Métaheuristiques pour l'allocation de mémoire dans les systèmes embarqués

María Soto, André Rossi, Marc Sevaux

Université de Bretagne-Sud. Lab-STICC, CNRS UMR 3192
Centre de Recherche B.P. 92116
F-56321 Lorient Cedex FRANCE
{maria.soto, andre.rossi, marc.sevaux}@univ-ubs.fr

Résumé :

La gestion de la mémoire cache a un impact significatif sur les performances et sur la consommation énergétique des systèmes embarqués. Cet article traîte de l'allocation de mémoire des structures de données d'une application à la mémoire cache de manière à optimiser les performances du système. Les concepteurs de circuits souhaitent trouver un compromis entre le coût de l'architecture (le nombre de bancs mémoire à embarquer) et la consommation électrique. Le problème abordé consiste à allouer un banc mémoire à toute structure de données de manière à minimiser les conflits d'accès aux données. Le modèle proposé pour ce problème est le k-weighted graph coloring problem. Une formulation par PLNE et deux métaheuristiques basées respectivement sur Evocol et sur Tabucol sont comparées sur un ensemble d'instances. Les résultats obtenus sont encourageants et suggèrent que l'utilisation de méthodes issues de la coloration de graphes est une piste prometteuse pour l'allocation de mémoire dans les système embarqués.

Mots-Clés : *Allocation de mémoire, métaheuristiques, coloration de graphes*

1 Introduction

Les progrès continus des technologies en microélectronique ont rendu possible le développement de puces miniatures permettant d'étendre considérablement les fonctionnalités des appareils multimédia portatifs (la téléphonie mobile en est un exemple spectaculaire). Alors que la technologie ouvre des perspectives et des marchés toujours plus vastes, la conception des circuits intégrés devient, elle, de plus en plus complexe. D'autant qu'une innovation permanente sur des produits à bas coût s'accompagne naturellement d'une obsolescence rapide des produits fabriqués, qui contraint les concepteurs de circuits à minimiser le temps de conception.

Traditionnellement, les circuits intégrés étaient conçus à la main par des spécialistes qui maîtrisaient la technologie et la complexité du produit dans son ensemble. Une telle façon de faire n'est plus possible aujourd'hui, et des outils de conception assistée par ordinateur comme Gaut ont été introduit pour générer le design (ou l'architecture) d'un circuit intégré à partir de ses spécifications. Bien que les solutions produites par ce biais puissent être générées très rapidement, elles souffrent généralement d'un manque d'optimisation qui se traduit notamment par une consommation énergétique importante, ce qui est particulièrement préjudiciable pour les systèmes embarqués.

Cet article aborde la problématique de l'optimisation de l'allocation de mémoire cache des processeurs embarqués. En effet, la gestion de la mémoire cache a un impact significatif sur les performances et sur la consommation énergétique comme l'ont montré Wuytack *et al.*. Lorsque l'on conçoit un circuit intégré pour un système embarqué (typiquement un décodeur MPEG, un filtre numérique ou tout autre algorithme de traitement du signal et de l'image), l'application à synthétiser est fournie sous la forme d'un code source écrit en langage C, et les structures de données manipulées doivent être placées dans des bancs mémoire (il s'agit de la mémoire cache du processeur qui exécute l'application).

Pour des raisons technologiques comme pour des raisons de coût, le nombre des bancs mémoire est limité. On fait également l'hypothèse que le processeur est capable d'accéder à tous ses bancs mémoire simultanément, ce qui autorise plusieurs chargements en temps masqué. Ainsi, les structures de données (ou plus simplement les variables) `a` et `b` peuvent être chargées en même temps lorsqu'on souhaite effectuer l'opération `a+b`, à condition que `a` et `b` se trouvent dans deux bancs mémoire différents. Si ces variables partagent le même banc mémoire, le processeur ne peut y accéder que séquentiellement, ce qui nécessite deux fois plus de temps qu'un accès en parallèle. On dira que `a` et `b` sont en conflit si elles apparaissent dans la même opération. Tout conflit a un coût, qui est proportionnel au nombre de fois où l'operation `a+b` est effectuée dans l'application (ce coût peut être un réel, dans le cas où le code a été "profilé" par un outil logiciel basé sur une estimation stochastique de la probabilité de branchement des instructions conditionnelles). Le recours à une estimation du nombre de fois où une opération est effectuée se présente lorsque l'opération apparaît dans une boucle `while` par exemple.

Un conflit entre deux structures de données sera dit *ouvert* si les deux structures de données partagent le même banc mémoire, il est dit *fermé* si les deux structures de donnée ont été affectées à deux bancs mémoire différents.

Ainsi, les concepteurs de circuits souhaitent trouver un compromis entre le coût de l'architecture (le nombre de bancs mémoires à embarquer) et la consommation énergétique de l'ensemble. En effet, les électroniciens considèrent généralement que dans une certaine mesure, minimiser la consommation est équivalent à minimiser le temps d'exécution de l'application. De plus, la consommation d'une architecture donnée peut également être estimée à l'aide d'un modèle empirique. Ainsi, on a tout intérêt à paralléliser l'accès aux données de manière à minimiser le temps total d'exécution de l'application.

2 Conclusion

La bibliographie, la méthode exacte et les métaheuristiques développées pour ce problème sont présentées dans l'article long. La campagne de tests effectués avec GLPK, Tabucol, Evocol sera également présentée plus en détails au cours de la conférence.