METHODOLOGIE D'IMPLEMENTATION DES ALGORITHMES DE FILTRAGE ADAPTATIF SUR LES PROCESSEURS EN VIRGULE FIXE

R. ATAY, M. NAJIM

ENSERB, Equipe Signal et Images & GDR 134 - TdSI - CNRS
351, cours de la Libération, 33405, Talence CEDEX

RÉSUMÉ

L'implémentation des algorithmes rapides de Filtrage Adaptatif, en temps réel, sur les processeurs en virgule fixe et à architecture parallèle en pipeline requiert un important effort de mise en oeuvre. Deux types de problèmes sont à considérer: les problèmes numériques qui se posent lors de la quantification des algorithmes en précision finie et les problèmes liés à l'adaptation des différentes opérations à l'architecture tout en gardant une complexité minimale. Dans cet article† , nous explicitons les règles à respecter afin de réaliser la transformation en virgule fixe ainsi que la méthodologie d'implémentation des opérations de base.

ABSTRACT

The real time implementation of adaptive filtering algorithms on fixed point DSPs needs an important effort of development. Two kind of problems should be considered: first, numerical problems which occur when we have to quantifiy the algorithms in finite precision and second the problems linked with handling the different operating functions on specific architecture with respect of minimal complexity.

In this paper, we present the rules which should be respected in order to achieve the appropriate transformation in fixed point arithmetic and the implementation methodology needed for the broad base arithmetic operations.

I - INTRODUCTION

Au cours de l'implémentation d'un algorithme adaptatif, les signaux d'entrée du système adaptatif ainsi que les variables internes sont quantifiés avec une précision limitée. Cette quantification introduit des erreurs numériques. En outre, la structure récursive des algorithmes adaptatifs favorise l'accumulation et l'amplification de ces erreurs. Ceci provoque l'éloignement du système adaptatif de ses performances optimales, obtenues théoriquement en précision infinie. Cet éloignement conduit dans certains cas à la *divergence*, qui se traduit en général par des dépassements de capacité (*overflows*) ou par des blocages de l'adaptation dûs à l'extinction de certaines variables (*underflows*).

Les contraintes du temps réel conduisent à utiliser des processeurs spécialisés en traitement numérique du signal (DSPs) munis d'une architecture "partagée" ou "partagée modifiée" avec un parallélisme au niveau de la lecture/écriture des données et des unités fonctionnelles. Ces processeurs effectuent des calculs numériques et des transferts de données avec une grande vitesse. Par contre, pour tirer réellement profit de ces avantages, on doit veiller à l'allocation des variables en mémoire, à l'ordonnancement des opérations et à l'exploitation du parallélisme.

Le passage de la simulation d'un algorithme récursif en virgule flottante à la simulation de celui ci en virgule fixe est un travail long et difficile [ALCA 87, TRAV 89, BENA 89].
Par ailleurs, à chaque fois que l'on change les signaux en entrée ou que l'on modifie les paramètres d'initialisation, on est amené à reprendre le calcul de la dynamique et de la précision de chaque variable de l'algorithme ainsi que l'introduction des troncatures et des arrondis. Autrement dit, on doit reprendre toutes les étapes délicates de la méthodologie.

Dans ce travail, nous proposons une méthodologie qui consiste à simplifier la transformation en virgule fixe des algorithmes. Il s'agit de définir des règles à respecter, pour réécrire les opérations. Ainsi, ces règles permettront le passage systématique et sans intervention de l'utilisateur, de la représentation en virgule flottante à la virgule fixe. Après la définition de ces règles et leur validation sur les algorithmes adaptatifs, on pourra effectuer des simulations en virgule fixe sans se soucier des problèmes de la quantification.

II - REPRÉSENTATION EN VIRGULE FIXE

La représentation en virgule fixe se distingue de celle en virgule flottante par une dynamique de codage plus réduite et par une quantification grossière des variables. En général, les

† Ce travail entre dans le cadre de la convention CNET / GDR 134 - axe n° 2 "Algorithmes et architectures pour le filtrage adaptatif ".

nombres en simple précision sont codés en complément à 2 sur deux octets contigus.

Soit x un nombre représenté par L bits bi en complément à 2:

$$x = - b_0 + \sum_{k=1}^{L-1} b_k \, 2^{-k} \qquad (1)$$

$$= \sum_{k=0}^{L-1} \bar{b}_k \, 2^{-k} \qquad (2)$$

avec $$\bar{b}_k = \begin{cases} - b_0 & : \quad k = 0 \\ b_k & : \quad k \neq 0 \end{cases} \qquad (3)$$

Les valeurs possibles de x sont limitées par:

$$- 1 \leq x \leq +1 - 2^{-L+1} \qquad (4)$$

et la distance entre deux valeurs possibles de x est

$$\Delta = 2^{-L+1} \qquad (5)$$

III - OPÉRATIONS EN VIRGULE FIXE

Multiplication

La multiplication de deux nombres x1 et x2 et $\left(x_i = \sum_{k=0}^{L-1} \bar{b}_{i,k} \, 2^{-k} \right)_{i=1,2}$ donne:

$$x_1 \; x_2 = \sum_{k=0}^{L-1} \sum_{l=0}^{L-1} \bar{b}_{1,k} \bar{b}_{2,l} \, 2^{-k-l}$$

$$= \sum_{k=0}^{2L-2} \bar{c}_k \, 2^{-k} \qquad (6)$$

Troncatures et arrondis

Les opérations de troncatures et d'arrondis sont nécessaires lorsque l'on est amené à limiter la précision des nombres. Ces opérations sont requises notamment lors de l'utilisation du résultat d'une multiplication/accumulation comme opérande d'une opération ultérieure ou lors de son stockage en mémoire. Certains processeurs effectuent l'arrondi d'une façon systématique. Pour d'autres processeurs, cette opération doit être explicitée dans la programmation.

la troncature consiste à ignorer les bits qui dépassent la longueur du mot désiré:

soit $$x = -b_0 + \sum_{k=1}^{L-1} b_k \, 2^{-k} \qquad (7)$$

sa valeur tronquée à une longueur de Lt bits sera:

$$x_t = -b_0 + \sum_{k=1}^{L_t-1} b_k 2^{-k} \qquad L_t < L \qquad (8)$$

L'erreur de troncature peut être évaluée par:

$$v = x_t - x \qquad (9)$$

$$= - \sum_{k=L_t}^{L-1} b_k \, 2^{-k} \qquad (10)$$

L'utilisation de la quantification par troncature pour implémenter les algorithmes adaptatifs peut s'avérer très difficile du fait que l'accumulation de cette erreur, à moyenne non nulle, peut provoquer des problèmes d'instabilité numérique. C'est pour cela qu'on préfère l'arrondi comme méthode de quantification:

$$x_a = \left(x + 2^{-L_t}\right) + v \qquad (11)$$

Cette opération est réalisée dans les DSPs en additionnant la valeur binaire 1 au $(L_t + 1)^{ième}$ bit et en tronquant à L_t bits. Par ailleurs, les éléments b_k avec k > L ne sont pas nécessaires pour cette opération.

Addition et soustraction

L'addition ou la soustraction de deux nombres en virgule fixe est bornés par:

$$|x_i \leq 1| \qquad (12)$$

$$|x_1 \pm x_2| \leq |x_1| + |x_2| \leq 2 \qquad (13)$$

Ce résultat ne peut être représenté en virgule fixe car il peut dépasser la valeur 1. Comme nous l'expliquerons par la suite, ce problème doit être résolu par la mise à l'échelle et par des recadrages qui permettent d'éviter tout dépassement de capacité.

Décalages

Un décalage correspond en général à une multiplication par une puissance de 2 de la valeur d'un registre. Après cette multiplication, il arrive que le résultat ne soit pas représentable sur la précision du registre. Cela signifie qu'on perd un certain nombre de valeurs binaires. Les décalages se divisent en deux classes: le décalage à gauche qui correspond à une multiplication par 2 et le décalage à droite qui correspond à une division par 2. Un décalage à gauche ne cause pas de perte de précision, par contre il peut provoquer un dépassement de capacité. On peut éviter ce problème en effectuant une bonne mise à l'échelle et en respectant la règle suivante:

Un décalage à gauche de k bits est valide si:

$$|x \, 2^k| < 1 \; , \; k > 0 \qquad (14)$$

Le décalage à droite conduit à une perte d'information, parce qu'il nécessite une troncature. Par conséquent un biais est introduit par cette opération sur la valeur décalée si les bits tronqués sont différents de zéro. l'importance de ce biais dépend de la longueur du registre décalé. Un biais introduit sur un registre de longueur 2L est négligeable devant un biais introduit sur un registre de longueur L. C'est pour cela que l'on préfère, lors de l'implémentation, effectuer des décalages à droite sur des

accumulateurs en double précision (32 bits) que sur des registres en simple précision (16 bits).

III - RÈGLES D'IMPLÉMENTATION DES OPÉRATIONS EN VIRGULE FIXE

Afin d'exploiter au mieux la capacité numérique d'un processeur, il nous faut utiliser une représentation allouant un maximum de précision pour chaque variable sans pour autant provoquer de dépassement de capacité. A cet effet nous allons introduire des décalages sur toutes les variables de l'algorithme et nous appellerons cette représentation: "représentation des variables décalées". Pour effectuer ceci, nous avons besoin de connaître les bornes de chaque variable.

Recherche des bornes

L'évolution des variables durant l'exécution d'un algorithme adaptatif dépend des signaux d'entrée et des paramètres introduits dans le calcul par l'utilisateur. Avec un choix judicieux de ces paramètres, on peut obtenir les bornes des variables par des simulations en virgule flottante, par exemple, pour une variable numérique a:

$$\hat{a} = max_k |a(k)| \quad k = 0,1 \ldots K \qquad (15)$$

avec K suffisamment grand pour considérer que â ne changera pas de valeur pour k > K.

Les bornes obtenues par les simulations en virgule flottante représentent en général une bonne estimation des bornes des variables. néanmoins, au cours de l'implémentation, il se peut que l'on rencontre des dépassements de capacité. Dans ce cas, il faut reprendre les simulations en réajustant la représentation des variables décalées.

Mise à l'échelle

Soit a une variable bornée: $|a| \le \hat{a}$

le décalage S_a et la variable décalée $\tilde{a}$ peuvent être définis par:

$$S_a = 2^k \quad , \quad k \in N$$
$$\tilde{a} = S_a\, a \qquad (16)$$

on peut alors écrire:

$$|\tilde{a}| = S_a |a| \le S_a \hat{a} \qquad (17)$$

D'autres part, il n'est possible de représenter la variable décalée $\tilde{a}$ en virgule fixe que si:

$$S_a \hat{a} < 1 \qquad (18)$$

donc, pour respecter cette condition, le décalage doit satisfaire: $S_a^{-1} > \hat{a}$ et pour introduire les variables décalées dans l'algorithme nous utiliserons:

$$a = S_a^{-1} S_a\, a = S_a^{-1} \tilde{a} \qquad (19)$$

Validité de la mise à l'échelle durant l'exécution

A ce niveau, nous devons étudier les situations où il faut appliquer les décalages. Ces opérations ne sont pas nécessairement associées à des variables mais découlent de la mise à l'echelle de celles ci.

exemple:

$$c = a\, b$$
$$S_c^{-1} \tilde{c} = S_a^{-1} \tilde{a}\, S_b^{-1} \tilde{b}$$
$$\tilde{c} = \frac{S_c}{S_a\, S_b} \tilde{a}\, \tilde{b}$$
$$= S\, \tilde{a}\, \tilde{b}$$

Supposons que $\tilde{a}$ $\tilde{b}$ proviennent de calculs antérieurs et sont représentées d'une manière optimale†. Il s'agit d'introduire les décalages S_a, S_b, et S_c tout en essayant de conserver cette optimalité concernant les valeurs intermédiaires. Pour cela, des règles d'introduction des recadrages doivent être établies.

1 - si $\tilde{x}$ est représentée d'une manière optimale et $|\tilde{x}| \le 1$, alors un décalage S sur cette variable n'est valable que si $|S\, \tilde{x}| \le 1$ c'est à dire que si $S \le 1$.

$$S\, \tilde{x} \ \ valide\ si \quad S \le 1 \qquad (20)$$

2 - Durant l'exécution, supposons que l'on obtienne une expression de la forme $\tilde{x} = S\, .\, f()$, et que f() soit un résultat intermédiaire. $\tilde{x}$ étant optimalement représentée induit que f() ne l'est pas. Donc le décalage S doit servir à augmenter la valeur de f() avant de l'affecter à $\tilde{x}$. d'où:

$$\tilde{x} = S\, .\, f() \ \ valide\ si \ \ S \ge 1 \qquad (21)$$

nous développons les opérations principales qui interviennent dans le déroulement d'un algorithme dans une représentation en virgule fixe. En respectant les deux règles ci-dessus, nous définissons les règles d'introduction des recadrages dans le calcul.

IV - APPLICATION DES RÈGLES AUX OPÉRATIONS DE BASE

Variables intermédiaires

Avant d'appliquer les règles décrites ci-dessus dans une représentation en virgule fixe, il faut d'abord décomposer l'algorithme en une succession d'opérations de base. Cela nécessite la définition d'un certain nombre de variables supplémentaires permettant la jonction entre les opérations.

---

† Une variable est représentée d'une manière optimale en virgule fixe si elle est égale à la plus grande valeur décalée strictement inferieur à 1.

exemple:

$d = a + b\,c$ sera transformée en: $\begin{matrix} l = b\,c \\ d = a + l \end{matrix}$

la variable l ne figure pas dans l'algorithme original, pourtant, elle doit être étudiée du point de vue numérique et de la mise à l'échelle de la même façon que a, b, c ou d et les règles citées ci-dessus s'appliquent de la même manière à toutes les variables de l'algorithme y compris celles qui sont intermédiaires.

Addition

$$c = a + b$$
$$S_c^{-1}\,\tilde{c} = S_a^{-1}\,\tilde{a} + S_b^{-1}\,\tilde{b}$$
$$\tilde{c} = \frac{S_c}{S_a}\,\tilde{a} + \frac{S_c}{S_b}\,\tilde{b}$$

pour déterminer l'ordre d'introduction des recadrages, il faut reécrire cette expression de la façon suivante:

$$\tilde{c} = \frac{S_c}{S_a}\,\tilde{a} + \frac{S_c}{S_b}\,\tilde{b} \;:\; \frac{S_c}{S_a} \le 1 \; et \; \frac{S_c}{S_b} \le 1 \Leftrightarrow S_a,\, S_b \ge S_c$$
$$\tilde{c} = \frac{S_c}{S_a}\left(\tilde{a} + \frac{S_a}{S_b}\,\tilde{b}\right) \;:\; \frac{S_c}{S_a} \le 1 \; et \; \frac{S_a}{S_b} \le 1 \Leftrightarrow S_b,\, S_c \ge S_a$$
$$\tilde{c} = \frac{S_c}{S_b}\left(\frac{S_b}{S_a}\,\tilde{a} + \tilde{b}\right) \;:\; \frac{S_b}{S_a} \le 1 \; et \; \frac{S_c}{S_b} \le 1 \Leftrightarrow S_a,\, S_c \ge S_b$$

On obtient donc, pour chaque combinaison de valeurs S_a, S_b, et S_c un ordre de calcul qui respecte les deux règles (20 et 21).

Multiplication

la multiplication de deux variables décalées donne:

$$\tilde{c} = \frac{S_c}{S_a\,S_b}\,\tilde{a}\,\tilde{b} \qquad (22)$$

le produit $\tilde{a}\,\tilde{b}$ donne toujours un résultat inférieur à 1. Pour avoir un représentation optimale pour $\tilde{c}$, il faut que S_c soit supérieur à $S_a\,S_b$ ou encore:

$$S = \frac{S_c}{S_a\,S_b} \ge 1 \qquad (23)$$

et le calcul est effectué par: $\frac{S_c}{S_a\,S_b}\left(\tilde{a}\,\tilde{b}\right)$ (24)

Division

Pour effectuer la division d'une variable par une autre, on préfère d'abord inverser le dénominateur et ensuite le multiplier au numérateur. L'inversion d'une variable décalée donne:

$$\tilde{b} = S_a\,S_b\,\frac{1}{\tilde{a}} \qquad (25)$$

$\tilde{a}$ étant inférieur à 1, il faut que $S_a\,S_b$ soit aussi inférieur à 1 pour avoir une représentation sans débordement de $\tilde{b}$. Le calcul doit être effectué par:

$$\tilde{b} = \left(\frac{S_a\,S_b}{\tilde{a}}\right) \qquad (26)$$

Produit Scalaire

Le produit scalaire de deux vecteurs est en général exécuté au sein d'une boucle contenant une seule instruction de multiplication/accumulation en pipeline. Il est donc nécessaire d'éviter d'introduire des décalages supplémentaires dans cette boucle afin de conserver une complexité minimale. En adoptant une mise à l'echelle uniforme de toutes les composantes des deux vecteurs, le produit scalaire peut être effectué de deux façons:

$$\tilde{c} = \frac{S_c}{S_a\,S_b}\sum_i \tilde{a}_i\,\tilde{b}_i \;\; et \;\; \frac{S_c}{S_a\,S_b} \ge 1 \qquad (27)$$

ou bien $$\tilde{c} = \sum_i \left(\frac{S_c}{S_a\,S_b}\,\tilde{a}_i\,\tilde{b}_i\right) et \; \frac{S_c}{S_a\,S_b} \ge 1 \qquad (28)$$

La deuxième forme effectue le décalage du résultat de la multiplication avant d'effectuer l'accumulation, cette forme peut conduire à une importante augmentation de la complexité ainsi qu'à une perte de la précision. Nous préférerons toujours la première forme - avec une mise à l'echelle adéquate - d'autant plus que, sur les DSPs, l'accumulation se déroule dans un accumulateur en double précision.

V - CONCLUSION

Dans cet article, nous avons présenté une méthodologie d'implémentation des algorithmes en virgule fixe. Cette méthodologie est constituée un ensemble de règles à respecter pour obtenir la transformation des variables et l'adaptation des opérations de base qui constituent les algorithmes de filtrage adaptatif. Elle nécessite d'abord une simulation en virgule flottante pour déterminer les bornes et ensuite une simulation en virgule fixe integrant les règles décrites ci-dessus. Les étapes à suivre peuvent être résumées comme suit:

1 - Transformation de l'algorithme en opérations de base et introduction des variables intermédiaires.

2 - Détermination des bornes de toutes les variables.

3 - En déduire l'échelle des décalages associés à chaque variable.

4 - Détermination des autres décalages résultant de l'adaptation des opérations à l'environnement en virgule fixe.

5 - Ordonnancement des opérations et des recadrages suivant les règles d'implémentation.

Références

[ALCA 87] R. Alcantara, J. Prado, C. Gueguen, "Simulation sur ordinateur des effets de quantification dans las algorithmes de filtrage adaptatif". GRETSI, Juin 1987.

[BENA 89] A. Benallal "Etude des algorithmes des moindres carrés transversaux rapides et application à l'identification de réponses impulsionnelles acoustiques", Thèse de Docteur de l'Université de Rennes1, Janvier 1989.

[TRAV 89] J. M. Travassos-Romano et J. Strub, "Implantation en temps réel d'un algorithme de moindres carrés rapide sur microprocesseur de traitement du signal", Revue "Traitement du Signal", Vol. 6 - n° 1 - 1989.

[FRIE 92] M. Fries, R. Atay, M. Najim, "Implémentation of FTF algorithm on fixed point DSP",Proccedings of the COST 229 Workshop. 1992, Bordeaux.