Optimisation du partitionnement de circuit pour des plateformes multi FPGA

Jean-Pierre Appéré[1], François Galea[2], Lilia Zaourar[2]

[1] Marine Nationale, Centre d'Expertise des Programmes Navals, section de Recherche Opérationnelle et Aide à la Décision, F-83800 Toulon Cedex 9, France
{jean-pierre.appere}@intradef.gouv.fr

[2] CEA, LIST, Laboratoire Calcul et Environnement de Conception, F-91191 Gif-sur-Yvette Cedex, France
{francois.galea, lilia.zaourar}@cea.fr

Mots-clés : *prototypage SoC, partitionnement multi FPGA, hypergraphe, solveur.*

1 Introduction

La réalisation de systèmes sur puces (SoC : System on Chip), se complexifie à mesure que la technologie permet une gravure fine et un nombre croissant de composants. Durant la phase de développement du système, le test du circuit par émulation informatique ne suffit alors plus. Il peut être effectué sur une plateforme de prototypage multi-FPGA.
Un FPGA est un circuit logique sur puce reconfigurable à l'envie mais possède des ressources limitées. La dimension importante d'un SoC moderne peut obliger à utiliser plusieurs FPGA pour un prototypage. Nous sommes alors confrontés à la gestion des interconnexions entre les FPGA en plus du respect des ressources disponibles par FPGA [1]. Notre étude propose une modélisation mathématique générique de ce problème et une approche de résolution par utilisation d'une heuristique de partitionnement générique et d'un solveur sur étagère prenant en compte les contraintes spécifiques.

2 Modélisation Mathématique

Nous utilisons la représentation sous forme de *netlist* du circuit à vérifier. La *netlist* étant la représentation du circuit sous forme de cellules correspondant aux ressources disponibles dans les FPGA et de leurs interconnexions. Elle est conçue pour le FPGA cible choisi.
Les interconnexions peuvent se faire entre plus de deux cellules, nous représenterons donc la *netlist* sous forme d'hypergraphe, chaque sommet représentant une cellule, et chaque hyperarête une interconnexion. Nous modélisons l'hypergraphe par sa matrice d'incidence (H) tq $h_{ij} = 1$ si la cellule i est reliée à l'hyperarête j.
Nous prenons en compte l'hétérogénéité des ressources du FPGA par un paramètre q_{ir} donnant le nombre de ressource r que consomme une cellule i. Un paramètre C_{kr} indiquera la capacité maximale d'un FPGA k pour une ressource r. La carte de prototypage impose un nombre limité de connexions physiques entre les FPGA. Le concepteur peut accepter de partager ces connexions entre différents signaux via du multiplexage, nous le modélisons par un coefficient Nb_{mux}. Nous proposons alors le modèle ($PM1$) ci-dessous où la pondération des hyperarêtes se fera par une fonction que le concepteur pourra ajuster. Notre objectif est de distribuer les cellules i dans les K partitions (4) en minimisant les hyperarêtes j coupées (1)(3). Une solution n'est alors réalisable que si l'on respecte les contraintes de capacité des FPGA (5) et si l'on respecte les contraintes d'interconnexion entre FPGA en tenant compte du multiplexage (2).
Nous avons introduit une variable intermédiaire y_{jk} valant 1 si l'hyperarête j est coupée et

possède une cellule dans k et 0 sinon. Le modèle devient quadratique mais nous proposons une adaptation en fonction du solveur utilisé.

$$Min \sum_{j=1}^{M} \sum_{k=1}^{K-1} \sum_{k'=k+1}^{K} y_{jk}.y_{jk'} \quad (1)$$

$$(PM1) \quad s.t. \begin{cases} \sum_{j=1}^{M} w_j.y_{jk}.y_{jk'} \leq Nb_{mux} & \forall k \neq k';\ k,k' \in [1;K] \quad (2) \\ y_{jk} \geq h_{ij}.h_{i'j}.x_{ik}.x_{i'k'} & \forall k \neq k', \forall i \neq i', \forall j = 1..M \quad (3) \\ & k,k' \in [1;K];\ i,i' \in [1;N] \\ \sum_{k=1}^{K} x_{ik} = 1 & \forall i = 1..N \quad (4) \\ \sum_{i=1}^{N} q_{ir}.x_{ik} \leq C_{kr} & \forall r = 1..R, \forall k = 1..K \quad (5) \\ x_{ik} \in \{0,1\} & \forall i = 1..N, \forall k = 1..K \\ y_{jk} \in \{0,1\} & \forall j = 1..M, \forall k = 1..K, \forall k' = 1..K \end{cases}$$

3 Résolution

Nous avons adapté le modèle à l'utilisation du solveur sur étagère « LocalSolver » [2] développé par l'entreprise Innovation24. Ses caractéristiques, optimisées pour les problèmes combinatoires de grande taille, sont bien adaptées à notre problème. LocalSolver utilisé seul peut mettre beaucoup de temps à trouver une solution réalisable, aussi nous procédons en deux étapes. Nous proposons de rechercher une solution initiale sur le problème relâché, c'est-à-dire de K-partitionnement seul. Cela peut se faire par une heuristique dédiée ou un logiciel de partitionnement d'hypergraphe de type hMetis [3] ou PaToH [4]. Cette solution est ensuite prise en entrée de notre modèle LocalSolver pour prendre en compte nos contraintes spécifiques.
Des résultats expérimentaux ont été réalisés pour des instances de l'ordre de quelques dizaines de milliers de cellules et d'hyperarêtes.

4 Conclusion et perspectives

L'utilisation d'un couple heuristique/solveur pour la résolution du problème de partitionnement multi FPGA peut apporter une solution réalisable à un concepteur. Cependant les circuits modernes possèdent plusieurs millions de cellules et hyperarêtes et notre validation n'a été faite que sur des instances de quelques dizaines de milliers. L'étape suivante des travaux consistera en un passage à l'échelle de la méthode.

Références

[1] Z. Marrachki, C. Alexandre and R. Farhat. *Overcome challenges in fpga-based prototyping.* EE Times-Asia, apr 2014.

[2] `http://www.localsolver.com/`

[3] Ümit V. Çatalyürek. `http://bmi.osu.edu/~umit/software.html`

[4] George Karypis `http://glaros.dtc.umn.edu/gkhome/metis/hmetis/publication`