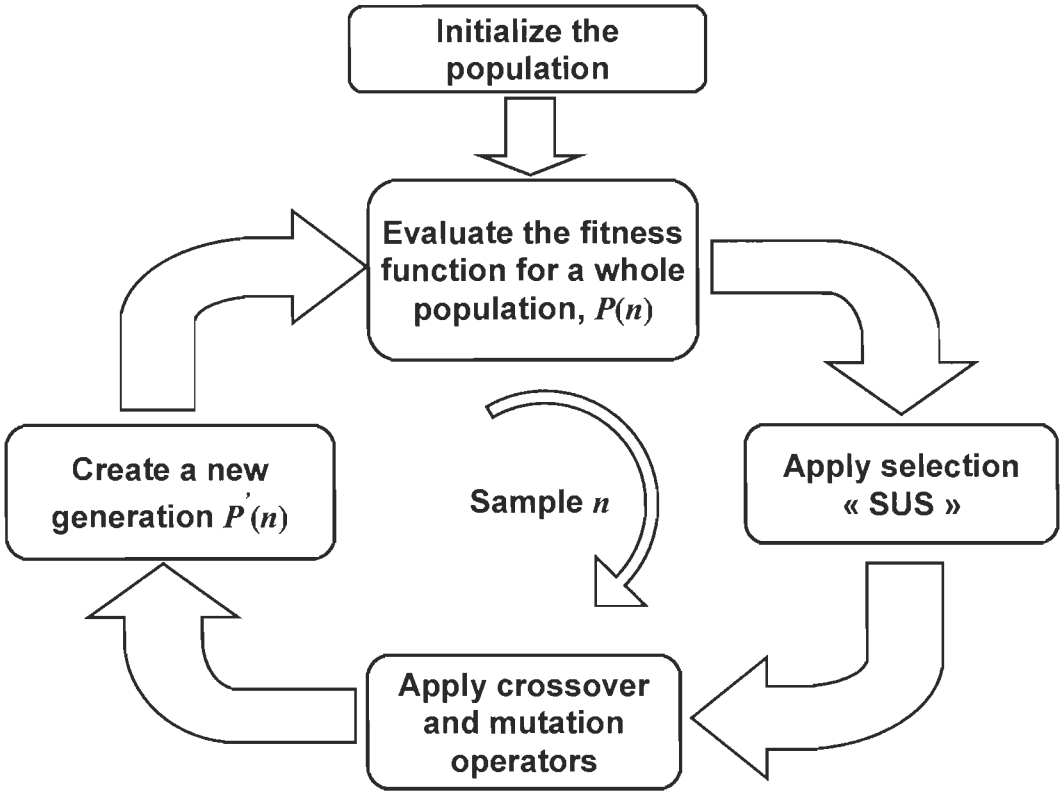


Figure 1 Main steps of genetic algorithm

The selected chromosomes will serve as a first pool of parents to the crossover operation, while the remaining chromosomes of the population form a second pool of parents. The selection process used in our work is based on the stochastic universal sampling technique (SUS) [2]. Next, one chromosome is picked up randomly from each pool. These parents are the chosen chromosomes for the crossing over. The crossover probability P_c was fixed at 100% during all the iterations.

Adaptive crossover probability using the fitness function can be used to enhance the convergence speed of the GA [6]. The idea behind this technique is to adjust automatically

some parameters of the algorithm for best overall performances. This principle is also used here, but with more simplicity and less computation complexity as explained below.

A new crossover approach is used in this paper and shown in Figure 2. First, the picked up parents are uniformly divided into M segments, representing the filter's coefficients. Then, a single segment is selected randomly from each parent. After that, a one point recombination is performed at the segment level. The position of the recombination point $Pos(n)$ is given in function of the best fitness score of the current population. The higher the fitness score is, the higher position of the crossover point is at the segment level. The function mapping fitness scores to crossover positions is totally linear. This move permits a guided research process instead of a totally random one. Improving results have been observed when compared to conventional crossover methods [5].

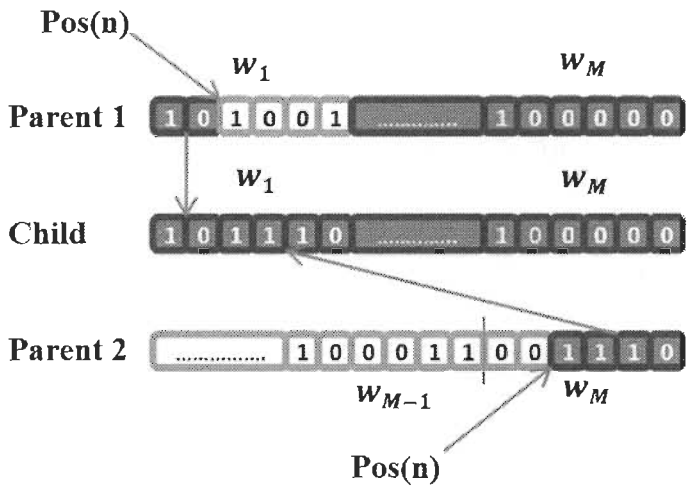


Figure 2 Proposed crossover operation

The last operation not the least, is the mutation operation. It's well known that mutation plays key roles in escaping the local optimization [3], which leads the GA to find global optimum values and converge quickly. During tests, we have observed that as the iterations

number increases, the mutation provides little new information to chromosomes which keep the GA stuck in local optimization. To overcome this drawback, we have adopted a variable mutation probability $P_m(n)$ based on the fitness score criterion as explained in the crossover section.

Mutation, shown in Figure 3, is applied to chromosomes resulting from the crossover operation (Offspring). One segment is selected randomly from an offspring, in Figure 2, segment corresponding to w_M is chosen. A random bit value is generated with equal bit wordlength that w_M. Then, a logical XOR operation is performed between the selected segment and the random value. The resulting parameter substitutes the original in the offspring. A single cycle is then completed, and a new generation of population, $P'(n)$, is created.

The process (fitness calculation, selection, crossover and mutation) is repeated each generation n until the specified number of total generations, G, or the optimal solution is reached. We have applied and evaluated the performances of the proposed GA based adaptive filter in adaptive filtering applications: system identification and channel equalization.

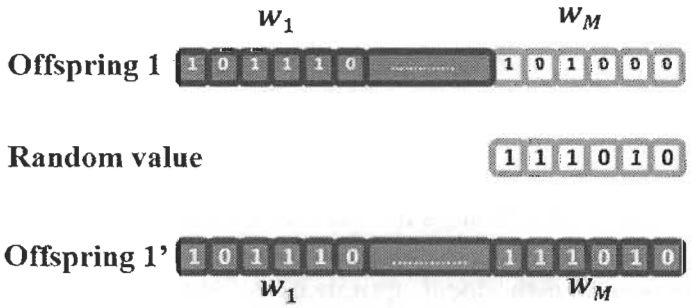


Figure 3 Mutation operation

Fixed-point adaptive filter –Identification

The performance study is done for an identification system described by the following ARMA model, Figure 4 [5]:

$$y_L(n) = \sum_{k=1}^{N_a} a_k y_L(n-k) + \sum_{k=0}^{N_b} b_k x(n-k) + \sum_{k=0}^{N_c} c_k u(n-k) \quad (1)$$

for linear system, and

$$y_{NL}(n) = tanh\left(v_{NL} y_L(n)\right) \quad (2)$$

for nonlinear system, where $y_L(n)$ is the output of the linear system; $y_{NL}(n)$ and v_{NL} are the output of the nonlinear system and the variance of the nonlinearity, respectively ; $x(n)$ is the input, $y(n)$ is the output and $u(n)$ is a noise signal. We suppose no correlation between $u(n)$ and the output signal. The input signal $x(n)$ is a random signal here, uncorrelated with $u(n)$. $u(n)$ is a white Gaussian noise with a zero mean and is 0.2 variance.

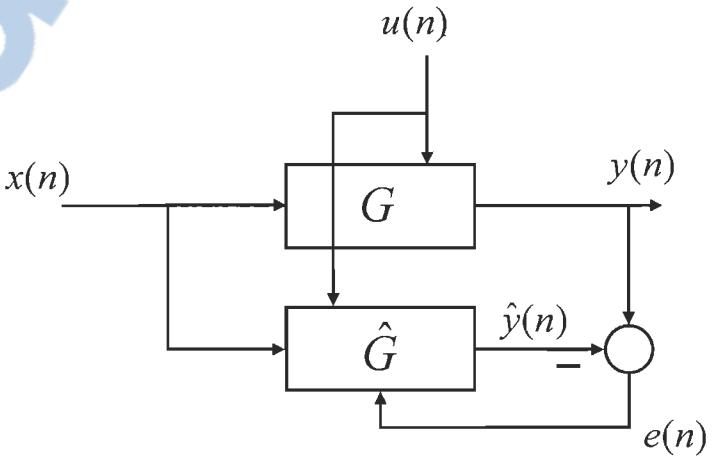


Figure 4 Block diagram for the system identification

The problem consists to identify the system coefficients $\{a_k\}, \{b_k\}$ et $\{c_k\}$ using the system input and output signals. The model ARMA has non zero coefficients for $a_1, a_2, b_5, b_6, c_0, c_1, \text{and}, c_2$.

To identify the ARMA parameters, we defined the input vector

$\mathbf{v}(n) = \left[y(n-1), y(n-2), x(n-5), x(n-6), u(n), u(n-1), u(n-2)\right]^T$, $\dim(\mathbf{v}) = M \times 1$ with the following vector used to estimate ARMA parameters $\mathbf{w}(n) = \left[w_1(n) \;\; w_2(n) \;\; \cdots \;\; w_M(n)\right]^T$ where M defines the number or ARMA parameters to identify (M=7).

The LMS filter is defined as following:

$$\hat{y}(n) = \hat{\mathbf{w}}^T(n)\mathbf{v}(n) \quad \text{with} \quad \hat{\mathbf{w}}(0) = \mathbf{0} \tag{3}$$

$$e(n) = y(n) - \hat{y}(n) \tag{4}$$

$$\hat{\mathbf{w}}(n+1) = \hat{\mathbf{w}}(n) + \mu\mathbf{v}(n)e(n) \tag{5}$$

For the GA, described in Section 2, we defined the fitness function method as:

$$f(n) = \sum_{m=0}^{B-1} \left| y(n-m) - \hat{y}(n-m) \right| \tag{6}$$

where, B is the block size of data over which we evaluate the filter and calculate the absolute error. Unlike the gradient methods where instantaneous error is used to update the filter coefficients, fitness function is based on summation of B absolute errors assuring stability of GA. Noted that, in low fixed-point we have no improvement of convergence when we replaced the equation (4) by (6) in the LMS or RLS method due to the fact that we increase the dynamic values of updating equation (5).

To compare the performance of the adaptive filter based on GA in low bit-wordlength, we compared with linear filter based on the LMS and RLS adaptive methods [1]. A learning rate of μ=0.01 was selected for the LMS. For all adaptive methods, all operations and signal are quantized to Q-bit wordlength in normalized fixed-point arithmetic. We studied two low bit-wordlength Q5 and Q7 with 1-bit was reserved for the sign and the rest 5-bit for the fractional part, corresponding to Q5 format.

To analyze the convergence curves, the mean of R repetitions is calculated. At each repetition, random ARMA coefficients are used to test the ability of the GA to converge regardless the coefficients values. The coefficients are generated randomly in the interval [0,1[for a_1, b_5, b_6, c_0, c_2, and [-1,0] for a_2, c_1. A population of N_P=20, and B=16 has been considered for GA.

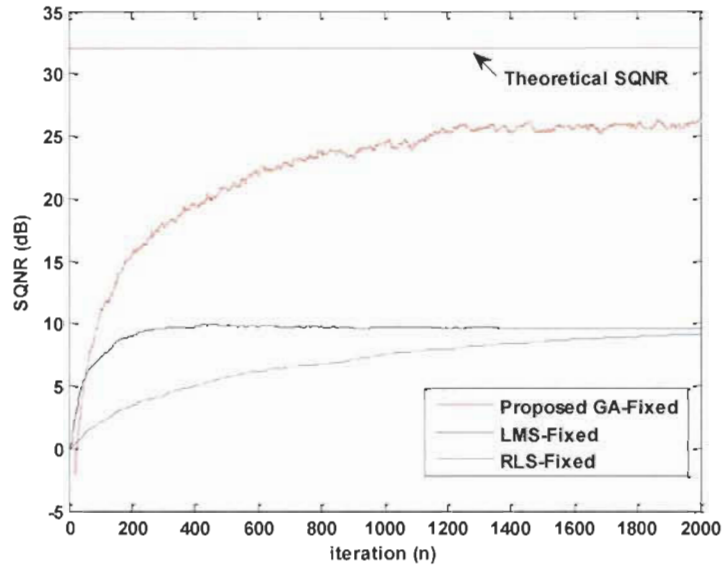
The analyses are done on the Signal to Quantization Noise Ratio (SQNR)

$$SQNR(n) = 10\log\left(\sum_{i=1}^{M} w_i^2 \Big/ \sum_{i=1}^{M}\left(w_i - \hat{w}_i(n)\right)^2\right)(\text{dB}) \tag{7}$$

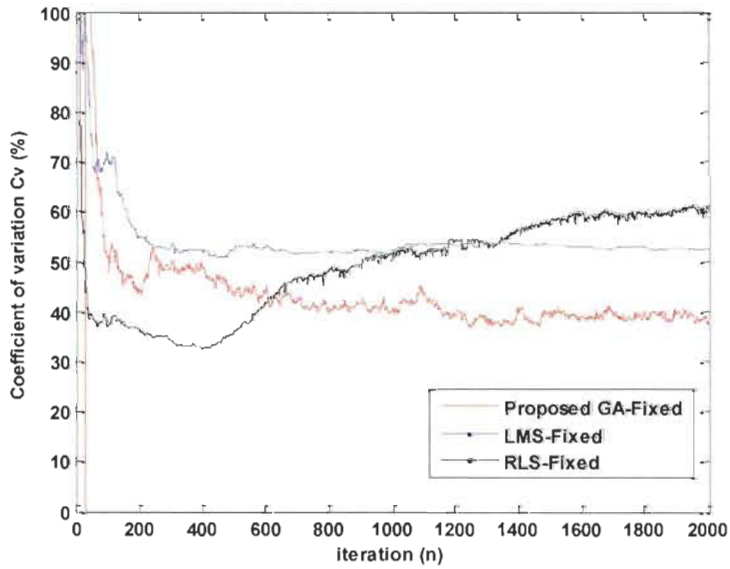
and to measure the repeatability of the adaptive methods, the coefficient of variation (C_v) is defined as

$$C_v(n) = \frac{\sqrt{\frac{1}{R-1}\sum_{i=1}^{R}\left(SQNR(n) - \frac{1}{R}\sum_{i=1}^{R} SQNR(n)\right)^2}}{\frac{1}{R}\sum_{i=1}^{R} SQNR(n)} \times 100\,(\%) \tag{8}$$

where R is the number of repetition.



a)



b)

Figure 5 Identification results (linear system) based on adaptive filter using 6-bit wordlength: a) SQNR and b) C_v.

As shown in Figure 5 for the linear system, the performance of the GA far surpasses those of the LMS and RLS in a 6-bits wordlength fixed-point system identifier. The GA reaches a SQNR of 26 dB in the last iterations while having a coefficient of variation around 40%. This is pretty good when considering randomly variable ARMA coefficients

to identify. In contrast, the LMS and the RLS show a saturate SQNR at 10 dB and a coefficient of variation of 50% for the LMS. The theoretical SQNR of 33 dB corresponds to the SQNR average of quantized coefficients corresponding to Q5 format.

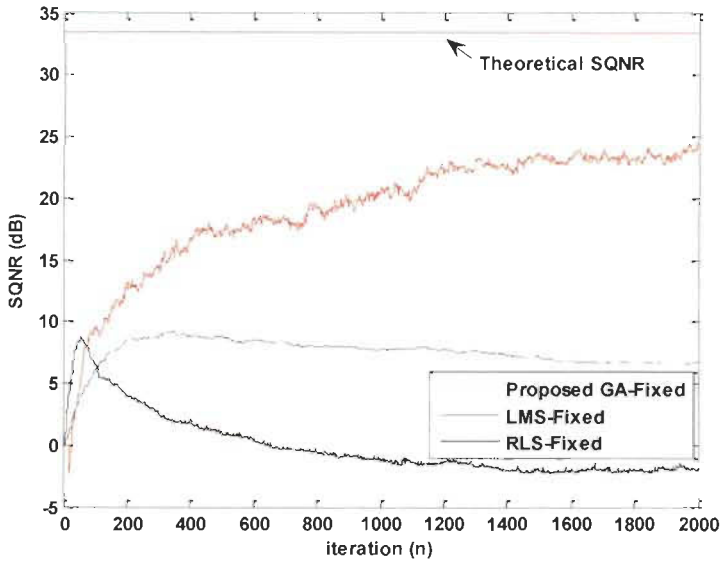


Figure 6 Identification results (nonlinear system) based on adaptive filter using 6-bit wordlength.

In the same manner, always considering the linear system, we repeat the study for 8-bit wordlength (Q7 format) for all methods keeping the same simulation parameters. Figure 7 shows the results the theoretical SQNR is around 45 dB, 12 dB more than 6-bit. We observe an SQNR of 35 dB at 2000 iterations for the proposed method compared to LMS and RLS which saturate at 15 dB, 5 dB more than 6-bit.

In both scenarios of low bit-wordlength, we observed a quasi linear progression of SQNR with iterations. The GA method keeps the same SQNR slope, for 6-bit and 8-bit scenarios, thus we have 25 dB at 2000 iterations.

To study the ability of the GA to perform well even when considering a nonlinear system, simulation results for the model represented by equations (1) and (2) with the nonlinearity coefficient $v_{NL} = 2$ are shown in Figure 6 for the Q5 format. The GA continues to perform almost as in the linear case while the LMS and RLS present significant performances degradation. This demonstrates the overwhelming benefit of using GAs in contrast to conventional adaptive filtering algorithms when addressing nonlinear problems.

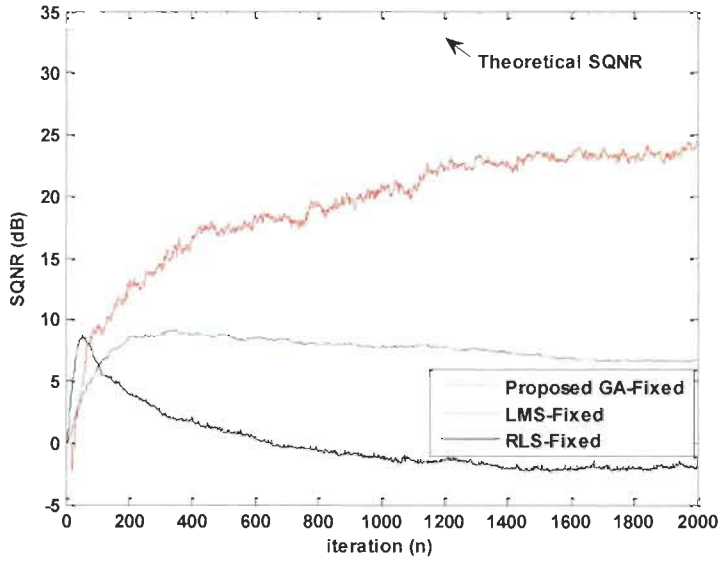


Figure 7 Identification SQNR results based on adaptive filter using 8-bit wordlength.

Fixed-point adaptive filter –Channel equalization

The performances study is done for a channel equalization system described in [1]. The channel impulse response h_n is given by

$$h_n = \begin{cases} \frac{1}{2}\left(1+\cos\left(\frac{2\pi}{W}(n-2)\right)\right) & n = 1,2,3 \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

The parameter W controls the amount of amplitude distortion produced by the channel. The adaptive equalizer input signal $v(n)$ is given by

$$v(n) = \sum_{k=1}^{3} h_k s(n-k) + u(n) \tag{10}$$

where, $s(n)$ is the system input signal, and $u(n)$ is a white Gaussian noise. The adaptive equalizer error signal $e(n)$ is given by

$$e(n) = s(n-D) - \hat{y}(n) \tag{11}$$

where D is the delay needed to provide the desired response for the equalizer, and M is the equalizer taps number.

For the GA, we defined the fitness function method as:

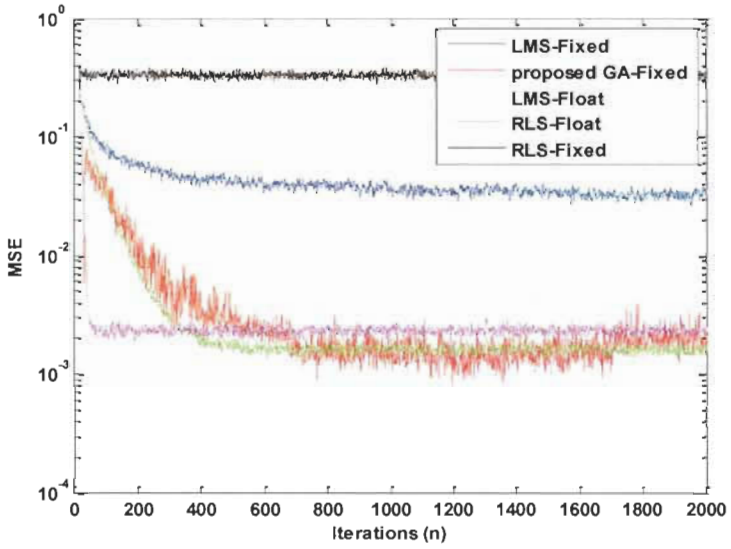
$$f(n) = \sum_{n-B+1}^{n} \left| s(n-D) - \hat{y}(n) \right| \tag{12}$$

As the previous case, again the fitness function is based on the summation of absolute error. In our simulations, we used W=2.9 for the channel, the input signal $s(n)$ is a random signal, with values {+1,-1}, $u(n)$ has zero mean Gaussian noise with a 0.001 variance, and the equalizer has M=11 taps with D=7.

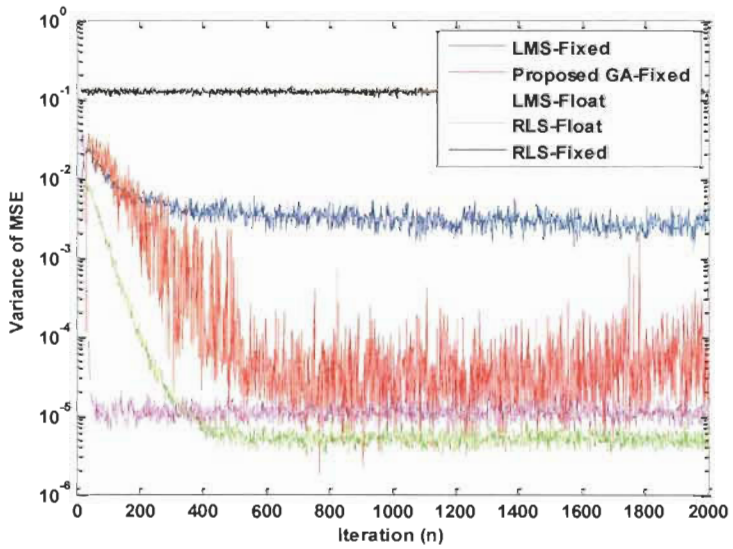
In LMS, RLS and proposed GA adaptive filter based method, all operations are performed with 6-bits wordlength fixed-point arithmetic using Q5 format, including the measured ARMA output signal $y(n)$. A population of N_P=20 and B=16 has been considered for GA.

To analyze the convergence curves, the mean of 400 repetitions is calculated. The analyses are done on the Mean Square Error (MSE) defined as following

$$MSE(n) = \frac{1}{R}\sum_{i=1}^{R}\left(\hat{y}_i(n) - s_i(n-D)\right)^2 \tag{13}$$



a)



b)

Figure 8 Equalization results based on adaptive filter using 6-bit wordlength: a) MSE and b) variance of MSE.

As shown in Figure 8, the proposed GA is performing well in a 6-bit wordlength fixed-point channel equalizer with an approximate MSE of 0.002 in the last iterations, reaching LMS and RLS MSE in floating point. In the last position, the RLS presents no sign of convergence, due to covariance matrix its high computation complexity which requires longer wordlength [5]. Figure 8b shows the repeatability of the adaptive methods defined by the variance of MSE on the 400 repetitions. We observed a superior repeatability of GA compared to other fixed-point method close to floating point methods.

Conclusion

Low fixed-point adaptive filter based on genetic algorithm has been presented. We have shown the robustness to low bit-wordlength of 6-bit and 8-bit in two well-known adaptive problems, identification of ARMA model and channel equalization. In both of them, our proposed method demonstrated robustness, convergence, stability, and accuracy. The proposed GA has far surpassed the conventional LMS and RLS adaptive filters. In identification parameters, we obtained SQNR close to the theoretical precision given by the quantified feedforward filter. In channel equalization, the GA using only 6-bit wordlength was able to perform as LMS and RLS in floaitong point representation. The future works will focus on the complexity analysis and parallel processing on FPGA implementation and possible application of the proposed GA to other adaptive filtering applications.

References

[1] S. Haykin, Adaptive Filter Theory, Prentice-Hall, 1996.

[2] J.E. Baker, "Reducing Bias and Inefficiency in the Selection Algorithm", ICGA, pp. 14-21, 1987.

[3] M.Gen and R.Cheng, "Genetic Algorithms and Engineering Design", Wiley-Interscience publication, 1997.

[4] H. Pohlheim, "Ein genetischer Algorithmus mit Mehrfachpopulationen zur Numerische Optimierung", at-Automatisierungstechnik, 3, 1995, pp. 127-135.

[5] D. Massicotte, D. Eke, "High Robustness to Quantification Effect of an Adaptive Filter based on Genetic Algorithm", IEEE NEWCAS/MWSCAS, Montreal, 2007, pp. 373-376.

[6] Cheng-Yuan, C. and C. Deng-Rui, "Active Noise Cancellation Without Secondary Path Identification by Using an Adaptive Genetic Algorithm."IEEE Transactions on Instrumentation and Measurement, 59(9), 2010, pp. 2315-2327.

[7] G. Geis, F. Gollas, R. Tetzlatt, " On the Implementation of Cellular Wave Computing Methods by Hardware Learning", IEEE International Symposium on Circuits and Systems, 2007, pp. 2930-2933.

[8] S.C Ng, S.H. Leung, C.Y. Chung, A. Luk, W.H. Lau, " The genetic search approach. A new learning algorithm for adaptive IIR filtering", IEEE Signal Processing Magazine, vol. 13, no. 6,1997, pp. 38-46.

[9] Russo, F and G. L. Sicuranza (2007). " Accuracy and Performance Evaluation in the Genetic Optimization of Nolinear Systems for Active Noise Control" IEEE transacations on Instrumentation and Measurement: 1443-1450.