Optimisation de chemins de données par l'utilisation de l'arithmétique redondante

Sophie Belloeil et Habib Mehrez
UPMC/LIP6/ASIM
12 rue Cuvier
75252 PARIS cedex 05

E-mail: sophie.belloeil@lip6.fr

Résumé

Lors de la conception de chemins de données arithmétiques, l'utilisation de l'arithmétique classique (complément à 2) n'est pas toujours optimale en terme de performances.

Récemment, il a été proposé d'utiliser d'autres formes d'arithmétiques, telle que l'arithmétique redondante, conjointement aux systèmes classiques.

Cependant, la conception d'un chemin de données en redondant s'avère complexe pour un concepteur, d'où l'utilité d'un cadre de conception ad-hoc.

Ce papier présente une méthode permettant d'automatiser le recours à plusieurs systèmes arithmétiques dans la synthèse d'architecture de chemins de données arithmétiques.

1. Introduction

Depuis l'apparition des circuits intégrés, leur complexité n'a cessé d'augmenter. Plusieurs techniques permettant l'amélioration des performances des chemins de données par l'utilisation de plusieurs systèmes arithmétiques ont été proposées ([1], [3], [4]).

En particulier, Y. Dumonteix a étudié l'enchainement d'opérateurs arithmétiques et a proposé des optimisations structurelles pouvant être systématiquement appliquées. Il a ainsi montré l'intérêt des notations redondantes en mettant en avant les performances des opérateurs redondants/mixtes : l'implémentation d'un générateur de DCT (Discrete Cosine Transform) redondant [2] permet d'améliorer la fréquence de 54.7% avec une amélioration de consomation de 15% pour un surcoût de seulement 11% en surface.

Cependant, l'utilisation de l'arithmétique redondante dans la conception d'un chemin de données s'avère complexe pour des concepteurs n'ayant pas forcément le savoir-faire arithmétique nécessaire.

Un cadre de développement et une méthode sont donc nécessaires ainsi que le développement d'opérateurs arithmétiques, afin d'automatiser ce que nous appelons la synthèse arithmétique des chemins de données.

Nous présentons dans ce papier une spécification d'une méthode d'automatisation pour la conception de chemins de données arithmétiques mettant en oeuvre des optimisations apportées grâce à l'arithmétique redondante.

Nous spécifions tout d'abord notre méthode. Pui nous présentons les différentes optimisations pouvant être apportées à un circuit. La section suivante est consacrée à la synthèse arithmétique. Les mécanismes d'automatisation de la méthode sont ensuite exposés.

2. Méthode envisagée

Dans la conception de chemins de données arithmétiques, la multiplicité des représentations des opérandes rend complexes les choix du concepteur. En effet, les opérateurs admettent plusieurs représentations en entrée. Le choix de l'architecture d'un opérateur impose la représentation des opérandes en sortie et la représentation d'opérandes impose l'architecture d'un opérateur.

Dû à la complexité qui en découle, il nous semble important de proposer une méthode de conception permettant d'éviter au concepteur de choisir les opérateurs ainsi que les représentations des opérandes.

Cette méthode permet la mise en oeuvre automatique de l'arithmétique redondante et des règles d'optimisation que nous allons décrire.

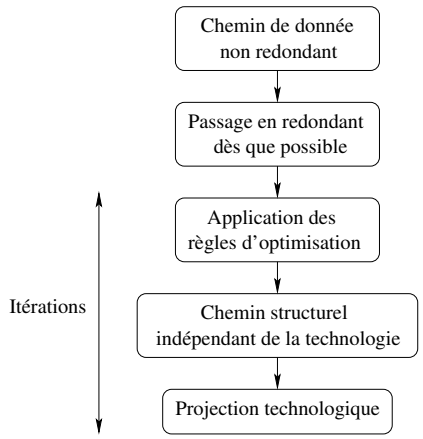


FIG. **1. Méthode**

Notre approche est de partir d'un modèle utilisant des représentations non-redondantes et d'introduire des opérateurs redondants/mixtes en effectuant des transformations locales sur le circuit (fig. 1).

Nous verrons par la suite que les différentes optimisations proposées n'engendrent pas toujours un gain. Il faut donc avoir la possibilité de corriger d'éventuelles contre-performances.

3. Etapes d'optimisation

3.1. Phase d'initialisation : Passage en redondant dès que possible

Des opérateurs redondants/mixtes sont instanciés partout où cela est possible (fig. 2 et 3).

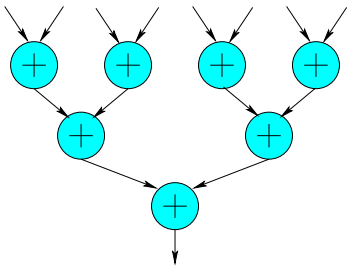


FIG. **2. Cellule non optimisée**

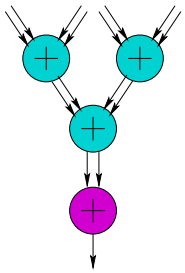
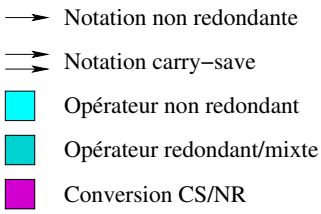


FIG. **3. Initialisation en redondant**



Une telle phase d'initialisation du circuit apporte des gains toujours positifs tant en surface qu'en délai grâce aux meilleures performances intrinsèques des opérateurs redondants.

En effet, les notations redondantes ont la particularité de restreindre la propagation des retenues de telle sorte que le calcul de l'addition est en temps constant.

3.2. Règles d'optimisation

Après cette phase d'initialisation, des optimisations sont encore possibles grâce à l'étude des enchaînements d'opérateurs et aux propriétés intrinsèques (commutativité, associativité) de certains opérateurs.

Regroupement Groupement d'opérateurs chainés effectuant la même opération (ex. : passage d'additions à une somme).

Fusion Absorption d'un premier opérateur par un second de nature différente (ex. : passage d'additions dans la somme d'un multiplieur).

Glissements Redistribution des ressources logiques de façon à dégager des portions arithmétiques plus importantes (ex. : déplacement de registres, multiplexeurs).

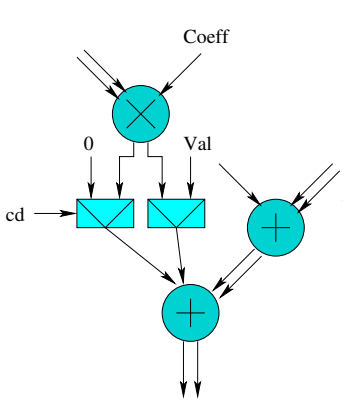


FIG. **4. Cellule**

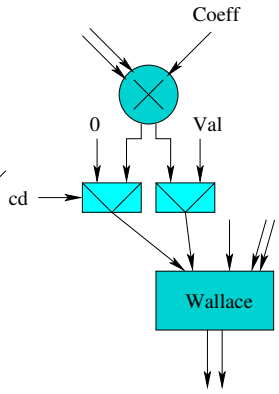


FIG. **5. Regroupement**

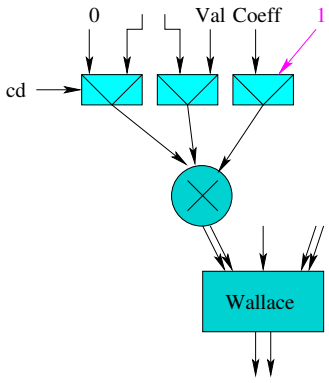


FIG. **6. Glissement**

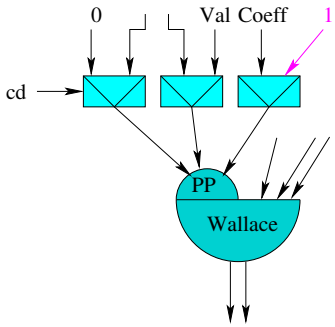


FIG. **7. Fusion**

Le regroupement (fig. 4 et 5) permet de tirer parti des qualités des réducteurs de Wallace. Le déplacement du multiplexeur (fig. 6 : glissement) se traduit par l'élargissement de la portion d'architecture purement arithmétique, ce qui permet d'exploiter d'autres optimisations (fig. 7 : fusion).

3.3. Application des règles

Pour être efficaces, les règles présentées requièrent une analyse des temps de propagation. En effet, elles dépendent très fortement du contexte et n'apportent pas toujours un gain.

Il faut donc mettre en place une méthodologie d'optimisation permettant de corriger d'éventuelles contre-performances.

4. Projection arithmétique

Nous nous placons dans un flot classique de conception VLSI. A partir d'une spécification fonctionnelle d'un circuit et d'un savoir-faire arithmétique, on obtient une description structurelle optimisée du circuit.

Notre approche est une projection d'une description structurelle vers une description "virtuelle" plus précise et optimisée de ce circuit.

Le concepteur doit spécifier :

- Les opérateurs par leur fonctionnalité et leur interface
- Les interconnexions entre opérateurs

– L'intervalle de valeurs et le mode de numération des entrées/sorties

Et laisse non spécifiés :

– L'architecture des opérateurs
– L'intervalle de valeurs et le mode de numération des interconnexions

Une desciption "virtuelle" est alors obtenue i.e. une description structurelle intermédiaire :

– Après spécification complète des différents opérateurs et des interconnexions
– Avant projection vers une technologie cible

Le principe de la synthèse arithmétique est de choisir les représentations des opérandes manipulées et les architectures des opérateurs employés tout en utilisant les règles d'optimisation présentées.

Une description du circuit est alors obtenue où les représentations des opérandes ainsi que les architectures des opérateurs sont spécifiées.

La suite du flot est inchangée : la projection vers une technologie cible peut être effectuée.

5. Mécanismes d'automatisation de la méthode

Automatiser la méthode présentée pose la double question de la forme de son encapsulation et de son utilisation.

L'encapsulation de la connaissance architecturale des opérateurs arithmétiques est assurée par le recours à des générateurs paramétrables.

Quant-à l'encapsulation des optimisations présentées, une première approche est de définir des modifications structurelles locales comme cela est présenté ci-après.

Cette première approche est basée sur une approche "bottom-up" [5] :

– Parcours de la cellule de sa sortie jusqu'aux entrées
– Recherche d'un ensemble d'opérateurs représentant un *motif*
– Remplacement de ce motif : application de transformations locales sur le circuit

Plusieurs parcours sont effectués si nécessaire jusqu'à ce que tout le chemin de données ait été optimisée. Le but est de remplacer un ensemble d'opérateurs par des opérateurs redondants avec de meilleures performances.

Le remplacement de motif s'effectue de la même manière pour la phase d'initialisation (fig. 8 à 11) et pour les différentes optimisations. Pour incorporer de nouvelles optimisations dans la méthode, il suffit de spécifier de nouveaux motifs.

Il est important de remarquer que les optimisations proposées dépendent du contexte. Le ou les critères améliorés ne sont pas toujours les mêmes et le gain peut s'avérer négatif. Pour être exploitables, ces méthodes doivent être encadrées par une méthodologie globale. Le choix de retenir ou non les modifications engendrées par les optimisations incombera de cette méthodologie.

6. Conclusion

Dans ce papier a été présentée la spécification d'une méthode permettant la gestion de plusieurs systèmes arithmétiques dans la conception de haut niveau des chemins de données arithmétiques.

La complexité dûe à l'utilisation de plusieurs systèmes arithmétiques pour un concepteur n'ayant pas le savoir-faire requis nous a amené à proposer une méthode de conception automatisant la mise en oeuvre des différents systèmes arithmétiques et des diverses optimisations pouvant être appliquées au circuit.

D'autres méthodes d'optimisation seront étudiées, en particulier lorsque le chemin de données est hétérogène et comprend aussi bien des blocs arithmétiques que logiques (registres, multiplexeurs).

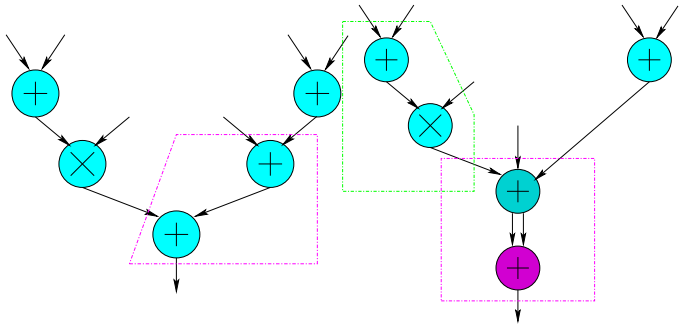


FIG. **8. Etape 1**

FIG. **9. Etape 2**

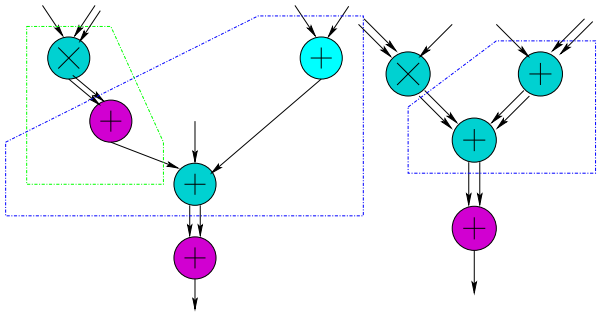


FIG. **10. Etape 3**

FIG. **11. Etape 4**

Références

[1] Yannick Dumonteix *Optimisation des chemins de données arithmétiques par l'utilisation de plusieurs systèmes de numération* Thèse de doctorat, Octobre 2001

[2] Roselyne Chotin, Yannick Dumonteix and Habib Mehrez *Use of Redondant Arithmetic on Architecture and Design of a High Performance DCT Macro-bloc generator* Proc. of 15th Conference on Design of Circuits and Integrated Systems, 2000

[3] Olivier Peyran *Synthèse d'architecures intégrées utilisant des arithmétiques redondantes* Thèse de doctorat, 1997

[4] M. Daumas et D.W. Matula *A booth multiplier accepting both a redundant or a non-redundant input with no additionnal delay* 2000

[5] Taewhan Kim, William Jao and Steve Tjiang *Arithmetic Optimization using Carry-Save-Adders* DAC 1998, pages 433-438