Optimisation de la largeur des opérateurs arithmétiques pour la synthèse de haut niveau

Nicolas HERVÉ , Daniel MENARD , Olivier SENTIEYS

Équipe de recherche R2D2
IRISA/ENSSAT
6, rue de kérampont
22300 Lannion FRANCE
eMail : *name*@enssat.fr

Résumé – Les applications embarquées de traitement numérique du signal requièrent l'utilisation de l'arithmétique virgule fixe pour satisfaire les contraintes de coût et de consommation d'énergie. Pour réduire le temps de développement des applications, des outils de haut niveau permettant d'automatiser la conversion en virgule fixe sont nécessaires. Dans cet article, une nouvelle méthodologie d'implantation d'algorithmes au sein d'une plate-forme matérielle utilisant l'arithmétique virgule fixe est proposée. L'objectif de cette méthode est de minimiser la surface de l'architecture synthétisée sous contrainte de précision des calculs. Notre approche permet de coupler, à travers un processus itératif, l'optimisation de la largeur des opérateurs et la synthèse d'architecture.

Abstract – Most of embedded signal processing applications uses fixed-point arithmetic to satisfy cost and power consumption constraints. To reduce the time-to-market, high-level tools enabling fixed-point conversion automatically, are required. In this article, a new methodology for hardware implementation using fixed-point arithmetic is proposed. The objective is to minimise the chip area under a computation accuracy constraint. Our approach links the operator word-length optimisation and the architecture synthesis through an iterative process.

1 Introduction

Les architectures reconfigurables telles que les FPGA représentent un bon compromis entre un niveau de performance élevé et une flexibilité de développement. Dans un contexte de systèmes embarqués où les contraintes de coût, de consommation et d'encombrement sont primordiales [2, 3], l'implantation des algorithmes de traitement numérique du signal nécessite l'utilisation de l'arithmétique virgule fixe. En effet, les opérateurs en virgule fixe ne travaillent que sur des entiers alors que les opérateurs en virgule flottante doivent manipuler la mantisse et l'exposant associés à chaque donnée. En conséquence, la largeur des opérateurs arithmétiques, des bus et des mémoires est plus faible dans les architectures virgule fixe, ce qui permet de diminuer le coût en surface, le temps d'exécution des opérations et la consommation d'énergie. En contrepartie, l'utilisation de l'arithmétique virgule fixe implique de déterminer le codage de chaque donnée de l'application. Cette tâche de codage est longue, fastidieuse et source d'erreurs. De plus, la réduction du temps de mise sur le marché des applications exige l'utilisation d'outils de développement de haut niveau, permettant d'automatiser certaines tâches.

L'implantation efficace d'un algorithme au sein d'une plate-forme matérielle (ASIC, FPGA) nécessite de minimiser la surface et la consommation d'énergie. Ainsi, dans le cadre d'une implantation matérielle en virgule fixe, la stratégie est de déterminer la largeur des opérateurs permettant de minimiser la surface de l'architecture tant que la contrainte de précision associée à l'application est satisfaite. Un problème se pose du fait que, d'une part, la synthèse d'architecture nécessite la connaissance de la largeur des opérations et que, d'autre part, l'optimisation de la largeur des opérateurs requiert la connaissance de l'assignation des opérations aux opérateurs. En conséquence, pour atteindre au mieux l'objectif, le processus d'optimisation doit être couplé avec la synthèse d'architecture.

Uniquement la méthode proposée dans [5] réalise un couplage entre les processus de synthèse d'architecture et d'optimisation de la largeur des données. De plus, de nombreuses méthodes utilisent une technique d'évaluation de la précision basée sur la simulation conduisant à des temps de simulation prohibitifs.

Dans cet article, une nouvelle méthodologie de synthèse d'architecture sous contrainte de précision est proposée. Un processus itératif d'optimisation de la largeur des données et de synthèse d'architecture est mis en œuvre afin de combiner les effets de ces deux taches. De plus, l'évaluation de la précision est réalisée par une approche analytique permettant d'obtenir des temps d'optimisation beaucoup plus raisonnables qu'avec les approches basées sur la simulation. Dans la section 2, les principales méthodes de synthèse d'architecture en virgule fixe sont présentées. Notre méthodologie et les différentes phases la constituant sont détaillées dans la section 3. Enfin, pour montrer l'intérêt de notre approche, les résultats d'expérimentations sont fournis dans la section 4.

2 État de l'art

L'approche classique utilisée pour optimiser la largeur des données consiste à représenter toutes les données dans un format unique [6]. Ceci réduit l'espace de recherche à une seule dimension et simplifie la synthèse car toutes les opérations s'exé-

cuteront sur des opérateurs de même largeur. Cependant, dans le but d'obtenir une implantation efficace d'un algorithme de traitement du signal en virgule fixe, tout en respectant la contrainte de précision de calcul imposée par l'environnement, il est nécessaire de considérer un format propre à chaque signal [1].

La méthodologie proposée par Constantinides et *al* [1] se compose de deux étapes principales. La première permet d'obtenir une spécification virgule fixe respectant une contrainte de précision. La seconde, correspond à la synthèse de l'architecture. Cette méthodologie réalise l'implantation d'une spécification virgule fixe de l'application conduisant à une précision des calculs souvent nettement supérieure à la contrainte de précision. En effet, la seconde phase affectant des opérations sur des opérateurs de largeur plus importante, cela se traduit par une augmentation de la précision des calculs.

Dans [5], les auteurs proposent une méthode pour laquelle, la synthèse d'architecture est effectuée entre deux phases d'optimisation de la largeur des données. Une première étape analyse le graphe flot de signal de l'application pour former des groupes de données. Ensuite, une étape détermine la largeur de données minimale requise pour chaque groupe. Cette combinaison de largeur est utilisée dans une troisième étape de synthèse pour ordonnancer et assigner une spécification virgule fixe. La dernière étape recherche la largeur des opérateurs optimale. Enfin, le processus de synthèse et d'optimisation de la largeur des données doit être itératif et se terminer par une synthèse permettant d'implanter exactement la spécification virgule fixe optimisée pour la contrainte de précision donnée.

3 Synthèse sous contrainte de précision

La méthodologie d'implantation automatique d'algorithmes spécifiés en virgule flottante au sein de FPGA utilisant l'arithmétique virgule fixe est détaillée à la figure 1. Le code C virgule flottante représentant l'application est tout d'abord transformé à l'aide de l'outil SUIF en une représentation intermédiaire correspondant à un graphe flot de données et de contrôle. Cette représentation intermédiaire est ensuite convertie en un graphe flot de signal (GFS). Les différentes phases de conversion en virgule fixe et de synthèse de l'architecture sont effectuées à partir de ce GFS.

La première phase du processus de conversion en virgule fixe correspond à la détermination de la position de la virgule permettant de définir le nombre de bits associés à la partie entière de chaque donnée.Ensuite, la largeur des différentes données est déterminée et le nombre de bits pour la partie fractionnaire en est déduit.L'objectif de cette seconde phase est d'optimiser la largeur des opérateurs pour une contrainte de précision donnée, de manière à obtenir le coût (e.g. surface) le plus faible. Un environnement permettant d'effectuer conjointement l'optimisation de la largeur des opérateurs et la synthèse d'architecture est en cours de développement (c.f. fig. 1).

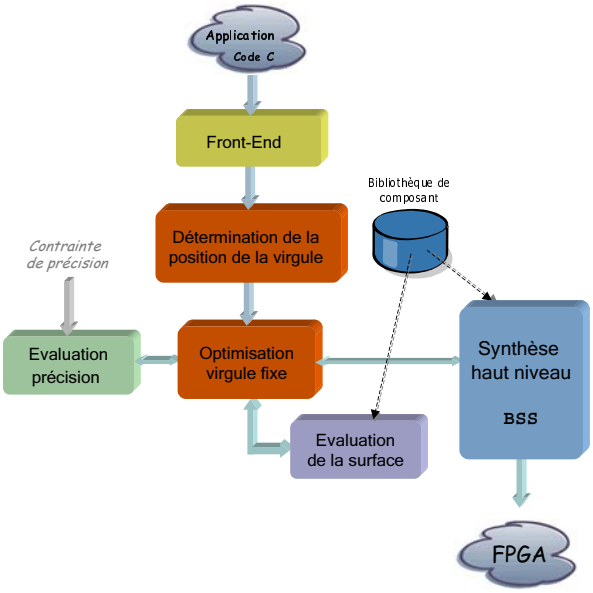

FIG. 1: *Environnement conjoint de synthèse d'architecture et d'optimisation des largeurs d'opérateurs*

3.1 Détermination de la position de la virgule

La première étape correspond à la détermination de la position de la virgule de chaque donnée. Le choix de cette position doit permettre de représenter toutes les valeurs prises par cette donnée (garantir l'absence de débordement) et minimiser le nombre de bits pour la partie entière de la donnée. Ce choix est réalisé à partir de la connaissance de la dynamique des données. Cette dernière est évaluée à l'aide d'une méthode analytique basée sur l'arithmétique d'intervalle [4] pour les systèmes non-récursifs [1] et la norme L1 [10] pour les systèmes linéaires invariants dans le temps.

Ces résultats sont directement utilisés pour définir la position de la virgule. Pour déterminer la position de la virgule des données en entrée et en sortie des opérateurs, des règles de propagation du format sont définies pour chaque type d'opération. Ensuite, ces règles sont appliquées à chaque opération lors d'un parcours du graphe flot de données (GFD) de l'application. Des opérations de recadrage sont insérées afin d'adapter le format de codage de la donnée à sa dynamique ou pour adapter le format des entrées d'un additionneur.

3.2 Synthèse d'architecture à largeurs multiples

La synthèse de l'architecture à largeurs multiples est effectuée après la détermination de la position de la virgule et va permettre de déterminer le nombre de bits nécessaires pour coder la partie fractionnaire de chaque donnée. L'objectif de cette phase est de minimiser la surface du circuit en réduisant la largeur des opérateurs. La contrainte de précision des calculs associée à l'application doit bien sûr être vérifiée. Ce processus itère tant que la contrainte de précision est vérifiée et doit évaluer la précision des calculs et la surface de la partie opérative à chaque itération.

Afin d'obtenir des temps d'optimisation du codage des don-

1. Dans une structure récursive, la sortie dépend des entrées et des échantillons précédents de la sortie. Pour une structure non-récursive, la sortie dépend uniquement des entrées.

nées raisonnables, une méthode analytique d'évaluation de la précision [8, 7] est utilisée. Cette approche détermine automatiquement l'expression analytique de la puissance du bruit de quantification et de la puissance du signal pour des systèmes linéaires invariants dans le temps ou non-linéaires non-récursifs.

L'outil de synthèse utilisé [9] s'appuie sur une bibliothèque d'opérateurs arithmétiques virgule fixe associée à une famille de FPGA donnée. Chaque opérateur est caractérisé en termes de surface, de temps d'execution et de consommation énergétique en fonction des différentes largeurs des entrées et de la sortie. La surface globale de la partie opérative est simplement la somme des surfaces de chaque opérateur.

3.2.1 Description du processus itératif

La méthodologie proposée pour la synthèse d'architecture à largeurs multiples est un processus itératif permettant de coupler efficacement la synthèse d'architecture et l'optimisation de la spécification en virgule fixe. Une itération du processus réalise successivement, le regroupement des opérations, l'optimisation de la largeur des groupes et la synthèse de l'architecture.

Pour commencer, la largeur propre de chaque opération est déterminée. Celle-ci est obtenue en optimisant la largeur des opérateurs pour une implantation spatiale. Dans ce cas, un opérateur est affecté à chaque opération.

Constitution de groupes d'opérations L'objectif de la première étape du processus d'optimisation est de constituer des groupes d'opérations. Un groupe rassemble les opérations d'un même type qui auront la même largeur. Tout d'abord, le nombre de groupes utilisés pour chaque type d'opérateur arithmétique est déterminé. Pour la première itération, tous les opérateurs d'un même type seront mis à la même largeur. Ainsi, un seul groupe est associé à chaque type d'opération arithmétique. Pour les itérations suivantes, le nombre de groupes sera défini à partir de l'analyse des résultats de la synthèse d'architecture.

Ensuite, un algorithme de groupement est appliqué afin d'affecter chaque opération à un groupe. Ce groupement des données est fait à partir de l'analyse des résultats de la synthèse d'architecture, de la largeur propre de chaque opération et des dépendances de données.

Notre technique de groupement se base sur la *mobilité* des opérations définie comme la différence des indices de cycle obtenus lors de deux ordonnancements par liste : un dans le sens directe et l'autre dans le sens inverse. Les opérations dont la mobilité est la plus faible sont traitées en premier et définissent les largeurs initiales associées à chaque groupe. La largeur d'un groupe donné correspond à la largeur maximale des opérations associées à ce groupe. Les opérations sont traitées ensuite par ordre décroissant de leur largeur propre. La mobilité de ces opérations est utilisée pour les déplacer en vue des les associer à un groupe disponible et de largeur la plus faible possible. L'objectif de cette approche est d'obtenir pour chaque groupe la largeur la plus faible.

Optimisation de la largeur des groupes d'opérations La troisième étape est l'optimisation de la largeur des différents groupes d'opérations. L'objectif est de minimiser la surface du circuit en respectant la contrainte de précision. Cette étape utilise les modules d'estimation de la précision et de la surface. L'algorithme d'optimisation se base sur la dérivée de la fonction de coût et de précision pour déterminer la direction à suivre.

Synthèse de l'architecture À partir de la spécification virgule fixe obtenue, une synthèse d'architecture est ensuite réalisée. Cette synthèse, effectuée à l'aide de l'outil BSS [9], va pouvoir remettre en cause le nombre d'opérateurs utilisés pour chaque type d'opération. En effet, par rapport à la synthèse précédente, la largeur de certains groupes d'opérations peut avoir diminué, entraînant également une réduction du temps d'exécution des opérations associées. Cette diminution peut conduire à une réduction du nombre d'opérateurs lors de la synthèse, ce qui justifie l'utilisation d'un processus itératif. En pratique, le nombre d'itérations est relativement faible. Le nombre de groupes utilisés pour l'itération suivante correspond au nombre d'opérateurs obtenus lors de cette phase de synthèse.

4 Expérimentations

Pour montrer l'intérêt d'une technique d'optimisation de la largeur des opérateurs en fonction d'une contrainte de précision des calculs, un filtre à réponse impulsionnelle infinie (IIR) d'ordre 8 a été testé. Ce filtre est implanté sous forme de 4 filtres cascadés d'ordre 2. Le graphe flot de signal de ce filtre récursif est présenté à la figure 3. Cette application est composée de 36 opérations correspondant à 20 multiplications et 16 additions.

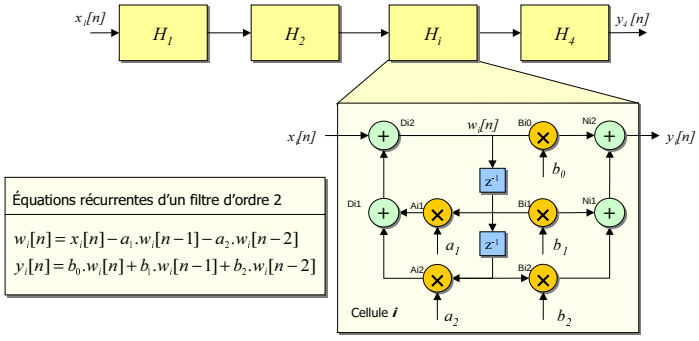

FIG. 3: *Synoptique du filtre IIR d'ordre 8 et graphe flot de signal d'une cellule d'ordre 2. Le filtre IIR d'ordre 8 est composé de 4 cellules d'ordre 2 cascadées.*

La méthodologie présentée dans cet article a été utilisée pour obtenir la dynamique et la position de la virgule de chaque donnée afin d'obtenir une spécification en virgule fixe correcte. De plus, l'expression analytique du rapport signal à bruit de quantification a été déterminée et la contrainte de précision a été fixée à 60 dB.

La largeur propre associée à chaque opération est présentée à la figure 2.a. Cette largeur est utilisée comme référence lors du processus de regroupement des données. Pour la première itération du processus global d'optimisation, un groupe est associé à chaque type d'opérateur et la largeur de celui-ci est fixée à la largeur maximale des opérations de ce groupe. Ainsi, les

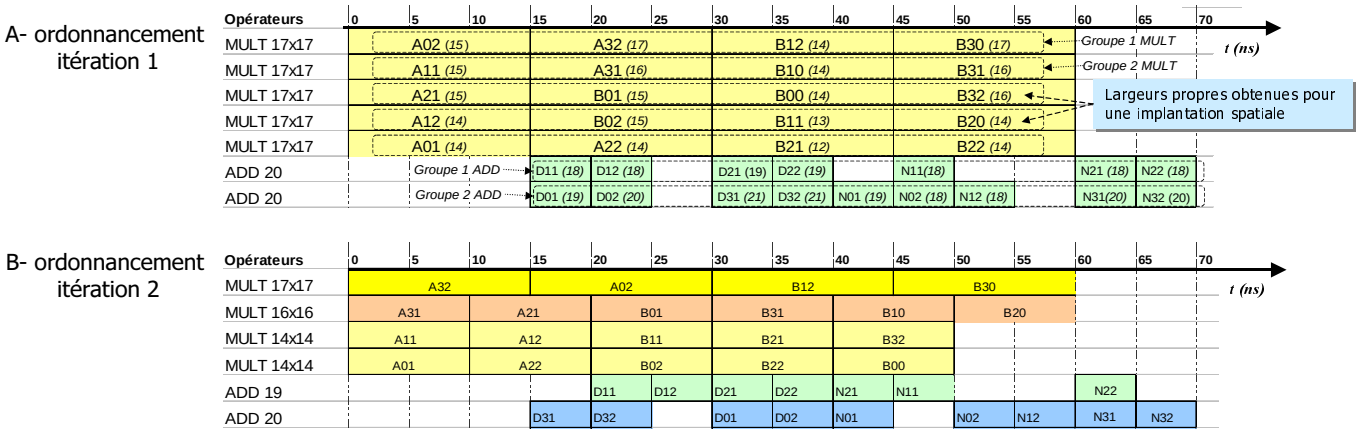

FIG. 2: *Ordonnancements et largeurs d'opérateurs obtenus pour le filtre IIR lors des deux premières itérations du processus.*

multiplications sont réalisées sur un multiplieur 17×17 bits et les additions par un additionneur 20 bits. La synthèse d'architecture pour cette spécification virgule fixe conduit à l'ordonnancement présenté à la figure 2.a. La fréquence minimale de l'horloge du système est fixée à 200 MHz. Ainsi, la latence de chaque opérateur est un multiple de $5\ ns$ Pour une contrainte de temps de $70\ ns$, 5 multiplieurs et 2 additionneurs sont nécessaires. Suite à cette synthèse, 5 nouveaux groupes sont définis pour les multiplications et 2 pour les additions. Ces groupes, représentés à la figure 2.a, sont formés en fonction de la largeur obtenue pour une implantation spatiale et de la mobilité des opérations.

Une optimisation de la largeur des groupes sous contrainte de précision est effectuée pour ces 7 groupes d'opérations. Cette optimisation permet d'obtenir des groupes de largeur plus faible. Les largeurs des 5 groupes pour les multiplications sont égales à 17, 16, 15, 14 et 14 bits et les largeurs pour les 2 groupes d'addition sont égales à 20 et 17 bits. La synthèse de l'architecture pour cette nouvelle spécification virgule fixe conduit à l'ordonnancement présenté à la figure 2.b. Les multiplieurs de largeur 14 à 16 bits ayant une latence plus faible ($10\ ns$), quatre multiplieurs seulement sont maintenant nécessaires. Ainsi, cette architecture utilise un multiplieur de moins que celle obtenue à la première itération. La réduction de la largeur de certains opérateurs combinée à la diminution du nombre d'opérateur, conduit à un gain au niveau de la surface des opérateurs de 35%.

La méthode proposée dans [6] réalise une architecture pour laquelle une largeur unique d'opérateur est utilisée. Pour cette application, 5 multiplieurs 19×19 et 2 additionneurs 19 bits sont donc utilisés. Par rapport à cette architecture, la surface des opérateurs obtenue avec notre approche est réduite de 47%. Ces résultats montrent l'intérêt d'utiliser des largeurs multiples pour les opérateurs et l'efficacité de notre approche.

5 Conclusion

Une nouvelle méthodologie de synthèse d'architecture pour FPGA sous contrainte de précision est décrite dans cet article. Pour obtenir une architecture optimisée en termes de largeur des opérateurs, un processus itératif d'optimisation de la largeur des données et de synthèse d'architecture est mis en œuvre afin de coupler ces deux processus. Pour obtenir des temps d'optimisation raisonnables l'évaluation de la précision est réalisée par une approche analytique.

Les premiers résultats obtenus avec cette approche montrent l'intérêt d'utiliser des largeurs multiples pour les opérateurs. Dans le cas du filtre IIR présenté, notre méthode permet tout en respectant la contrainte de temps et de précision de réduire la surface de la partie opérative de l'architecture de 47% par rapport à une approche classique pour laquelle les différents opérateurs possèdent la même largeur.

Ces travaux s'inscrivent dans le cadre du projet RNTL OSGAR (Outils de Synthèse Génériques d'Architectures Reconfigurables), en collaboration avec le CEA LIST à Saclay, l'UBO à Brest et la société TNI-Valiosys.

Références

[1] G. A. Constantinides, P. Y. K. Cheung, and W. Luk. *Synthesis and Optimization of DSP Algorithms*. Kluwer Academic, 2004.

[2] J. Eyre and J. Bier. The Evolution of DSP Processors. *IEEE Signal Processing Magazine*, 17(2):44–51, March 2000.

[3] G. Goossens and al. Embedded Software in Real-Time Signal Processing Systems: Design Technologies. *Proceedings of the IEEE*, 85(3):436–453, March 1997.

[4] R. Kearfott. Interval Computations: Introduction, Uses, and Resources. *Euromath Bulletin*, 2(1):95–112, 1996.

[5] K. Kum and W. Sung. Combined Word-Length Optimization and High-level Synthesis of Digital Signal Processing Systems. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 20:921–930, August 2001.

[6] E. Martin, J. Toureilles, and C. Nouet. Conception optimisée d'architectures en précision finie pour les applications de traitement du signal. *Traitement du Signal 2001*, 18(1):47–56, 2001.

[7] D. Menard, R. Rocher, P. Scalart, and O. Sentieys. SQNR determination in non-linear and non-recursive fixed-point systems. In *XII European Signal Processing Conference (EUSIPCO 2004)*, pages 1349–1352, Vienna, Austria, September 2004.

[8] D. Menard and O. Sentieys. A methodology for evaluating the precision of fixed-point systems. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2002)*, Orlando, May 2002.

[9] O.Sentieys, J.P.Diguet, and J.L.Philippe. GAUT: a High Level Synthesis Tool dedicated to real time signal processing application. In *University Booth, EURO-DAC*, Brighton, 18-22 september 1995.

[10] T.W. Parks and C.S. Burrus. *Digital Filter Design*. John Willey and Sons, 1987.