EasyChair Preprint

№ 747

Génération de colonnes pour le partitionnement de Circuits Intégrés sur plate-forme multiFPGA

Lahcène Mezouari, Lilia Zaourar and Francois Galea

January 22, 2019

Génération de colonnes pour le partitionnement de Circuits Intégrés sur plate-forme multiFPGA

Lahcène Mezouari, Lilia Zaourar, François Galea

CEA, LIST, PC 172, F-91191 Gif-sur-Yvette Cedex, France
{lahcene.mezouari,lilia.zaourar,françois.galea}@cea.fr

Mots-clés : *partitionnement, hypergraphes, génération de colonnes.*

1 Introduction

Actuellement la vérification des System On Chip (SOC) devient difficile en raison de la complexité grandissante des modules matériels à tester. Pour cela, le prototypage matériel représente une solution intéressante. Cette technique consiste à réaliser un prototype du système sur un circuit logique programmable de type FPGA (Field Programmable Gate Arrays). Un FPGA désigne un Circuit Intégré (CI) composé d'un réseau de cellules programmables avec différentes ressources mais en nombre limité. De nos jours, les circuits complexes dépassent la capacité logique d'un seul FPGA, d'où la nécessité d'utiliser plusieurs FPGA pour le prototypage. Les communications entre FPGA sont très coûteuses par rapport à une communication intra-FPGA. Le problème que nous adressons ici est le partitionnement d'un CI sur plusieurs FPGA de façons à minimiser les connexions entre ces dernières. De plus, les contraintes en nombre de ressources logiques disponibles sont à respecter lors du partitionnement. Notre étude propose une modélisation mathématique et sa résolution par génération de colonnes pour améliorer la stratégie de partitionnement développé au CEA.

2 Modélisation mathématique

Nous utilisons une représentation sous forme de *netlist* du CI à vérifier. Une *netlist* est modélisée sous la forme d'un hypergraphe (V, E) dont l'ensemble des sommets $V = \{v_1, v_2, ..., v_N\}$ représente les cellules logiques, et l'ensemble des hyperarêtes $E = \{e_1, e_2, ..., e_M\}$ correspond aux fils du circuit. Nous modélisons l'hypergraphe par sa matrice d'incidence (H) telle que $h_{ij} = 1$ si la cellule i est reliée à l'hyperarête j, 0 sinon. Une première formulation de ce problème de partitionnement est présentée dans [4]. A partir de cette proposition, nous proposons une formulation en problème de génération de colonnes qui consiste en un Problème Maître (PM) ainsi qu'un sous Problème Auxiliaire (PA) [1]. La solution du (PM) détermine certains des paramètres du sous problème, tandis que le sous problème sera utilisé pour déterminer s'il existe des colonnes (partitions) pouvant entrer dans la base. Nous proposons alors le PM ci-dessous, inspiré des travaux de [3] pour la coloration de graphe. Dans notre cas, une couleur représente une partition avec les contraintes de capacité en plus. S est l'ensemble des partitions possibles tel que $|S| \leq 2^N$. N est le nombre de sommets. Les variables du problème maître sont $\sigma_s = 1$ si la partition s est utilisée, et 0 sinon. La fonction objectif consiste à minimiser le coût total de partitionnement tel que C_s est le coût de choisir la partition s. La contrainte (1) assure que chaque cellule est dans une seule partition choisie, la contrainte (2) oblige de prendre exactement k parties (FPGA). Les variables du (PA) sont $x_i = 1$ si le sommet i est dans la partition 0 sinon et les variables intermédiaires t_i le coût de choisir le sommet i dans la partition. Le Problème Auxiliaire utilise les solutions duales u_i et δ du problème maître pour générer chaque fois une colonne améliorante. Comme une colonne représente une partition des

sommets, elle doit donc respecter les contraintes de capacité de ressources (5) avec P_r la capacité de ressource r, R est le nombre de ressources et q_{ir} la quantité de ressource r consommée par la cellule v_i. Pour calculer le coût de choisir le sommet i dans la partition nous introduisons les contraintes intermédiaires (4) qui utilisent une constante F avec w_j le poids de l'hyperarête e_j et la condition suivante :

$$F \geq \sum_j h_{ij}(\sum_i (1 - x_i)h_{ij})w_j$$

$$\min \sum_{s \in S} C_s \sigma_s$$

$$(PM) \begin{cases} \sum_{s \in S : i \in s} \sigma_s = 1 & \forall i \in N \quad (1) \\ \sum_{s \in S} \sigma_s = k & (2) \\ \sigma_s \in \{0, 1\} \end{cases}$$

$$\min \sum_i t_i - \sum_i u_i x_i - \delta$$

$$(PA) \begin{cases} t_i \geq \sum_j h_{ij}(\sum_i (1 - x_{i'})h_{i'j})w_j - F(1 - x_i) & \forall i \in N (4) \\ \sum_i q_{ir} x_i \leq P_r & \forall r \in R (5) \\ x_i \in \{0, 1\} & \forall i \in N \\ t_i \in \mathbb{R}^+ & \forall i \in N \end{cases}$$

3 Résolution

Nous avons implémenté l'algorithme de génération de colonnes avec le framework SCIP [2] qui est efficace pour la résolution de programmation par contraintes et les algorithmes de génération de colonnes.

Dans notre cas, le problème auxiliaire a $2N$ variables et $N + R$ contraintes. Résoudre ce problème à chaque fois à l'optimum prend un temps très long puisque c'est un programme linéaire en nombres entiers (PLNE) (pour N de l'ordre de 2000 sommets). Afin d'accélérer cela, nous avons proposé une heuristique basée sur la résolution de la relaxation linéaire du problème auxiliaire. Nous avons également relaxé le problème maître en autorisant qu'un sommet soit dans plusieurs parties. Enfin, nous avons effectué des tests sur des instances réelles issues de circuits intégrés avec plusieurs milliers de sommets et une plate-forme composée de 4 FPGA.

Références

[1] G. Desaulniers, J. Desrosiers, and M. M. Solomon. *Column generation*, volume 5. Springer Science & Business Media, 2006.

[2] A. Gleixner, L. Eifler, T. Gally, and G. Gamrath. The SCIP Optimization Suite 5.0. Technical report, Optimization Online, December 2017.

[3] B. Yüceoğlu, G. Şahin, and S. P. van Hoesel. A column generation based algorithm for the robust graph coloring problem. *Discrete Applied Mathematics*, 217 :340–352, 2017.

[4] L. Zaourar and F. Galea. Graph Partitioning for System On Chip emulation on Multi FPGA Platforms. *5th International Symposium on Combinatorial Optimization*, 2018.