Flot de conception automatisé pour la relocation de bitstreams sur FPGAs Xilinx

André LALEVÉE[1], Pierre-Henri HORREIN[1], Matthieu ARZEL[1], Michael HÜBNER[2], Michel JEZEQUEL[1]

[1]IMT Atlantique
Technopôle Brest-Iroise, CS 83818, 29238 Brest Cedex 3, France

[2]Ruhr-Universität Bochum - RUB
Universitäatstraße 150, 44801 Bochum, Germany

{andre.lalevee, ph.horrein, matthieu.arzel, michel.jezequel}@imt-atlantique.fr,
michael.huebner@ruhr-uni-bochum.de

Résumé – La reconfiguration dynamique partielle des FPGAs est une technique permettant la réutilisation de ressources logiques dans le cas d'applications pouvant être chargées à la demande sur le FPGA. Une fonction logique est ainsi remplacée par une autre pendant que le reste du système demeure inchangé. Pour cela, un bitstream partiel est alors chargé dans la SRAM de configuration du FPGA. Dans le cas d'applications nécessitant une grande modularité, le nombre de bitstreams partiels à générer et à stocker peut devenir problématique. La relocation de bitstreams permet de contrer ce problème, car un seul bitstream partiel n'est désormais nécessaire quel que soit le nombre de régions dans lesquelles une fonction doit pouvoir être implémentée. Cependant, ce procédé nécessite de respecter de fortes contraintes et requière généralement une connaissance intensive de la cible FPGA. Or, ceci n'est actuellement pas supporté par les outils de CAO. Dans cet article, un flot de conception complètement automatisé pour la relocation de bitstreams est présenté, ainsi que de nouvelles techniques nécessaires à cette automatisation.

Abstract – Dynamic and partial reconfiguration of FPGAs can enable logic resources to be reused in case of applications that are loaded on demand. A logic function is then swapped by another one while the remainder of the system remains unchanged. To achieve this, a partial bitstream has to be loaded in the configuration SRAM of the device. For applications that need high modularity, the number of partial bitstreams to be generated and stored can become problematic. Bitstream relocation can solve this problem, as it enables to use only one partial bitstream to configure one function in several regions of the FPGA. However, this needs to fulfill strong contraints, and often requires high knowledge of the FPGA fabric. Unfortunately, this has not been yet integrated in EDA tools. In this paper, a fully automated design flow for bitstream relocation is presented, as well as new techniques needed for this automation.

1 Introduction

La reconfiguration dynamique partielle (RDP) des FPGAs, par sa capacité à offrir la possibilité de modifier le comportement d'un circuit pendant son exécution, est devenue une solution envisageable pour améliorer la flexibilité des circuits numériques. En effet, cette technique offre entre autres des solutions de multiplexage temporel des ressources, ainsi que la possibilité de réaliser des circuits pouvant s'adapter au fur et à mesure de leur fonctionnement. Pour ce faire, il est couramment nécessaire de prédéfinir des zones du FPGA qui seront reconfigurables à l'aide de bitstreams partiels ne contenant que les informations concernant uniquement la zone à reconfigurer.

Cependant, cette technique présente plusieurs inconvénients. En effet, l'utilisation de la RDP requière en général des connaissances supplémentaires concernant la structure des FPGAs ainsi que des flots de conceptions dédiés. De plus, dans le cas de designs utilisant massivement cette technique, beaucoup de fonctions doivent toutes pouvoir être placées dans beaucoup de régions reconfigurables. Par exemple, dans le cas d'une plateforme FPGA dédiée au traitement du signal, il est fréquent d'avoir des opérateurs communs à différentes applications (filtres, FFTs, *etc*, ...) mais pouvant être séquencés dans un ordre différent en fonction de la chaîne de traitemant souhaitée. Dans ce cas, il est possible que ces opérateurs doivent pouvoir être implémentés dans n'importe quelles région reconfigurable prévue. Le nombre de bitstreams partiels à générer et à stocker peut rapidement alors augmenter.

La relocation de bitstreams est une technique permettant de palier ce problème du nombre de bitstreams. En effet, grâce à ce procédé, il est possible, selon certaines contraintes, d'utiliser un bitstream partiel généré pour reconfigurer une zone afin de configurer la même fonctionnalité dans une autre zone du FPGA, ce qui a le potentiel de fortement diminuer le nombre de bitstreams partiels à générer et stocker. Cependant, cette technique nécessite encore plus de connaissances de la cible et des outils que lors de la RDP classique, car il est nécessaire d'identifier des régions relogeables (*i.e.* compatibles pour la relocation), ainsi que de manipuler des informations contenues dans les bitstreams partiels. Ainsi, la relocation est généralement une technique difficile nécessitant beaucoup de temps, et propice aux erreurs, ce qui fait qu'elle est en général très peu utilisée.

Afin de rendre la relocation plus abordable, des efforts ont été effectués pour automatiser certaines étapes problématiques. Cependant, deux étapes restent non traitées : le floorplanning et le management des contraintes de timing aux interfaces des régions relogeables. Dans cet article, nous présenterons un flot de conception complètement automatisé pour la relocation intégrant des solutions à ces deux étapes, basé sur la suite ISE de Xilinx.

2 Contexte

Parmi les technologies FPGA existantes, la plus courante repose sur une couche de configuration basée sur de la SRAM, permettant des configurations rapides et volatiles. Une caractéristique intéressante des SRAMs est que chaque élément peut être accédé indépendamment des autres via son adresse. Cette propriété offre donc la possibilité de modifier certaines parties de la couche de configuration sans affecter le reste, c'est ce qu'on appelle la reconfiguration dynamique partielle.

De manière générale, les conceptions basées sur de la RDP sont séparées en deux parties distinctes : la partie statique, qui comprend toute la zone du FPGA qui ne sera jamais modifiée lors de l'exécution, et la partie dynamique, constituée de toutes les régions du FPGA qui pourront être modifiées. Chaque module reconfigurable pourra alors être hébergé par une région reconfigurable pour laquelle il a été implémenté via l'utilisation de bitstreams partiels.

Cependant, il peut arriver que certains modules puissent être implémentés sur plusieurs régions du FPGA. Pour chacun de ces modules, il sera alors nécessaire de générer un bitstream partiel par région reconfigurable. Par exemple, si M modules doivent pouvoir être implémentés dans N régions différentes, il sera nécessaire de générer $M \times N$ bitstreams partiels. Dans le cas où M et N sont grands, cela peut induire de longs temps d'implémentation, ainsi qu'un grand espace mémoire nécessaire pour stocker les bitstreams.

2.1 Relocation de bitstreams

La relocation de bitstreams permet d'utiliser un bitstream partiel généré pour une zone du FPGA pour configurer la même fonctionnalité dans une autre zone. Il est alors nécessaire de ne générer qu'un seul bitstream partiel par module quel que soit le nombre de régions dans lesquelles ce module doit pouvoir être implémenté. En reprenant l'exemple précédent, seulement M bitstreams partiels sont requis au lieu de $M \times N$. Ainsi, il est possible de réduire à la fois les temps d'implémentation et l'espace mémoire dédié au stockage des bitstreams partiels.

Cependant, de lourdes contraintes sont à respecter pour mettre en oeuvre cette technique. La plupart de ces contraintes sont résumées dans [1, 2], ainsi que des moyens de les respecter. Les principaux pré-requis présentés dans ces articles sont les suivants :

— les régions doivent être identiques en termes de taille et d'arrangement des ressources ;
— les interfaces partie statique/régions relogeables doivent être placées de manière identique ;
— le routage des connexions entre la partie statique et les régions relogeables doit être identique indépendamment des régions.

Cette dernière condition peut être respectée en empêchant la partie statique d'utiliser des ressources à l'intérieur des régions reconfigurables (via la contrainte PRIVATE disponible dans l'outil *PlanAhead* de Xilinx, (cette contrainte n'ayant pas encore d'équivalent sous Vivado, c'est la raison pour laquelle notre flot n'est compatible qu'avec ISE)) et en ajoutant des LUTs à côté de chaque *partition pin* (points d'interface entre partie statique et régions reconfigurables).

Une fois les bitstreams partiels générés, il est possible de les reloger en modifiant le *FAR* (Frame Address Register, *i.e.* l'adresse de départ de la zone considérée dans la SRAM de configuration) du bitstream considéré.

Bien que certains travaux récents [1, 2] permettent d'assurer la relocation, peu d'efforts semblent avoir été réalisés dans le but d'automatiser ce processus. GoAhead [3] est un outil supportant la relocation, cependant, au lieu d'assurer la possibilité de relocation à l'aide de contraintes spécifiques, l'outil ne fait que vérifier si une relocation est possible après implémentation, et regénère un bitstream partiel dans le cas contraire, ce qui ne laisse qu'un contrôle très limité au concepteur.

De plus, deux étapes primordiales restent non traitées : un algorithme de floorplanning adapté à la relocation, ainsi que le management des contraintes de timing aux interfaces des régions relogeables.

2.2 Floorplanning automatisé

Le but d'un algorithme de floorplanning est de partitionner efficacement le FPGA en une partie statique et des régions dynamiques.

Plusieurs algorithmes ont déjà été proposés dans la cadre de RDP classique [4, 5, 6, 7], cependant ceux-ci se révèlent tous inadéquats dans le cadre de designs relogeables. En effet, ces algorithmes ont pour but de limiter la fragmentation de la partie statique, ce qui a pour résultat de juxtaposer les régions reconfigurables. Cependant, étant donné que dans le cas d'une relocation, la partie statique ne peut utiliser de ressources à l'intérieur des régions relogeables, ce genre de solutions entraînent généralement une congestion de la partie statique. De plus, ces algorithmes ne prennent pas avantage du fait que toutes les régions doivent être identiques, ce qui limite fortement l'espace de recherche des solutions, et ainsi les temps de calculs.

3 Flot de conception automatisé

Dans cette section, notre flot de conception dédié à la relocation est présenté, ainsi qu'un nouvel algorithme de floorplanning dédié, et une méthode de gestion des contraintes de timing. Ce flot se veut facile d'utilisation, et ne nécessite pas de connaissances poussées en termes de circuits reconfigurables.

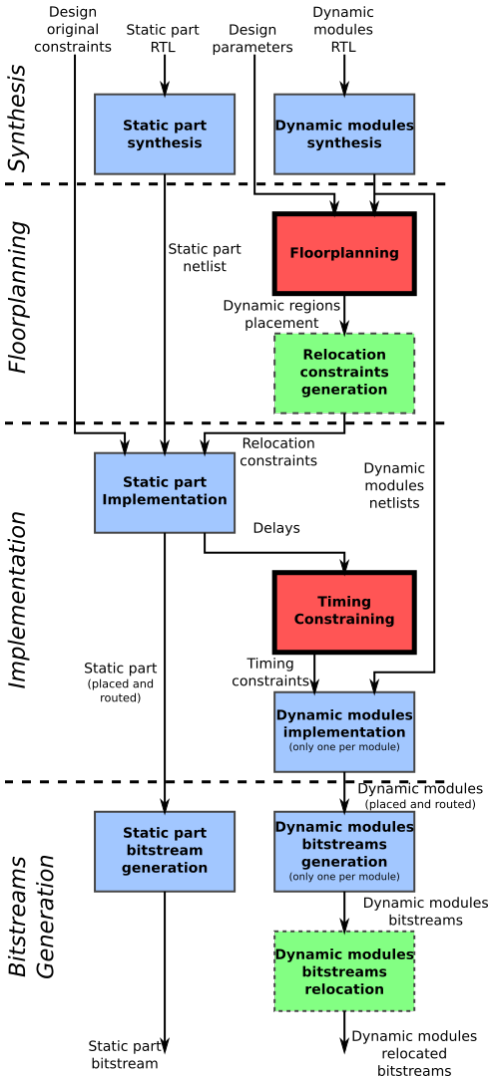

FIGURE 1 – Vue générale du flot de conception

Ainsi, ce flot est plus simple d'utilisation que la plupart des outils de RDP, tout en assurant la relocation de bitstreams.

3.1 Vue générale

Une vue générale de ce flot est donnée en figure 1. Sur cette figure, les étapes en vert à bord pointillé représentent les étapes détaillées dans la littérature mais jusqu'alors non intégrées dans des flots automatisés, celles en rouge à bord épais représentent les étapes introduites dans ce papier, et celles en bleu à bord simple représentent les étapes traditionnelles de RDP.

Ce flot prend en entrée :

— une description RTL de la partie statique
— une description RTL de chaque module reconfigurable
— un fichier de contraintes (contrairement à un design RDP classique, les informations concernant les régions reconfigurables sont superflues)
— un fichier de configuration contenant : le FPGA ciblé, son package et speed grade, le module top du projet, le nombre de régions relogeables souhaité et la liste des modules relogeables.

Il est intéressant de noter que, à part une structure où la partie statique doit être séparée des parties reconfigurables, toutes les autres entrées sont nécessaires dans le cas de designs classiques pour FPGAs. Une fois ces entrées fournies par le designer, une simple commande *make* suffit à générer toutes les sorties nécessaires sans aucune intervention.

3.2 Algorithme de floorplanning

Afin de trouver automatiquement un placement valide pour des régions relogeables, un nouvel algorithme de floorplanning a été développé. Étant donné que toutes les régions doivent être identiques en terme de taille et d'arrangement en ressources, cet algorithme est divisé en deux étapes : le choix du motif (*i.e.* la forme et l'arrangement en ressources) et la sélection des régions.

3.2.1 Choix du motif

La première étape de l'algorithme est de trouver un motif contenant assez de ressources pour accueillir les modules reconfigurables (une estimation des ressources est obtenue via les rapports de synthèse des modules dynamiques) et étant présent sur le FPGA un nombre de fois au moins égal à celui spécifié par le concepteur.

Certaines méthodes pour identifier des motifs sur un FPGA ont déjà été proposées dans la littérature. En particulier, la méthode proposée dans [8] permet d'identifier simplement un motif valide, cependant elle s'arrête au premier motif trouvé. Nous avons donc simplement itéré cette procédure en continuant la recherche sur la totalité du FPGA, tout en stockant tout les motifs valides.

Parmi tous les motifs retenus, ceux n'apparaissant pas assez de fois sur le FPGA sont écartés. Le motif final est alors choisi selon des critères pour l'instant empiriques (dans cet ordre : nombre de lignes minimal, plus grand nombre d'occurences, gaspillage de ressources minimal), cependant il est très difficile de définir un critère objectif pour cette étape.

3.2.2 Sélection des régions

Une fois les motifs identifiés, il faut alors sélectionner un sous-ensemble de taille n (n étant le nombre de région relogeables spécifié par le concepteur) parmi toutes les occurrences du motif sur le FPGA.

Cette sélection doit respecter deux contraintes. D'une part les régions ne doivent pas être trop éloignées les unes des autres, afin de ne pas engendrer de trop longs délais entre elles et ainsi ne pas dégrader les performances. D'autre part ces régions ne doivent pas non plus être trop proches, auquel cas la partie statique ayant besoin d'être placée entre ces régions risque de rencontrer des problèmes de congestion. Ainsi, notre algorithme doit placer les régions assez proches les unes des autres tout en laissant suffisamment de place à la partie statique.

Cette étape est réalisée en effectuant un recuit simulé minimisant la somme de la moyenne des distances entre régions et de leur écart type, tout en ajoutant une pénalité aux distances dépassant un seuil prédéfini, afin d'éliminer les placements de régions trop proches.

3.3 Contraintes de timing

En effectuant une relocation sans prendre de précaution, il est possible que les contraintes de timing, qui étaient valides pour la région d'origine, ne le soient pas pour la région de destination. En effet, bien que les délais à l'intérieur des zones reconfigurables restent inchangés après relocation (étant donné que les régions sont identiques), il n'y a aucune garantie que les délais entre la partie statique et les régions dynamiques soient identiques.

Pour résoudre ce problème, notre solution consiste à identifier, pour chaque pin d'interface des régions dynamiques, le délai maximum avec la partie statique parmi toutes les instances de régions relogeables, et à l'utiliser pour appliquer de nouvelles contraintes aux interfaces :

PIN "rp_inst_region_number.module_out<*pin_number*>" TPSYNC = regions_output_*pin_number* ;
TIMESPEC TS_from_RM_to_PP_output_*pin_number* = TO "regions_output_*pin_number*" (*clock_period* - *max_delay*) ns ;
PIN "rp_inst_region_number.module_in<*pin_number*>" TPSYNC = regions_input_*pin_number* ;
TIMESPEC TS_from_PP_input_to_RM_*pin_number* = TO "regions_input_*pin_number*" (*clock_period* - *max_delay*) ns ;

4 Premiers résultats

IP	#Slices	#BRAMs	#DSPs	Max frequency (Mhz)
DFT_8	1048	4	8	542.505
DFT_16	1470	5	12	542.505
Cordic_r_8_8_8	272	0	0	775.964
Cordic_v_8_8_8	305	0	0	475.884
Uniform_Generator	129	0	0	1402.328

TABLE 1: Estimation post-synthèse en ressources et fréquence maximale des modules testés

Des premiers tests ont été effectués et ont permis de valider notre approche sur les modules présentés en table 1 à une fréquence de 450MHz sur une cible Virtex7 690t. Les modules testés incluent : deux DFTs Spiral [9], deux opérateurs Cordic [10], et un générateur aléatoire uniforme 128 bits décrit dans [11]. Ils ont tous été relogés avec succès dans 4 régions différentes.

5 Conclusion

En conclusion, nous avons obtenu un flot de conception entièrement automatisé permettant d'obtenir relativement simplement des designs relogeables, basé à la fois sur la suite ISE de Xilinx, sur des techniques préexistantes dans la littérature, ainsi que sur de nouvelles méthodes spécialement adaptées à l'automatisation de la relocation.

Ce flot a été testé et validé sur des cas d'application pour l'instant très simples, et nous travaillons actuellement sur un cas d'application plus poussé afin de le valider pour des circuits plus complexes.

Références

[1] T. Drahonovsky, M. Rozkovec, and O. Novak. Relocation of reconfigurable modules on xilinx fpga. In *Design and Diagnostics of Electronic Circuits Systems (DDECS), 2013 IEEE 16th International Symposium on*, pages 175–180, April 2013.

[2] T. Drahonovsky, M. Rozkovec, and O. Novak. A highly flexible reconfigurable system on a xilinx fpga. In *ReConFigurable Computing and FPGAs (ReConFig), 2014 International Conference on*, pages 1–6, Dec 2014.

[3] C. Beckhoff, D. Koch, and J. Torresen. Go ahead : A partial reconfiguration framework. In *Field-Programmable Custom Computing Machines (FCCM), 2012 IEEE 20th Annual International Symposium on*, pages 37–44, April 2012.

[4] L. Singhal and E. Bozorgzadeh. Multi-layer floorplanning on a sequence of reconfigurable designs. In *Field Programmable Logic and Applications, 2006. FPL '06. International Conference on*, pages 1–8, Aug 2006.

[5] A.M. Smith, G.A. Constantinides, and P.Y.K. Cheung. Integrated floorplanning, module-selection, and architecture generation for reconfigurable devices. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 16(6) :733–744, June 2008.

[6] Alessio Montone, Marco D. Santambrogio, Donatella Sciuto, and Seda Ogrenci Memik. Placement and floorplanning in dynamically reconfigurable fpgas. *ACM Trans. Reconfigurable Technol. Syst.*, 3(4) :24 :1–24 :34, November 2010.

[7] C. Bolchini, A. Miele, and C. Sandionigi. Automated resource-aware floorplanning of reconfigurable areas in partially-reconfigurable fpga systems. In *Field Programmable Logic and Applications (FPL), 2011 International Conference on*, pages 532–538, Sept 2011.

[8] T. Becker, M. Koester, and W. Luk. Automated placement of reconfigurable regions for relocatable modules. In *Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on*, pages 3341–3344, May 2010.

[9] Spiral website. http ://www.spiral.net/hardware/dftgen.html.

[10] Opencores website. http ://opencores.org/.

[11] David B Thomas and Wayne Luk. High quality uniform random number generation using lut optimised state-transition matrices. *The Journal of VLSI Signal Processing Systems for Signal, Image, and Video Technology*, 47(1) :77–92, 2007.