UNIVERSITE SAAD DAHLAB DE BLIDA

Faculté de Technologie

Département d'Electronique

THESE DE DOCTORAT

en Electronique

GESTION DE LA RECONFIGURATION DYNAMIQUE PARTIELLE DANS UN SYSTEME SUR PUCE RECONFIGURABLE

Par

Maamar TOUIZA

Devant le jury composé de:

H. SALHI	Professeur, Université Saad Dahlab, Blida	Président
A. FERDJOUNI	Maître de conférences, Université Saad Dahlab, Blida	Examinateur
M. OULD-ZMIRLI	Maître de conférences, Université Yahia Fares, Médéa	Examinateur
L. MITICHE	Maître de conférences, Université Ziane Achour, Djelfa	Examinateur
E. BOURENNANE	Professeur, Université de Bourgogne, Dijon, France	Co-Directeur
A. GUESSOUM	Professeur, Université Saad Dahlab., Blida	Directeur

Blida, 2014

ملـــــخص

تقدم شرائح الFPGA من نوع VIRTEX XILINX ميزة إعادة التشكيل الجزئي (DPR). يمكن توطين المهام داخل الشريحة واخراجها دون مقاطعة شغل النظام. في المقابل ، فإن وضع اي مهمة في فضاء الشريحة يتطلب بيتستريم (BITSTREAM) جزئي فريد خاص لكل منطقة قابلة للتشكيل مما يؤدي إلى زيادة الحاجة للذاكرة لتخزين البيتستريمات الجزئية المختلفة.

من أجل إيجاد حلول لهذه القيود و زيادة قدرة عملية إعادة التشكيل الجزئي ، تم اقتراح عدة مناهج لإعادة توطين البيتستريمات توصف في أغلبيتها كإمتداد لمراحل التصميم التقليدي لإعادة التشكيل الجزئي. لكن استخدام هاته الأساليب أوجد بعض العيوب المتعلقة بتشغيلها. أولا، تؤدي المناهج المبرمجة التي يشغلها المعالج إلى زيادة في وقت التنفيذ الإجمالي عند التعامل مع البيتستريم و الذي يمكن أن يعتبر مفرطا في التطبيقات الحرجة ذات الوقت الحقيقي. ثانيا ، تعتبر المناهج المشغلة داخل الجهاز هي أسرع من حيث الأداء لكنها تحتاج إلى الكثير من الموارد المنطقية لتنفيذ التغييرات الضرورية لإنتاج البيتستريم القابل للتوطين.

لجعل عملية إعادة التوطين أقل تكلفة من حيث الوقت المستغرق من قبل الموطن لتغيير البيستريم وكذلك من حيث استخدام الموارد المنطقية لتنصيب الموطن داخل الشريحة، قمنا بتطوير طريقة جديدة لإعادة التوطين أسميناها أوربيت (OORBIT) . يمكن لهذا المنهاج الجديد تسريع عملية التوطين الى حد أعظم وذلك بدون أي تأثير على موارد النظام.

سيتم وصف منهاجنا بالتفصيل من خلال التركيز على تفاعله مع مراحل التصميم الجديد لإعادة التشكيل التي تستند إلى مبدأ التقسيم. سوف نعطي في الأخير تحليل مقارن للأداء مع المناهج الأخرى للتوطين المقترحة سابقا، حيث سوف يتم تسليط الضوء على التسارع الكبير الذي ينجزه منهجنا.

كلمات البحث : حساب مشكل ، النظام في رقاقة، إعادة تشكيل جزئي، إعادة توطين البيستريم , FPGA .

ABSTRACT

Xilinx Virtex FPGAs offer the possibility of Partial Reconfiguration (PR). Arbitrary tasks can be allocated and de-allocated onto FPGA without system interruption. However, mapping a task to any available PR region requires a unique partial bitstream for each partition, hence reducing memory storage requirements.

In order to find solutions to these limitations and increase the capacity of the partial dynamic reconfiguration process, several approaches of bitstream relocation have been proposed and are presented in the majority as extensions of the traditional design flow for dynamic reconfiguration. However the use of these approaches has some drawbacks related to their implementations. Firstly, software approaches executed by embedded processor introduce additional runtime when manipulating bitstream, which can be considered excessive in critical real-time applications. Secondly, relocation approaches implemented in hardware are faster in terms of performance but need many costly logic resources for their implementation in order to produce a relocatable bitstream.

To make the relocation process less costly in terms of time taken by the relocator to manipulate bitstream and in terms of resources utilization to implement the logic relocator, we have developed a new methodology of relocation called OORBIT (for offline / online relocation of bitstream). It is based on a combination of two treatments of the bitstream in offline and online. This new approach has accelerated impressively the process of relocation without any impact on system resources.

Our methodology is described in detail by focusing on its interaction with the new design flow based on the principle of partition. A comparative performance analysis is given to highlight the very significant acceleration of the process of relocation performed by our approach.

Keywords: Reconfigurable Computing, System on Chip, Dynamic Partial Reconfiguration, Bitstream Relocation, FPGA.

RESUME

Les circuits FPGA Virtex de Xilinx offrent la fonctionnalité de la reconfiguration partielle (PR). Des tâches arbitraires peuvent être allouées et désallouées dans le FPGA sans interrompre le système. Cependant le placement d'une tâche dans l'espace du FPGA nécessite un bitstream partiel unique spécifique pour chaque partition ce qui mène à un besoin plus grand en capacité mémoire pour le stockage des différents bitstreams partiaux.

En vue de trouver des solutions à ces limitations et d'augmenter les capacités du processus de la reconfiguration dynamique partielle, plusieurs approches de relocation ont été proposées et se présentent dans leur majorité comme des extensions du flot traditionnel de conception de la reconfiguration dynamique. Toutefois, l'utilisation de ces approches présente quelques inconvénients liés à leurs implémentations. En premier lieu, les approches logicielles exécutées par le processeur embarqué introduisent un temps d'exécution supplémentaire, lors de manipulation de bitstream, qui peut être considéré excessif dans des applications critiques en temps réel. En second lieu, les approches de relocation implémentées en matériel sont plus rapides en terme d'exécution mais ont besoin de nombreuses ressources logiques coûteuses pour leurs implémentation en vue d'effectuer toutes les modifications nécessaires pour produire un bitstream relogeable.

Afin de rendre le processus de relocation moins couteux en matière de temps pris par le relocateur pour manipuler le bitstream et en terme de taux d'utilisation des ressources logiques pour implémenter le relocateur, nous avons développé une nouvelle méthodologie de relocation que nous avons appelée OORBIT (pour offline/online relocation of bitstream). Elle est basée sur une combinaison de deux méthodes de manipulation du bitstream, en hors ligne et en ligne. Cette nouvelle approche a permet d'accélérer considérablement le processus de relocation et cela sans aucune incidence sur les ressources du système.

Notre méthodologie sera décrite en détail en focalisant sur son interaction avec le nouveau flot de conception basé sur le principe des partitions. Une analyse comparative des performances, par rapport aux autres approches proposées, sera donnée pour mettre en évidence l'accélération très significative du processus de relocation réalisée par notre approche.

Mots clés : Calcul Reconfigurable, Système sur Puce, Reconfiguration Dynamique Partielle, Relocation de Bitstream, FPGA.

REMERCIEMENTS

Cette thèse s'est effectuée en cotutelle entre le Laboratoire d'automatique et de traitement de signal et d'image (LATSI) de l'université SAAD DAHLAB de Blida (Algérie) et l'équipe architecture du laboratoire d'électronique, d'informatique et d'image (LE2I) de l'université de Bourgogne à Dijon en France.

Je remercie en tout premier lieu mes directeurs de thèse, Monsieur El-Bay BOURENNANE et Monsieur Abderrezak GUESSOUM, pour m'avoir confié ce sujet de thèse, pour m'avoir accueilli au sein de leurs équipes de recherche, pour la confiance qu'ils m'ont témoignée tout au long de ces années de travail et pour leur soutien tant scientifique que moral.

Je voudrais exprimer ma gratitude à Monsieur Hassan SALHI, Professeur à l'Université de Blida, qui m'a fait l'honneur de présider ce jury.

Je suis reconnaissant à Monsieur Abdelaziz FERDJOUNI, Maître de conférences à l'Université de Blida, Monsieur Mohamed OULD ZMIRLI, Maître de conférences à l'Université de Médéa et Monsieur Lahcène MITICHE, Maître de conférences à l'Université de Djelfa, qui m'ont beaucoup honoré par leur participation en tant que membres de jury de cette thèse.

Mes remerciements les plus chaleureux vont à mes collègues au laboratoire LE2I à Dijon. Tout particulièrement à Gilberto Ochoa Ruiz (Beto) et kamel Messaouadi pour leurs contributions et encore pour l'ambiance qu'on a pu créer ensemble.

Je remercie également mes collègues, du Centre de Recherche Nucléaire de Birine, pour m'avoir aidé de près ou de loin à la réalisation de ce travail.

Je salué fortement l`Agence Universitaire de la Francophonie (AUF), pour m'avoir octroyé une bourse de formation à la recherche de deux ans. Je le remercie vivement pour ses efforts déployés dans la promotion et le développement des échanges scientifiques qui contribuent au soutien et au renforcement de l'excellence universitaire.

Enfin, je remercie tous les membres de ma famille pour leur soutien et en particulier ma fille Amina qui a bien voulu laisser sa touche dans ce manuscrit.

Maamar TOUIZA,
Septembre 2013

A mes chers parents

A mes frères et sœurs

A ma femme

A mes enfants : Amina, Bouchra, Khedidja, Noura, Abderahmane et Zineb

A toute ma famille

TABLE DES MATIERES

LISTE DES ILLUSTRATIONS, GRAPHIQUES ET TABLEAUX

LISTE DES FIGURES

LISTE DES TABLEAUX

INTRODUCTION

INTRODUCTION

Avec la convergence de l'électronique, de l'informatique et des télécommunications, les technologies de l'information occupent une place préférentielle dans notre quotidien. En effet, l'évolution des systèmes embarqués et des systèmes sur puces permettent de développer des fonctions de plus en plus complexes sans négliger la vitesse de traitement et la consommation d'énergie. Les circuits intégrés spécifiques appelés ASIC (Application Specific Integrated Circuit) représentent une solution optimale sur le plan du taux d'intégration au détriment d'un coût de développement et de fabrication important, et d'un degré de flexibilité loin d'être idéal. Par ailleurs, avec la réduction de l'échelle de gravure des circuits, la consommation statique augmente considérablement ce qui devient un problème majeur pour les applications nomades. Egalement, toute modification ou mise à jour d'un ASIC se soldera par la fabrication d'un nouveau circuit. Or, les systèmes électroniques embarqués ont besoin de circuits capables de répondre aux exigences de coût, de performance, de flexibilité et de consommation. Les circuits logiques programmables de type FPGA (Field Programmable Gate Array) constituent une solution qui dispose d'une meilleure flexibilité, de performances acceptables et d'un temps de mise en œuvre réduit par rapport aux ASICs [1]. Toutefois, les solutions FPGA n'ont pas encore la flexibilité offerte par leur homologues à base de processeur où DSP. Car une fois l'application a été mise en œuvre, un petit changement plus tard conduira à la reconfiguration de l'ensemble du système. En plus, Les applications FPGA de type SoC implémentent souvent des fonctionnalités complexes dont plusieurs modules peuvent rester inactifs durant des périodes de temps étendues. Ces modules sont continuellement placés dans le composant consommant ainsi des ressources chères en matériel et compliquant encore l'optimisation du système lors de la phase d'implémentation due entre autres à l'accroissement de l'infrastructure de communication.

La reconfiguration dynamique a été introduite ces dernières années comme un moyen permettant de virtualiser les tâches matérielles dans les systèmes à base de FPGA [2]. Cependant, ce n'était qu'avec l'introduction de la technologie de reconfiguration dynamique partielle (RDP) par Xilinx que ces systèmes sont devenus une réalité. Dans ces systèmes, les modules reconfigurables dynamiquement peuvent être implémentés comme

des tâches matérielles indépendantes qui peuvent être chargées dans le composant en cas de besoin en remplaçant d'autres fonctionnalités. Originellement, cette modification impliquait la reconfiguration du composant entier. Actuellement, la reconfiguration dynamique partielle offre la possibilité de modifier sur demande la fonctionnalité d'une partie du circuit pendant que le système fonctionne. Ainsi, le FPGA continue son opération alors qu'une portion du système subit une mise à jour [3].

La reconfiguration dynamique partielle a plusieurs avantages. En plus de l'augmentation de la flexibilité du système déjà mentionnée, elle permet d'utiliser plus de matériel que celui présent physiquement dans le FPGA en virtualisant les tâches matérielles et les plaçant par la suite à la demande dans le FPGA. Cette possibilité peut être exploitée pour réduire la taille du FPGA et sa consommation d'énergie globale. Cela permet aussi d'exécuter un algorithme et optimiser son implémentation en fonction de l'ensemble de ses paramètres et ses données. De nombreuses applications industrielles ont déjà trouvé le jour en exploitant l'approche RDP dans plusieurs domaines tels que le radio logiciel [4], la biométrie [5], l'industrie automobile [6] et bien d'autres. Ces applications sont généralement implémentées sur des architectures reconfigurables de type SoC, ce qui leur permet d'exploiter les avantages de la flexibilité du matériel reconfigurable et la programmabilité du processeur qui contrôle le SoC. Les FPGA actuels peuvent contenir un ou plusieurs processeurs directement incorporés dans le silicium où encore des processeurs logiciels qui peuvent être synthétisés et ajoutés dans la plateforme.

Ces dernières années, un grand effort de recherche est en train d'être déployé pour étendre les capacités des SoC reconfigurables et formaliser l'utilisation de la reconfiguration partielle [7,8]. L'idée de base est que des régions spécifiques dans le FPGA peuvent être utilisées pour placer des tâches matérielles à la demande permettant de promouvoir d'une façon efficace la notion de virtualisation des tâches matérielles. Ces dernières sont stockées sous forme de bitstreams précompilés dans une mémoire externe en général où ils peuvent être chargés par le processeur à la demande dans des régions où on veut remplacer leurs missions. D'autres applications peuvent avoir besoin de plusieurs instances du même module telles que les systèmes qui supportent la migration chaude [9] et les systèmes à tolérance de panne [10]. Dans le premier cas, plusieurs copies de la même fonction peuvent être potentiellement nécessaires à n'importe quel moment donné.

Dans le deuxième cas, l'emplacement du même module nécessite d'être changé en fonction des conditions physiques du système.

Similairement aux tâches logicielles, ces tâches matérielles ont besoin aussi d'être gérées, ordonnancées et placées dans la logique reconfigurable. Elles ne sont plus considérées comme périphériques esclaves attachées à un bus et contrôlées par des pilotes logicielles. Elles sont maintenant plus actives et peuvent demander, tout comme les tâches logicielles, des primitives de synchronisation, l'accès au bus et à la mémoire et peuvent être dynamiquement crées et terminées. Tous ces comportements des tâches matérielles dans un système reconfigurable doivent être gérés par un système d'exploitation (OS). En conséquence, un OS destiné aux systèmes reconfigurables doit être en mesure de fournir des services supplémentaires pour gérer l'ensemble des traitements dans la plateforme reconfigurable.

Un service plus avancé de l'OS pourrait être aussi la relocation qui permet à une tâche matérielle déjà placée et configurée d'être relogée dans une autre région de la logique reconfigurable en utilisant d'autres ressources. Les services précédents s'appuient tous sur la propriété de relocation de tâches, qui permet de déplacer une tâche matérielle au sein de l'architecture reconfigurable. Cette propriété est nécessaire pour l'implémentation de la migration de tâche de matériel vers matériel. Après relocation, Cette tâche va continuer à remplir la même fonction de l'endroit où il a été préempté.

Toutefois, la relocation doit être en mesure de placer un bitstream dans n'importe quelle région reconfigurable disponible ce qui n'est pas possible avec le flot standard de la RDP. Les flots de conception de la reconfiguration partielle fournis par Xilinx [11,12] limitent l'emplacement des modules partiellement reconfigurables (PRM) dans des régions reconfigurables prédéfinies dans lesquelles les PRM seront ensuite placées. La communauté des chercheurs et les créateurs de systèmes sur puce ont trouvé cette méthode trop restrictive pour la mise en œuvre des systèmes complexes avec plusieurs fonctionnalités. Les limitations dans le flot de conception RDP sont multiples mais le problème majeur provient du fait que le placement des PRM est limité à des régions spécifiques ; les applications doivent être donc soigneusement planifiées en avance ce qui aboutit à des systèmes avec une flexibilité limitée.

En vue de trouver des solutions à ces limitations et d'augmenter les capacités du processus de la reconfiguration dynamique partielle, plusieurs approches de relocations ont été proposées et se présentent dans leur majorité comme des extensions du flot traditionnel de conception de la reconfiguration dynamique. Le processus de relocation cherche à étendre la plaçabilité d'un bitstream en effectuant des modifications sur son contenu. Ces modifications affectent principalement les informations liées aux adresses de configuration et consiste à déplacer les coordonnées de la partition prévue vers d'autres régions dans l'espace reconfigurable du FPGA. Les approches de relocation proposées dans la littérature sont basées sur une manipulation en ligne du bitstream. Quelques modules de relocation sont implémentés en matériel et des autres sont implémentés en logiciel. Les deux variantes sont généralement efficaces mais présentent les deux des inconvénients. En premier lieu, les approches logicielles exécutées par le processeur embarqué introduisent un temps d'exécution, lors de manipulation de bitstream, qui peut être considéré excessif dans des applications critiques ou en temps réel. En second lieu, les approches de relocation implémentées en matériel sont plus rapides en terme d'exécution mais ont besoin de nombreuses ressources logiques coûteuses pour leurs implémentation en vue d'effectuer toutes les modifications nécessaires pour produire un bitstream relogeable.

Afin de rendre le processus de relocation moins coûteux en termes de temps de manipulation de bitstream et de taux d'utilisation des ressources, nous proposons une nouvelle méthodologie que nous appelons OORBIT (pour Offline/Online Relocation of Bitstream). Pour la phase hors ligne, nous avons développé un outil qui permet en première phase d'analyser le fichier bitstream généré par le flot de conception traditionnel de la reconfiguration partielle. Toutes les adresses de configurations dans le fichier seront identifiées et par la suite toutes les régions relogeables possibles seront spécifiées. En second lieu, nous procédons au calcul de nouvelles adresses de configurations pour chaque allocation possible du bitstream. Toutes ces nouvelles informations liées à la relocation seront ensuite ajoutées au bitstream original sous forme d'un complément de données. Le bitstream modifié supportant la relocation sera utilisé durant l'exécution de l'application par notre module développé pour la relocation en ligne qui sera exécutée par le processeur. Il consiste principalement à modifier les anciennes adresses de configuration du bitstream original par ceux pré-calculés en hors ligne correspondants à la nouvelle allocation désirée.

Cette solution permet d'accélérer le processus de relocation d'une façon spectaculaire et cela sans introduire aucune incidence sur les ressources logiques du système.

Le présent manuscrit est organisé comme suit : dans le premier chapitre, nous donnons une description plus au moins détaillée sur les architectures configurables à base de FPGA et les outils nécessaires associés pour la mise en œuvre de systèmes embarqués complexes. Le concept de la reconfiguration dynamique sera décrit en suite en détail dans le deuxième chapitre. Dans sa première partie, nous décrirons les différents modes de reconfiguration, les avantages de la reconfiguration partielle et la terminologie pour la reconfiguration des FPGA de Xilinx utilisée dans ce manuscrit. Puis, une présentation dédiée aux techniques de reconfiguration des FPGA et à l'évolution des flots de conception existants associés à la reconfiguration partielle. Nous montrerons, ensuite, les limitations de ces flots et les problèmes rencontrés pour bien gérer le plaçabilité des tâches matérielles dans le FPGA.

Dans le troisième chapitre, nous allons passer en revue les différents services qui peuvent être offerts à une architecture reconfigurable dynamiquement pour améliorer sa flexibilité. Par la suite, nous discuterons le service de relocation en se focalisant tout d'abord sur les aspects technologiques et architecturaux des FPGA qui limitent l'application de la relocation dans le flot de conception de la reconfiguration partielle. Ensuite nous présenterons les principales approches de relocation proposées dans la littérature et leurs limites en termes de ressources déployées par le relocateur, du temps nécessaire pour effectuer la relocation, de flot de conception DPR déployé et d'autres contraintes. Ensuite, nous introduirons brièvement notre contribution dans le domaine en présentant notre nouvelle méthodologie de relocation et les améliorations qu'elle apporte pour lever ces contraintes.

La description détaillée de notre méthodologie de relocation va faire l'objet du quatrième chapitre. Nous décrivons tout d'abord l'outil hors ligne d'OORBIT, en se focalisant principalement sur les deux phases d'analyse et de relocation du bitstream relogeable. Nous montrons ensuite la manière avec laquelle notre outil de relocation hors-ligne est utilisé dans le contexte de nouveau flot de conception DPR basé sur les partitions. Nous détaillons comment OORBIT interagit avec les différents outils utilisés dans le nouveau flot et comment il exploite les fichiers intermédiaires pour faire réussir le processus de relocation. Par la suite, nous expliquons la façon avec laquelle ce fichier

bitstream relogeable sera utilisé par le module en ligne d’OORBIT pour effectuer la relocation. En fin de ce chapitre, nous présentons un ensemble d'expériences exécutées en vue d’évaluer et de valider notre approche de relocation. Nous montrons également l'importance de certains paramètres, en particulier le temps de relocation et la quantité des ressources nécessaires pour implémenter le composant de relocation dans le système. Ces informations seront utilisées pour comparer notre approche avec les autres approches de relocation étudiées.

La dernière partie tiendra lieu de conclusion générale. Nous discuterons et dresserons le bilan du travail de thèse en présentant les apports proposés et en dégageant les perspectives ouvertes par ces travaux.

Chapitre 1

ARCHITECTURE RECONFIGURABLE

A BASE DE FPGA

CHAPITRE 1

ARCHITECTURE RECONFIGURABLE A BASE DE FPGA

1.1 Introduction

Le monde de la micro-électronique a énormément évolué dans les 50 dernières années. Depuis l'avènement des premiers circuits intégrés dans les années 60, le nombre d'applications découlant de l'utilisation de tels composants n'a cessé de croitre. Pour réaliser des circuits complexes, le choix se portait alors soit sur les microprocesseurs à prédominance logicielle ou encore sur les circuits intégrés dédiés à prédominance matérielle. En plus de disposer d'une myriade de composants élémentaires de type portes logiques de base. Les circuits intégrés dédiés qu'on nomme aussi ASIC (pour Application Specific Integrated Circuit) sont des composants, qui ont certes beaucoup évolué depuis leur début en terme de capacité, mais qui n'en demeurent pas moins longs et coûteux à concevoir. Pour répondre aux exigences du marché, avec une diversité d'applications, l'électronique a dû progressivement s'adapter pour devenir portable, puis ultra miniaturisée, c'est la naissance des systèmes (numériques) embarqués. Ces systèmes ont vu leur importance progresser au rythme de l'importance prise par les microprocesseurs. Ces microprocesseurs sont en mesure d'accomplir des instructions plus complexes et atteignent des vitesses de traitement toujours plus rapides. Ils conservent cependant un champ d'application restreint c'est-à-dire strictement logiciel. A l'aide de ces deux ressources principales, le microprocesseur et l'ASIC, nous avons pu par le passé, construire des systèmes efficaces, basés soit sur l'un ou sur l'autre sans toutefois atteindre la rapidité de développement que requiert le marché.

Au milieu des années 80, est né le tout premier FPGA (Field Programmable Gate Array). Ce composant de la logique programmable regroupe un nombre fixe de portes logiques de base qui peuvent être utilisées afin de former un circuit spécifique. Le FPGA se différencie des autres membres de la logique programmable puisqu'il permet de construire des circuits plus complexes. En effet, ayant en général plus de ressources matérielles et une architecture plus étoffée, le FPGA permet de concevoir des circuits adaptés à des applications aussi différentes que le contrôle et le traitement de données.

Les FPGA ont comme avantage d'être plus rapides que les microprocesseurs dans certaines applications et définitivement plus flexibles que les circuits intégrés dédiés. Ils peuvent donc rallier le meilleur des deux mondes à moindre coût de développement qu'un circuit intégré dédié.

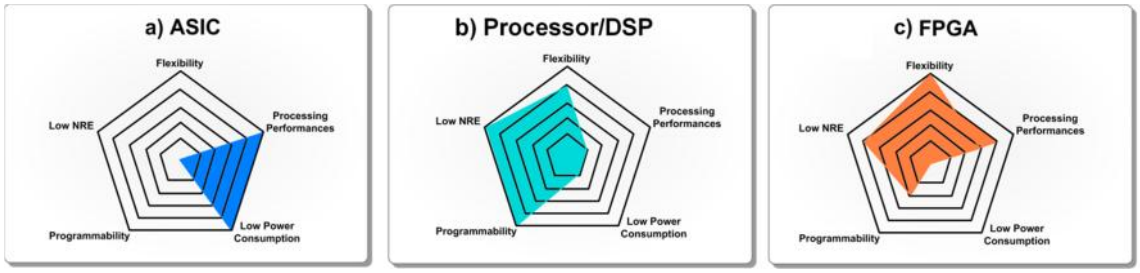


Figure 1.1: Comparaison entre les différentes technologies utilisées dans les SoC

Plusieurs avantages militent en faveur des FPGA. Le premier est sans doute sa reprogrammabilité. Plusieurs applications utilisent les FPGA soit à titre d'élément de base (traitement) ou à titre de soutien (contrôle). On voit aussi très souvent les FPGA comme coprocesseur à certains systèmes exécutant une tâche spécifique qui permet de libérer le processeur principal. La popularité des FPGA fait en sorte qu'on exige plus de rendement, et plus de flexibilité de leur part. On veut à la fois des outils plus performants pour le placement routage et pouvoir les reprogrammer le plus rapidement possible afin de construire des systèmes de traitement temps réel moins coûteux. La demande pour une flexibilité accrue a poussé les chercheurs et les industriels vers un type particulier de reprogrammation qui s'effectue pendant que le système s'exécute. C'est ce qu'on appelle la reconfiguration dynamique, traduction de RTR pour Run Time Reconfiguration. La technologie FPGA actuellement disponible permet de construire des systèmes qui sont configurables et/ou reconfigurables. L'idée commune à tous les circuits configurables est de proposer, non pas une fonction figée dont le champ d'application est le plus large possible, mais une structure adaptable. La reconfiguration consiste à modifier en cours de fonctionnement l'architecture matérielle, cela permet de concevoir des circuits intelligents qui peuvent s'auto adapter à leur environnement.

Le but de ce chapitre est de fournir une description plus au moins détaillée sur les architectures configurables à base de FPGA et les outils nécessaires associés pour la mise en œuvre de systèmes embarqués complexes. Le concept de la reconfiguration dynamique sera décrit en suite en détail dans le prochain chapitre.

1.2 Description des FPGA

Les FPGA (Field Programmable Gate Array) sont inventés par la société Xilinx en 1985. Xilinx fut le précurseur du domaine en lançant le premier circuit FPGA commercial, le XC2000. Xilinx ne sera suivi qu'un peu plus tard, et jamais lâché, par son plus sérieux concurrent Altera qui lança en 1992 la famille de FPGA FLEX 8000 dont la capacité maximum atteignait 15 000 portes logiques.

Les FPGA se situent entre les réseaux logiques programmables et les circuits logiques près diffusés. Les réseaux logiques programmables sont des composants qui ne nécessitent aucune étape technologique supplémentaire pour être personnalisés, ce sont des circuits standards, programmables par l'utilisateur grâce aux différents outils de développement et qui incluent un grand nombre de solutions basées sur les variantes de l'architecture des portes ET et OU. Les prés diffusés sont des circuits intégrés basés sur l'utilisation des réseaux de cellules dont les blocs ont été préalablement diffusés, il faut créer les connexions entre ces blocs. Les FPGA combinent donc à la fois la souplesse de la programmation des réseaux logiques programmables et les performances des circuits prés diffuses. En d'autres termes le FPGA permet d'avoir une architecture conçue sur mesure à haute densité dans un circuit électronique, avec la possibilité de modifier cette architecture quand des nouvelles applications apparaissent. Les langages HDL comme le VHDL ou le Verilog sont utilisés pour décrire les fonctionnalités qui seront implémentées sur le composant, puis la description matérielle est traduite dans un fichier de configuration pour le FPGA cible. La description matérielle peut être aussi utilisée pour la description d'un composant ASIC.

En 2000 et 2001, les deux concurrents Xilinx et Altera ont franchi une nouvelle étape au niveau de la densité d'intégration en proposant respectivement leurs circuits Virtex et Apex-II dont les capacités maximums avoisinaient les 4 millions de portes logiques équivalentes avec de plus l'introduction de larges bancs de mémoires embarquées. Aujourd'hui, les fréquences de fonctionnement de ces circuits sont de l'ordre de quelques centaines de Méga Hertz (ces dernières sont en réalité très dépendantes de l'application). Bien que ces valeurs soient relativement réduites par rapport aux ASICs, elles sont suffisantes pour une très large majorité d'applications actuelles. À partir des années 2000, les capacités des FPGA ont permis d'offrir aux concepteurs une solution supplémentaire de réalisation pour une majorité d'applications. De plus, les outils de mise

en œuvre des FPGA ont évolué et bien qu'encore pénalisants lors de la conception, ils permettent la réalisation rapide d'applications complexes.

La flexibilité du FPGA permet de changer sa description matérielle en fonction de l'application, c'est le facteur principal qui a déterminé la popularité des composants FPGA. De plus, les FPGA peuvent être couplés avec un processeur à usage général. Ainsi la section la plus exigeante du logiciel peut être traduite sur la partie matérielle qui permet d'accélérer l'exécution du programme. Ce qui donne des accélérations notables dans l'exécution, en particulier lorsque le programme exécuté en série sur un processeur peut être traduire en matériel pour exploiter le parallélisme de l'algorithme. Par conséquent, les FPGA permettent d'obtenir des performances de calcul importantes. De plus, le nombre de portes logiques sur les FPGA les plus récents permet l'implémentation complète d'un système sur puce (SoC - System On Chip).

1.2.1 Architecture des circuits FPGA

Cette partie présente les caractéristiques des FPGA, en particulier pour clarifier la façon dont ils peuvent mettre en œuvre la personnalisation du matériel via leur reconfiguration. Bien qu'il existe actuellement plusieurs fabricants de circuits FPGA, dont Xilinx [13] et Altera [14] qui sont les plus connus, nous décrirons l'architecture des FPGA à partir des circuits Xilinx que nous avons employés dans la suite de notre étude. L'architecture, retenue par Xilinx (figure 1.2), se présente sous la forme de deux couches distinctes, la première est la couche appelée circuit configurable et la deuxième est une couche de réseau mémoire SRAM.

La première couche (circuit configurable) est constituée d'une matrice de blocs logiques configurables (CLB - Configurable Logic Bloc). Les CLB permettent de réaliser des fonctions séquentielles et combinatoires. Autour de ces blocs logiques configurables, nous trouvons les blocs d'entrées-sorties (IOB - Input Output Bloc). Ils permettent de gérer les entrées-sorties pour réaliser l'interface avec les modules extérieurs.

La seconde couche est un réseau de mémoire SRAM qui permet la programmation du circuit FPGA. La programmation est réalisée en appliquant les potentiels adéquats sur la grille de certains transistors à effet de champ pour interconnecter les éléments des CLB et des IOB afin de réaliser les fonctions souhaitées et d'assurer la propagation des signaux.

Ces potentiels sont mémorisés dans le réseau de mémoire SRAM. Un dispositif interne au FPGA permet à chaque mise sous tension de charger les SRAM internes à partir d'une mémoire externe où est stockée la configuration du FPGA.

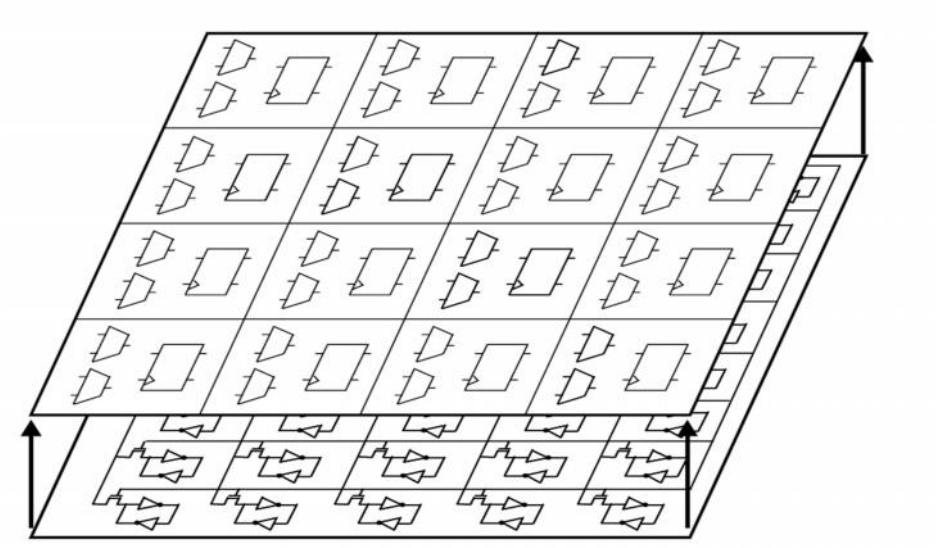


Figure 1.2 : Architecture FPGA retenue par Xilinx formée de deux couches, la première est appelée circuit configurable et la deuxième est le réseau mémoire SRAM

1.2.2 Présentation interne d'un FPGA

Un FPGA est un circuit intégré constitué d'un réseau central de blocs logiques qui peuvent être connectés grâce à une matrice d'interconnexion configurable. Dans la périphérie du réseau logique se trouve un anneau de blocs d'E/S qui peuvent être configurées pour supporter différentes normes d'interfaçage avec l'extérieur. Cette architecture flexible peut être utilisée pour mettre en œuvre un large éventail de fonctions logiques numériques combinatoires et séquentielles. Cela est dû au fait que leur tissu sous-jacent se compose principalement d'un grand nombre de blocs logiques programmables relativement simples noyés dans une mer d'interconnexion programmable, comme représenté sur la figure 1.3 a.

Les blocs logiques configurables CLB sont les principaux éléments d'un FPGA. Après plusieurs décennies de recherche, il a été établi que le meilleur bloc fonctionnel standard d'un FPGA est un bloc logique construit autour d'une table de correspondance LUT (pour look up table). Une LUT à N entrées est essentiellement une mémoire qui, lorsqu'elle est programmée de façon appropriée, peut calculer une fonction de N entrées.

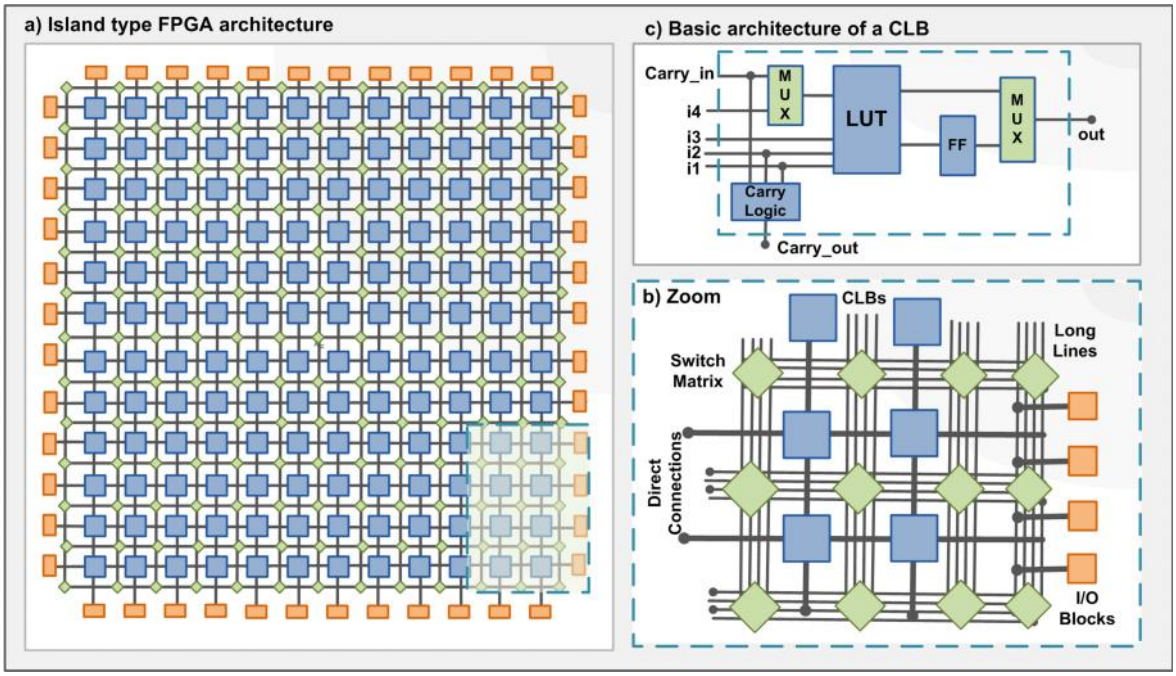


Figure 1.3: Architecture matérielle générique d'un FPGA

Cette flexibilité, avec des exigences de routage relativement simples s'avère très puissante pour l'implémentation de n'importe quelle fonction logique. Un bloc logique d'un FPGA typique est constitué d'un ou plusieurs LUT à N entrées, des bascules D(FF), et une certaine logique de traitement rapide de retenue, comme présenté dans la figure 1.3c.

A titre d'exemple, Xilinx a introduit des LUT à 6 entrées dans ses FPGA de familles Virtex 5, Virtex 6, Virtex 7 et Spartan 6 [15]. Autour d'une LUT, il y a une logique d'interconnexion qui permet les liaisons à destination et vers la LUT. Elle est mise en œuvre à l'aide de portes logiques et de multiplexeurs. Pendant le processus de configuration d'un FPGA, les mémoires des LUT sont écrites pour y implémenter une fonction, et la logique qui l'entoure est configurée pour router correctement les signaux afin de construire un système complexe. Les bascules FF dans un CLB peuvent être utilisées pour réaliser des pipelines, implémenter des registres et réaliser des circuits de rétention des états pour les machines d'états finis ou dans tout autre cas où la présence de l'horloge est nécessaire. La logique de traitement rapide de retenue dans un bloc logique est utilisée pour accélérer les calculs à base de retenue telle que l'addition, la parité, les larges opérations AND et pour d'autres fonctions.

Parallèlement aux efforts consentis pour le développement de l'architecture des blocs logiques CLB dans un FPGA, il y a eu également autant d'investigations dans les structures d'interconnexion. Xilinx a adopté une structure sous forme d'une ile où on trouve la logique entourée par des canaux de routage permettant une communication rapide et efficace le long des rangées et des colonnes des blocs logiques (figure 1.3b). Ces ressources d'interconnexion au sein d'un FPGA permettent une connexion arbitraire des CLB d'abord entre eux puis avec les blocs d'entrée-sortie (IOB). Les connexions internes dans les circuits FPGA sont composées de segments métallisés. Parallèlement à ces lignes, nous trouvons des matrices programmables réparties horizontalement et verticalement entre les divers CLB sur la totalité du circuit. Elles permettent les connexions entre les diverses lignes à travers des transistors MOS dont l'état est contrôlé par des cellules de mémoire SRAM. Le rôle de ces interconnexions est de relier avec un maximum d'efficacité les blocs logiques et les entrées-sorties afin que le taux d'utilisation dans un circuit donné soit le plus élevé possible. Pour parvenir à cet objectif, Xilinx propose trois sortes d'interconnexions selon la longueur et la destination des liaisons.

- Les interconnexions directes : Ces interconnexions permettent l'établissement des liaisons entre les CLB et les IOB. Il est possible aussi de connecter directement certaines entrées d'un CLB aux sorties d'un autre.
- Les longues lignes : Ce sont de longs segments métallisés parcourant toute la longueur et la largeur du FPGA, elles permettent éventuellement de transmettre avec un minimum de retard les signaux entre les différents éléments dans le but d'assurer un synchronisme aussi parfait que possible. De plus, ces longues lignes permettent d'éviter la multiplicité des points d'interconnexion.
- Les matrices d'interconnexion : Ce sont des aiguilleurs situés à chaque intersection, leur rôle consiste à raccorder les longues lignes entre elles selon diverses configurations. Ces interconnexions sont utilisées pour relier un CLB à n'importe quel autre CLB sur le FPGA pour assurer la communication des signaux. Pour éviter l'affaiblissement des signaux traversant les longues lignes, des buffers sont implantés dans chaque matrice d'interconnexion.

Les blocs d'entrée-sortie ou IOB (Input Output Bloc) permettent l'interface entre les broches du composant FPGA et la logique interne développée à l'intérieur du composant. Ils sont présents sur toute la périphérie du circuit FPGA. Chaque bloc IOB

contrôle une broche du composant et il peut être défini en entrée, en sortie, en signal bidirectionnel ou être inutilisé (état haute impédance). Les IOB peuvent s'adapter à un grand nombre de standards de communication: LVTTL, LVCMOS, PCI, LVDS, LVPECL, etc. Ils sont regroupés en banques, et chaque banque a sa propre tension d'alimentation. Ceci permet de combiner des standards incompatibles entre eux sur le même circuit FPGA en utilisant des banques différentes.

Les trois blocs présentés jusqu'ici sont interconnectés ensemble dans le dispositif pour créer une infrastructure de communication composée d'un réseau de communication et d'IOBs autour des CLBs. Des cellules de mémoire liées à chaque bloc détiennent les caractéristiques principales, de telle sorte que les interconnexions entre l'infrastructure de communication, les normes de tension des entrées-sorties d'un IOB, et les équations soient commandées par des valeurs particulières stockées dans une mémoire. Toutes ces configurations sont stockées dans des SRAM qui sont volatiles : lorsque le composant est mis hors tension, toute sa configuration est perdue et il doit être redémarré avec une nouvelle configuration. Habituellement, une machine externe se charge de télécharger la configuration sur le FPGA via une de ses interfaces de configuration, et envoie une commande de démarrage pour signaler que la configuration a eu lieu. Certaines cartes ont une mémoire ROM intégrée. Elle permet de stocker la configuration, de sorte qu'elle puisse ensuite être téléchargée sur le FPGA. Dans ce cas, les données de configuration sont copiées sur la mémoire SRAM de configuration du FPGA au démarrage de celui-ci.

1.2.3 Plateformes hétérogènes :

En vue de fournir plus de performance et de flexibilité à leurs circuits, les constructeurs des FPGA ont graduellement incorporé plus de fonctionnalités à leurs composants programmables [16]. Ces fonctionnalités sont embarquées avec les ressources logiques de circuit produisant ainsi une structure hétérogène. Cette approche est devenue une nécessité due à l'adoption des FPGA dans l'implémentation des plateformes complexes telles que des systèmes sur puce où plusieurs processeurs, des standards de communication et des contrôleurs de mémoires sont nécessaires. La composition détaillée d'un circuit FPGA (figure 1.4), illustre la disposition de ces blocs additionnels dans le composant. Le fonctionnement et le placement de ces blocs diffèrent d'une référence de FPGA à une autre.

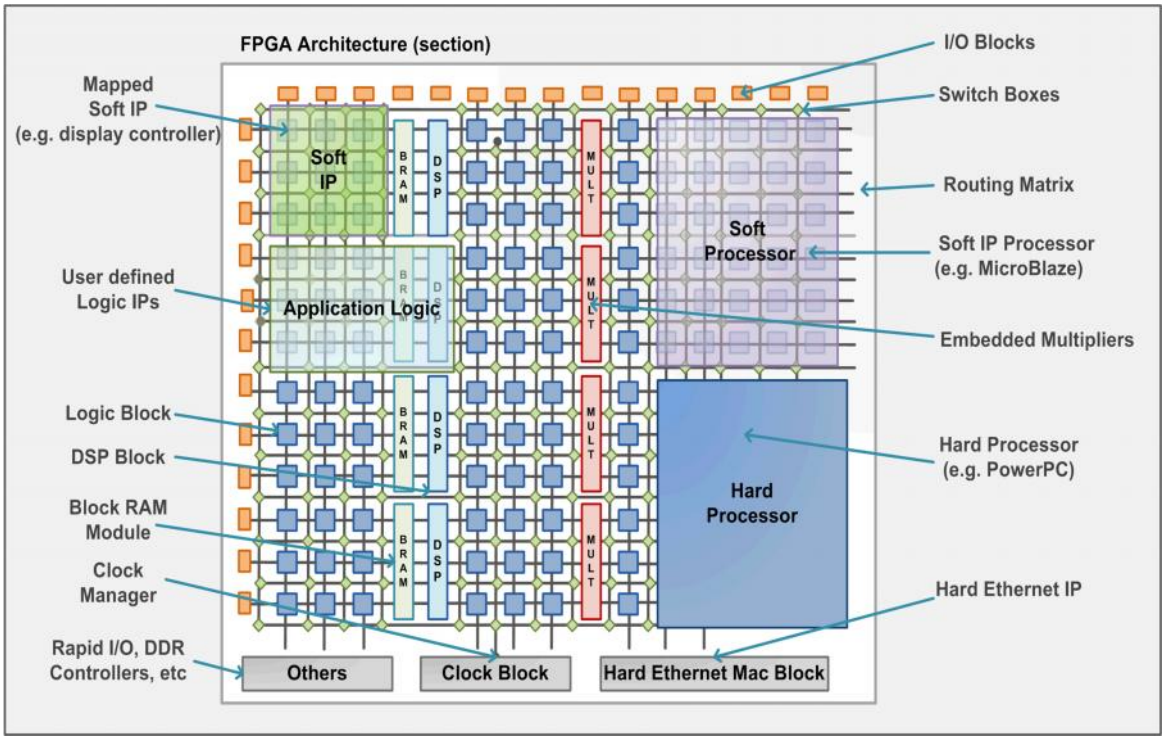


Figure 1.4: Plateforme FPGA hétérogène

Les applications DSP (comme le traitement de l'image et de vidéo) nécessitent souvent des fonctions complexes telles que des multiplicateurs accumulateurs (MAC). Ces fonctions sont difficiles à mettre en œuvre de manière optimale en utilisant les ressources de logique FPGA génériques, par conséquent, les fournisseurs FPGA ont introduit des blocs DSP complexes sous la forme de fonctions câblées. Ces blocs sont conçus pour être aussi efficaces que possible en termes de consommation d'énergie et de performance. Chaque bloc DSP permet de réaliser un multiplicateur, un accumulateur, un additionneur, et des opérations logiques (AND, OR, NOT, et NAND) sur un bit. Il est possible de combiner les blocs DSP pour effectuer des opérations plus importantes, telles que l'addition avec virgule flottante simple, la soustraction, la multiplication, la division, et la racine carrée. Chaque famille FPGA comporte différentes combinaisons de ces blocs, avec diverses quantités de blocs logiques programmables, tel que représenté sur la figure 1.4.

Un autre type de structures logiques hétérogènes dispersées dans le matériel reconfigurable sont les blocs de mémoire, connu sous le nom de BRAM. Ces blocs permettent le stockage des données et des variables fréquemment utilisées, et permettent un accès rapide à ces valeurs en raison de la proximité de la mémoire à l'intérieur du

FPGA aux blocs logiques qui y accèdent [17]. Ces blocs embarqués peuvent également être utilisés pour la mise en œuvre des RAM simples, des mémoires FIFO, des machines d'état, etc. Ils sont configurables en blocs de 16 ou de 32 kilo-octets de type double ports permettant ainsi une écriture et une lecture indépendante sur chaque port avec une horloge différente.

Une autre tendance majeure dans la conception des FPGA est l'introduction de cœurs de microprocesseurs complets dans le tissu du FPGA. Ces cœurs de processeurs peuvent être, soit des macro-cellules spécialisées, dites processeurs Hardcores (ex : PowerPC et ARM), soit des modules synthétisables, dits processeurs softcores (MicroBlaze, NIOS-II, etc.) [18]. Ces softcores sont réalisés à partir de la logique programmable présente sur la puce. Ainsi, un concepteur peut mettre autant de processeurs softcore que la puce le permet. A titre illustratif, un processeur MicroBlaze occupe seulement 4% de la surface d'un FPGA de la famille Virtex5-LXT50 (composant à 28.800 cellules logiques). L'intérêt majeur des FPGA est la possibilité de faire du prototypage rapide. La présence des cœurs de processeurs à côté de la logique programmable à l'intérieur du FPGA rend possible de concevoir un système sur puce complet. Les avantages d'un tel système sont nombreux, il permet de réduire le nombre de puces donc le nombre de ressources et des pistes. Cela conduira, par conséquence, à la réduction de la consommation d'énergie et à une forte diminution de la complexité des cartes électroniques.

En plus de ces fonctionnalités citées, on peut inclure encore dans le FPGA d'autres fonctions de hauts niveaux appelées propriétés intellectuelles ou IP. Ces fonctions sont généralement représentées en utilisant un langage HDL de description de matériel, comme Verilog ou VHDL. Toutes les fonctions IP, que le concepteur décide d'utiliser, seront intégrées dans le corps principal de la conception et par la suite synthétisées et projetées dans le FPGA. La figure 1.4 présente deux scénarios pour l'utilisation d'un IP: dans le premier, la fonction peut être une fonctionnalité fournie par les fournisseurs FPGA, telle que l'interface TFT. Le deuxième scénario est la mise en œuvre d'un IP spécialement implémenté par l'utilisateur pour une application donnée en utilisant un ensemble de modules de base (blocs logiques, DSP, BRAM…etc).

Les composants FPGA modernes offrent également un appui au domaine de communications de données en intégrant différentes normes de transmission à haute

vitesse pour empêcher ainsi que le FPGA soit le goulot d'étranglement de la plateforme. Des exemples parmi d'autres de ces normes sont PCI Express, RapidIO et gigabit internet. De même, les vendeurs de FPGA offrent une variété de noyaux câblés optimisés tels que des contrôleurs DDR, des contrôleurs MAC, des transmetteurs LVDS…etc. [13].

Enfin, l'élément primaire pour gérer la cadence de traitement dans un FPGA est l'horloge. Les FPGAs de Xilinx assurent la gestion et l'ajustement de l'horloge à travers des blocs de gestion numérique d'horloge appelés DCM. Ces derniers permettent d'avoir des périodes d'horloge différentes générées à partir d'une horloge de référence unique. La plupart des systèmes disposent d'une horloge externe unique qui produit une fréquence d'horloge fixe. Cependant, il y a un certain nombre de raisons pour lesquelles un concepteur peut avoir besoin de fonctions logiques fonctionnant à des fréquences différentes soient pour travailler de façon autonome ou en conjonction avec d'autres modules hétérogènes. L'avantage d'utiliser un DCM est que les horloges générées auront moins de gigue.

1.2.4 Configuration d'un FPGA

Pour configurer un FPGA, un bitstream de configuration est nécessaire. C'est un fichier binaire dans lequel les informations de configuration pour un périphérique particulier sont stockées. Les bitstreams peuvent être partiels ou complets. Un bitstream complet (Merged bitstream) configure la mémoire de configuration de l'ensemble du circuit. Il est utilisé pour la conception statique ou au début de l'exécution d'un système de reconfiguration dynamique afin de définir l'état initial des cellules SRAM. Le bitstream partiel configure seulement une partie du circuit, il est utilisé pour la reconfiguration partielle. Les FPGA de Xilinx fournissent différents moyens de configuration à savoir le port JTAG, l'interface SelectMAP pour la configuration de plusieurs FPGA, ou encore le port de configuration interne (ICAP) utilisé pour l'auto-reconfiguration.

1.3 Flot de conception traditionnel des FPGA de Xilinx

Le flot de conception FPGA standard, comme représenté sur la figure 1.5 transforme une description de conception FPGA en un bitstream configurable. En règle générale, la description de conception est écrite dans un langage de description matérielle,

comme Verilog, VHDL, Système C ou Handel C. Les modèles les plus complexes font souvent appel à des outils d'intégration de systèmes pour construire la description de la conception initiale à partir de bibliothèques des composants de propriétés intellectuelles (IP), qui sont utilisés avec d'autres fonctions définies par l'utilisateur pour créer des applications System-on-Chip. Chaque passage à travers le flot de conception standard est indépendant des passages suivants. Le bitstream, produit à la fin de flot, ne dépend que de la spécification de la conception initiale et il est utilisé pour configurer le composant entier.

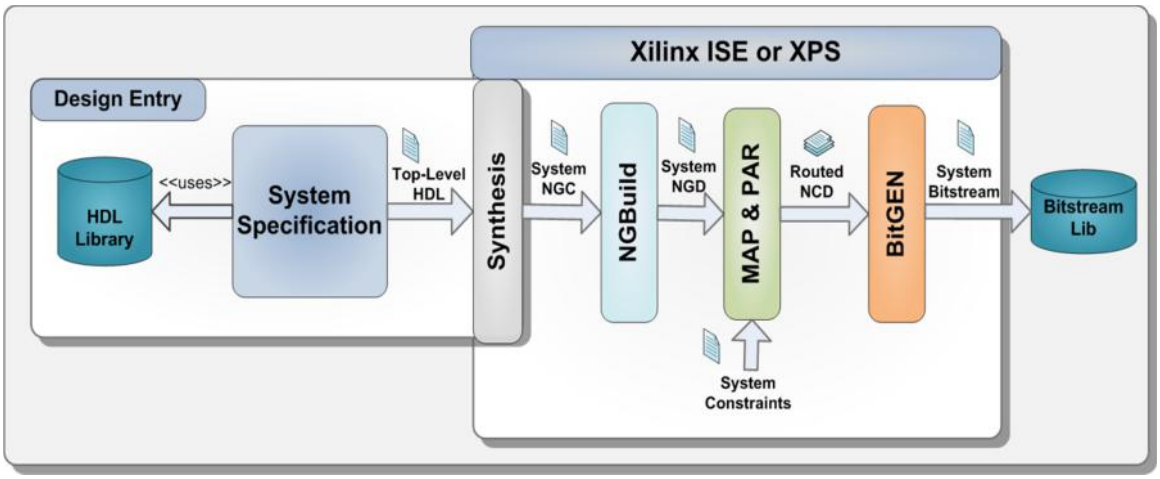


Figure 1.5: Flot de conception traditionnel des FPGA de Xilinx

Une description approfondie de chacune de ces étapes est en dehors du cadre de notre travail compte tenu de leur complexité. De plus amples détails sont disponibles au guide de référence du système de développement de Xilinx [19]. Néanmoins, il est important de comprendre les principes de base afin de donner un aperçu de la méthodologie. Chacune des étapes est donc brièvement discutée comme suit :

Design Entry : Dans cette étape du processus de conception, le design est créé en utilisant soit un éditeur schématique, soit à base de texte (HDL) ou les deux (ie. L'outil Xilinx ISE). De la même manière, la plateforme peut être basée sur une combinaison d'un processeur embarqué, des périphériques et des modules logiques créés par l'utilisateur. Dans un tel scénario, la conception peut être créée en utilisant un outil spécialisé, comme Xilinx Platform Studio (XPS) [20]. Ces outils permettent au concepteur de créer un design dans un environnement convivial et tout l'effort de conception pour obtenir la description de haut niveau VHDL est considérablement réduit.

Synthèse: Habituellement, la spécification de conception est faite indépendamment de la famille de FPGA ciblé par l'implémentation. Toutefois, si certaines caractéristiques spécifiques d'un FPGA sont nécessaires, alors ces contraintes sont prises en compte dès le début. Néanmoins, lors de la phase de synthèse, le support d'outil vise un circuit FPGA particulier. Et procède alors à la conversion du format logique qui a été créée à l'étape précédente (Design Entry) au format de fichier physique. L'information physique est contenue dans le fichier NGC pour les FPGAs de Xilinx.

NGDBuild: Cette étape effectue toutes les démarches nécessaires pour lire un fichier Netlist de format NGC ou EDIF et crée un fichier NGD décrivant la conception logique (une conception logique en termes d'éléments logiques telles que des portes ET, des portes OU, des décodeurs, des bascules et les BRAM). Le fichier NGD résultant d'une exécution de NgdBuild contient à la fois une description logique de la conception réduit au Xilinx NGD (base de données générique native) et une description en termes de la hiérarchie originale exprimée dans la liste des connexions d'entrée. Le fichier NGD de sortie peut être associé au composant FPGA spécifique.

MAP: Les algorithmes de mappage traduisent une conception logique dans une description qui peut être implémentée dans un composant FPGA spécifique. L'entrée de mappage est un fichier NGD, qui contient une description logique de la conception en termes de composants hiérarchiques utilisés pour développer la conception, des primitives Xilinx de bas niveau et de certain nombre de fichiers NMC contenant chacun la définition d'une macro physique. MAP effectue d'abord une vérification de la conception contenu dans le fichier NGD via l'outil DRC (Design Rule Check). Ensuite, il mappe la logique dans les composants de base (cellules logiques, les cellules d'E-S et les autres composants) contenus dans le FPGA cible en tenant compte des contraintes temporelles et de localisation qui peuvent être imposées. La conception de sortie est un fichier NCD (Native Description Circuit), une représentation physique de la conception mappée sur les composants d'un FPGA de Xilinx.

PAR: Après avoir subi la traduction nécessaire et ramené dans le format NCD (Circuit Description), la conception est prête pour le placement et le routage. Cette phase est effectuée par l'outil Xilinx PAR (Place & Route) en prenant un fichier NCD, place la conception, réalise son routage dans le circuit et renvoie un autre fichier NCD qui sera utilisé pour générer un fichier bitstream de configuration.

Bitstream Generation: produit un bitstream de configuration du circuit. Après que le design a été complètement routé, il est nécessaire de configurer le circuit pour qu'il puisse exécuter la fonction désirée. Cette phase est assurée par l'outil Xilinx Bitgen (Bitstream Generation) qui prend un fichier NCD complètement routé et produit un fichier binaire de configuration. Ce dernier contient toutes les informations de configuration et définit la logique interne avec toutes ses interconnexions dans le FPGA. Les données binaires du fichier bitstream peuvent être directement chargées dans la mémoire de configuration du FPGA cible.

1.4 Limitations de flot de conception traditionnel

En dépit du succès de FPGA dans plusieurs domaines, de nombreux concepteurs d'architectures des systèmes ont trouvé la manière traditionnelle d'implémentation dans les FPGA trop restrictive [1], car tout changement ayant lieu après la mise en place de la conception dans le circuit, exigerait de reconsidérer la totalité des étapes de spécification définies dans la figure 1.5. Ce fait pose un problème dans les applications où une adaptabilité plus élevée sur l'application serait souhaitable. Les FPGA sont basés sur des cellules de configuration SRAM, en conséquence, le bitstream obtenu à partir du flot conception est chargé sur le FPGA à la mise sous tension, lorsque le composant est éteint, la totalité de ses fonctionnalités est perdue. Tels que définis précédemment, les FPGA sont la solution préférée dans les applications où la fabrication de millions de circuits ASIC n'est pas envisagée. Les FPGA fournissent également une plus grande flexibilité et peuvent être en plus reprogrammés sur le terrain (d'où le nom Field dans le nom du FPGA). Cela leur permet alors de faire des mises à jour similaires aux applications à base de processeurs, qui ne nécessitent pas habituellement un changement d'usine pour effectuer des nouvelles tâches.

Néanmoins, si l'application doit être modifiée en temps d'exécution [21], l'application exigerait d'arrêter le FPGA et de le reconfigurer dans sa totalité telle qu'illustré dans la figure 1.6. En outre, ce processus nécessite traditionnellement l'utilisation d'un processeur externe en charge de retrouver les bitstreams de configuration de la mémoire non-volatile en utilisant une machine d'état relativement simple [21]. L'utilisation du processeur externe augmente les ressources et donc le coût d'un tel système. Un autre problème se pose, il consiste à mémoriser la totalité du bitstream pour

chaque configuration même si seulement un petit changement est nécessaire comme le montre la figure 1.6.b, où seules les fonctions A et C sont à remplacer par des modules D et E. Cette situation entraine une augmentation des besoins en mémoire de stockage ce qui augmente davantage le coût du système.

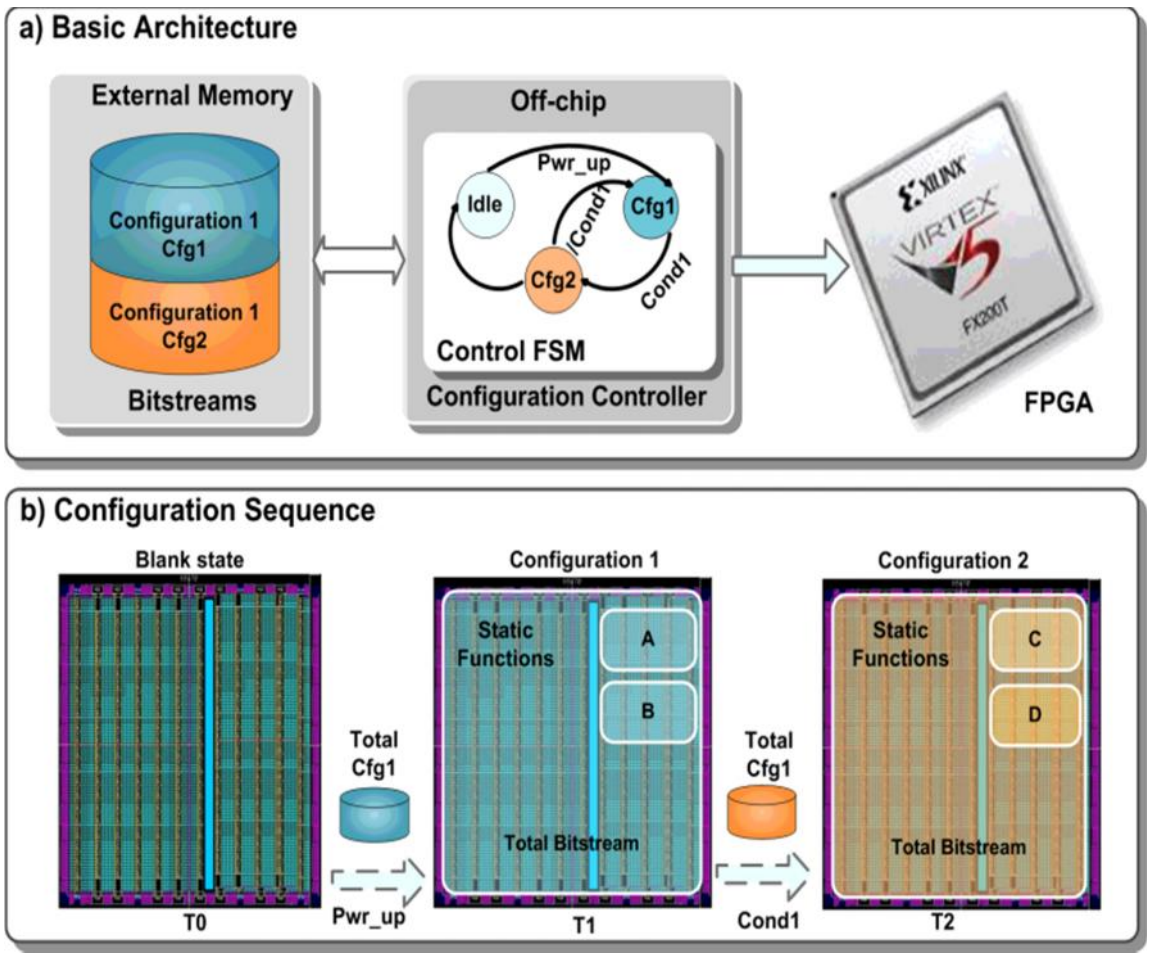


Figure 1.6: Reconfiguration dynamique procédée par le flot de conception traditionnel

D'autre part, le temps pris par le système pour qu'il soit reconfiguré qui est de l'ordre de quelques secondes pour les grands FPGA n'est pas négligeable, il sera perdu pour effectuer le processus de reconfiguration du FPGA entier [22] [23]. Cela sans tenir compte encore du temps pris pour récupérer les bitstreams à partir de la mémoire non volatile [24].

1.5 Reconfiguration dynamique partielle

Au cours des deux dernières décennies, le concept de la reconfiguration dynamique (RD) a été introduit pour décrire des systèmes capables d'échanger différentes configurations dans le matériel reconfigurable au cours de l'exécution selon le besoin de l'application [2]. Cependant, la nature des flots de conception traditionnels et d'autres fondements technologiques ont empêché l'adoption de la technique RD et avaient fait leur œuvre carrément prohibitive. Ces dernières années, le concept de la reconfiguration dynamique partielle (DPR) a été introduit comme une nouvelle méthode afin d'augmenter la flexibilité des solutions à base de FPGA. Grâce à l'utilisation du DPR, un module du système peut être remplacé par d'autres fonctionnalités ou tout simplement retiré de tissu reconfigurable à la demande, comme le montre la figure 1.7. Cela est fait sans retenir le circuit pour effectuer la modification souhaitée et donc sans impact sur l'exécution du système.

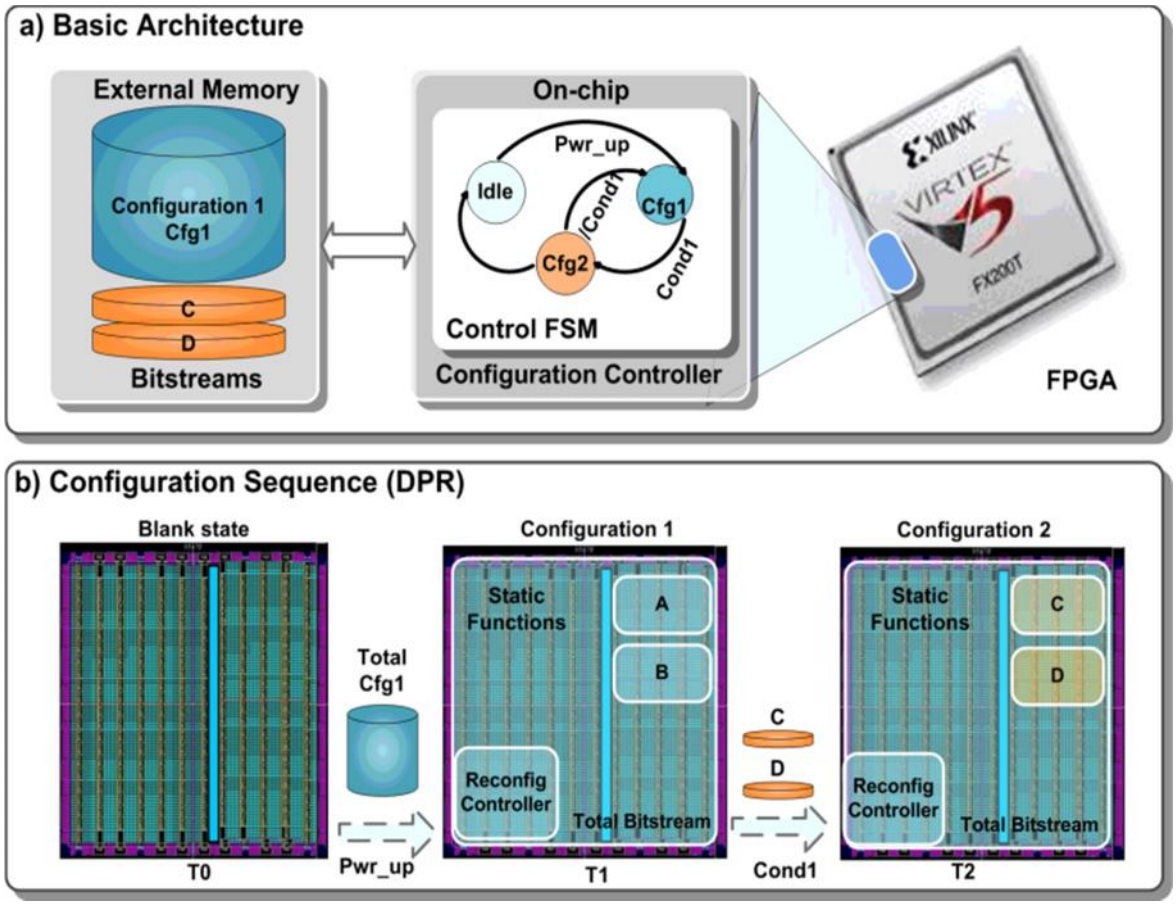


Figure 1.7: Reconfiguration dynamique partielle

Le processus de reconfiguration partielle est réalisé en séparant la logique en une région statique et plusieurs d'autres reconfigurable dynamiquement, ce qui signifie qu'une partie du FPGA reste opérationnelle pendant toute la durée d'exécution de l'application, tandis que d'autres régions du FPGA peuvent être reconfigurées à la volée. A titre d'illustration, notre exemple de la figure 1.7 montre un système contenant un microprocesseur et quelques périphériques appartenant à la logique statique, alors que deux modules forment la partie dynamique du système. Chaque région dynamique est reconfigurée de façon indépendante, en chargeant uniquement ses données de configuration (à travers un bitstream partiel), et ce en vue de modifier sa fonctionnalité.

La DPR présente de nombreux avantages intrinsèques. En plus de fournir une plus grande flexibilité à l'application, le temps de reconfiguration est considérablement diminué, puisque seule une fraction de la configuration totale sera extraite de la mémoire externe et placée dans le circuit. De plus, ces bitstreams partiels sont beaucoup plus petit qu'un bitstream complet, ce qui conduit à un besoin plus réduit en mémoire externe, comme représenté sur la figure 1.7 b. L'utilisation d'un processeur, comme dans le cas précédent, est également nécessaire, mais il peut maintenant être intégré dans la région statique de la logique (en utilisant par exemple un processeur synthétisable sur la puce comme le Microblaze) permettant ainsi de réduire considérablement la complexité du système.

1.8 Conclusion

Ce chapitre nous a permis de fournir une description plus au moins détaillée sur les architectures reconfigurables à base de FPGA et les outils nécessaires associés pour la mise en œuvre de systèmes embarqués complexes. Il nous a permis également de montrer l'intérêt de la reconfiguration dynamique comme solution servant à augmenter l'efficacité d'un FPGA. Le premier avantage qui suscite un intérêt croissant est la flexibilité introduite par la reconfiguration dynamique des FPGA qui se manifeste à deux niveaux. Le premier niveau permet au concepteur, d'adapter facilement son application pour faire face à de nouvelles contraintes ou pour remplacer un algorithme par un autre plus efficace. Cette flexibilité assure une certaine évolution du système. Le second niveau se situe sur le choix des tâches à exécuter, qui peut se faire dynamiquement en fonction des données à traiter. On peut donc tout à fait imaginer que les données d'entrée puissent influer en

temps réel sur l'enchaînement des algorithmes. Le matériel devient extrêmement malléable et contrôlable par le logiciel. Un second avantage réside dans le fait qu'en allouant des ressources logiques à plusieurs tâches, il est possible d'implémenter une application de taille plus grande que la mémoire physique disponible sur le FPGA. Par conséquent, un FPGA reconfigurable dynamiquement peut offrir une quantité plus importante de ressources logiques qu'il en dispose. Dans le prochain chapitre de ce travail de thèse, nous allons exposer plus en détail les techniques utilisées pour la reconfiguration dynamique partielle et les flots de conception associés pour sa mise en œuvre.

Chapitre 2

RECONFIGUATION DYNAMIQUE PARTIELLE

CHAPITRE 2

RECONFIGUATION DYNAMIQUE PARTIELLE

2.1 Introduction

Quelques années auparavant, les algorithmes qui pouvaient être mis en œuvre sur une puce FPGA unique étaient très peu répandus. En 1995, par exemple, les plus grands FPGA qui pouvaient être programmés disposaient d'environ 15 000 portes logiques au maximum. Plus récemment, les FPGA ont dépassé le million de portes logiques. Les chercheurs et les concepteurs ont, donc pensé à réaliser des systèmes auto adaptatifs. Pour cela le circuit logique doit être capable de procéder à une modification de sa fonctionnalité au cours de son fonctionnement. Les FPGA basiques ne le permettent pas, il est nécessaire d'arrêter le système pour y implémenter une nouvelle configuration. Cette situation a amené les sociétés ATMEL et Xilinx à concevoir des nouvelles architectures capables de procéder à une modification de leurs ressources sans arrêter le système d'où la naissance des premières architectures reconfigurables dynamiquement.

La reconfiguration est un terme composé du préfixe « re » qui exprime la réitération (refaire) ou le retour à un état antérieur (recourber). Donc la reconfiguration est une réitération d'une configuration, c'est une nouvelle modification ou un nouveau réglage de paramètres qui permet l'optimisation du fonctionnement d'un système. Pour les FPGA, la reconfiguration désigne le fait que l'architecture matérielle du système soit capable de s'adapter en changeant de forme ou de fonction. Le FPGA reconfigurable peut donc changer sa configuration à n'importe quel moment. Cette reconfiguration peut être dynamique, car le processus de réitération permet d'avoir à chaque nouvelle configuration une nouvelle action. On peut donc définir les FPGA comme des systèmes reconfigurables et c'est leur exécution qui est dynamique.

La capacité de l'architecture à s'adapter de manière dynamique à son environnement est caractérisée par le temps que met le système à se reconfigurer. Cette caractéristique est directement liée au temps d'accès à la mémoire de configuration SRAM. Le temps de reconfiguration dépend des technologies FPGA et des tâches matérielles utilisées, il peut varier de quelques microsecondes à quelques secondes [25].

Les systèmes qui utilisent les architectures reconfigurables sont souvent utilisés pour réaliser des accélérateurs matériels pour traiter des applications gourmandes en ressources et en temps de calcul [26, 27, 28]. La mise en œuvre de ce type de fonctionnement permet de reconfigurer autant de fois que nécessaire l'architecture matérielle du système tout au long de son exécution. Les tâches matérielles sont alors obtenues par un découpage de l'application sous forme de partitions exécutées séquentiellement [29, 30]. Les ressources sur lesquelles sont exécutées ces partitions sont le résultat du découpage spatial du circuit reconfigurable. La manière dont un algorithme sera implémenté sur une architecture reconfigurable dynamiquement va déterminer d'une part le niveau de performance dans le traitement des données et d'autre part, le niveau de dynamicité et de flexibilité dont cette architecture pourrait faire preuve. Pour réaliser cette adéquation, il faut utiliser des techniques de conception et des architectures adéquates. Toutes ces architectures et ces techniques de conception sont en constante évolution pour améliorer la flexibilité des FPGA reconfigurables.

La suite de ce chapitre est organisée de la manière suivante : dans une première partie, nous décrirons les différents modes de reconfiguration, les avantages de la reconfiguration partielle et la terminologie pour la reconfiguration des FPGA de Xilinx utilisée dans ce manuscrit. Puis, une présentation dédiée aux techniques de reconfiguration des FPGA et à l'évolution des flots de conception existants associés à la reconfiguration partielle. Nous montrerons, ensuite, les limitations de ces flots et les problèmes rencontrés pour bien gérer le placement des tâches matérielles dans le FPGA. En fin, nous verrons quelques solutions proposées dans les différents travaux de recherche et ainsi notre contribution dans le domaine.

2.2 Les divers scénarios de reconfiguration dans les FPGA

Un facteur de différenciation des FPGA réside à leurs modes (ou possibilités) de reconfigurations qui varient d'une famille à l'autre. Ces modes sont souvent liés à la technologie de fabrication et peuvent être des critères déterminants lors du choix du FPGA cible pour implémenter une application. On peut classifier les architectures reconfigurables en trois catégories suivant leurs modèles de reconfiguration [31]. Ces modèles, qui peuvent s'appliquer à l'un ou l'autre des FPGAs existants, sont:

2.2.1 La reconfiguration à contexte unique (Single Context)

Elle est encore appelée Reconfiguration totale car la reconfiguration se fait entièrement sur toute la surface de la matrice de configuration du FPGA. Tous les bits de la matrice de configuration sont donc réécrits, même s'ils n'ont pas changé entre deux configurations (Figure 2.1.a) ce qui présente évidemment une perte de temps. La reconfiguration à contexte unique est utilisée pour des applications statiquement reconfigurables (SRD - Statically Reconfigurable Design). Plusieurs applications sont programmées successivement sur le FPGA (par chargement du fichier bitstream correspondant); un Reset du FPGA est effectué entre deux programmations.

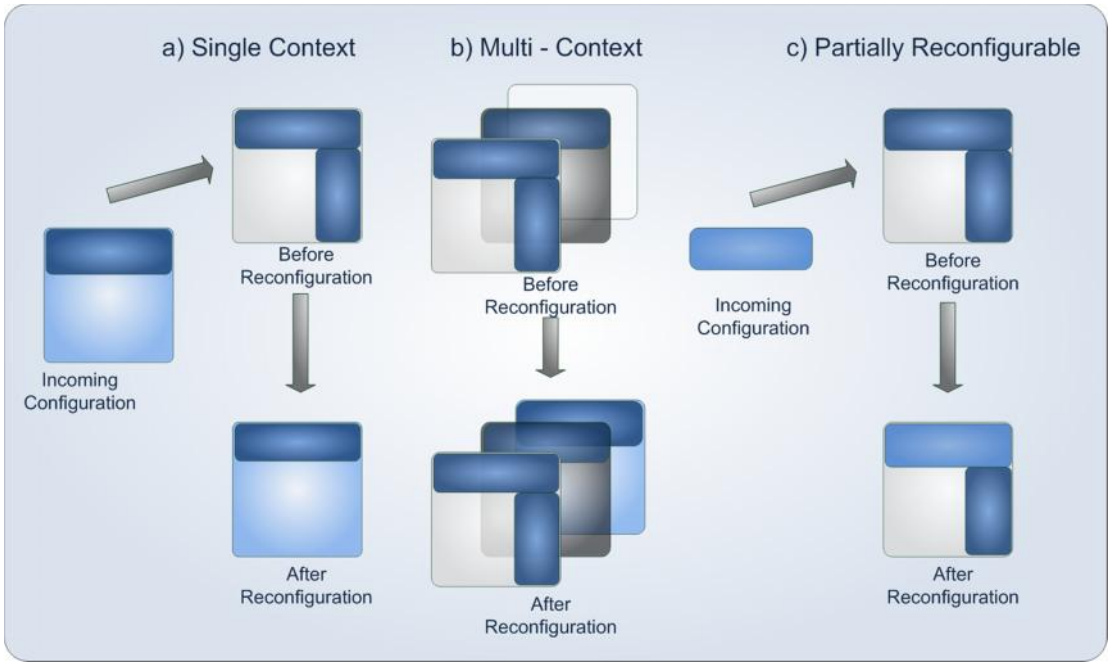


Figure 2.1 : Les différents modes de reconfiguration

2.2.2 La reconfiguration multi-contextes (Multi-Context)

Elle se fait sur des FPGA ayant plusieurs mémoires de configuration se trouvant sur des plans différents (Figure 2.1.b). On peut ainsi "charger" plusieurs plans de configurations dans le FPGA, et le passage d'une configuration à l'autre (commutation de contexte) se fait par commutation d'un plan mémoire à un autre. Par rapport à la reconfiguration à contexte unique, le temps de reconfiguration en est fortement accéléré. En outre, une mémoire inactive peut être reprogrammée pendant que l'autre est active sur le FPGA. Une sorte de mémoire cache de contexte existe alors dans le FPGA.

2.2.3 La reconfiguration partielle (Partially Reconfigurable)

Contrairement à la reconfiguration à contexte unique ou multi-contextes, on peut configurer partiellement le FPGA afin de n'utiliser que les ressources nécessaires à l'implémentation de l'application (Figure 2.1.c); on réduit ainsi la consommation en énergie et en ressources logiques du FPGA, ainsi que le temps de reconfiguration. En outre, une portion du FPGA peut être reconfigurée dynamiquement (c'est-à-dire reconfigurer une partie du FPGA pendant que l'autre est active), ce qui cache la latence due à la reconfiguration qui peut s'observer dans la reconfiguration à contexte unique ou multiple. Reconfigurer dynamiquement permet l'implémentation matérielle de beaucoup plus de sections de l'application et d'accélérer ainsi l'exécution.

2.3 Avantages de la reconfiguration partielle

La reconfiguration partielle (DPR) permet d'utiliser plus de matériel que celui présent physiquement dans le FPGA en virtualisant les tâches matérielles et les plaçant par la suite à la demande dans le FPGA. Cette possibilité peut être exploitée pour réduire la taille du FPGA et sa consommation d'énergie globale. Cela permet aussi d'exécuter un algorithme et optimiser son implémentation en fonction de l'ensemble de ses paramètres et ses données. En outre, la DPR offre des avantages au niveau système pour l'électronique professionnelle [32]. Certains des scénarios dans lesquels la DPR a fait ses preuves sont répertoriées comme suit :

• la virtualisation matérielle : Souvent, les parties d'un programme qui peuvent être accélérées grâce à l'utilisation de matériel reconfigurable sont complexes ou trop nombreux pour être chargées simultanément sur le matériel disponible. Pour ces cas, il peut être bénéfique d'échanger selon le besoin des configurations différentes dans le FPGA. Puisque le placement dans ce cas se fait à la demande, cela permettrait de promouvoir d'une façon efficace, similairement à un traitement logiciel, la notion de virtualisation des tâches matérielles.

• Réduction de l'utilisation des ressources logiques et de puissance : Un système pourrait avoir besoin de contenir plusieurs implémentations d'un module qui effectuent des tâches similaires mais qui s'excluent mutuellement. Cette multiplicité augmente la quantité de logique nécessaire pour loger les différentes implémentations d'un même module. La

reconfiguration partielle est capable de réduire le montant de la logique et de réduire par conséquent, la consommation de puissance statique requise en chargeant à chaque moment donné juste l'implémentation requise par l'application.

• Capacité de survie. Il y a beaucoup de recherche dans les domaines des systèmes tolérants aux fautes où ils peuvent fonctionner en mode dégradé lorsqu'une partie de leurs systèmes est endommagée. Ceci peut être réalisé par configuration partielle en déplaçant la fonction initialement placée dans la zone endommagée vers une autre zone seine dans le FPGA sans interrompre l'exécution de l'application.

• Changement de mission : Un système peut être reconfiguré pour une nouvelle mission sans interrompre son fonctionnement. Cette fonctionnalité permet d'ajouter des fonctionnalités qui n'ont pas été précisées au début, ou bien de réaliser des mises à jour de ces fonctionnalités en vue d'améliorer les performances des applications.

2.4 Terminologies pour la reconfiguration dynamique partielle

La terminologie spécifique au domaine de la reconfiguration est montrée dans la figure 2.2, nous la citons comme suit :

• Région reconfigurable : Région délimitée initialement dans le circuit FPGA et destinée à être le siège des reconfigurations dynamiques.

• Région statique *:* Région délimitée qui, à l'inverse de la région reconfigurable, est destinée à un fonctionnement purement statique.

• Partition Reconfigurable Partiellement (Partial Reconfigurable Partition) : est une partition de la région reconfigurable. Une région reconfigurable peut être composée de plusieurs partitions PR.

• Tâches matérielles ou Intellectual Property (IP core) *:* Blocs fonctionnels complexes réutilisables.

• Module Partialement Reconfigurable (Module PR) : tâche matérielle qui peut peupler une région reconfigurable. Il peut y avoir plusieurs modules reconfigurables par région. Un module PR peut aussi occuper une ou plusieurs partitions PR.

• Wrapper *:* Circuit matériel permettant de connecter une tâche matérielle au reste du système.

• Interfaces de connexions *:* Point de connexion d'une région reconfigurable.

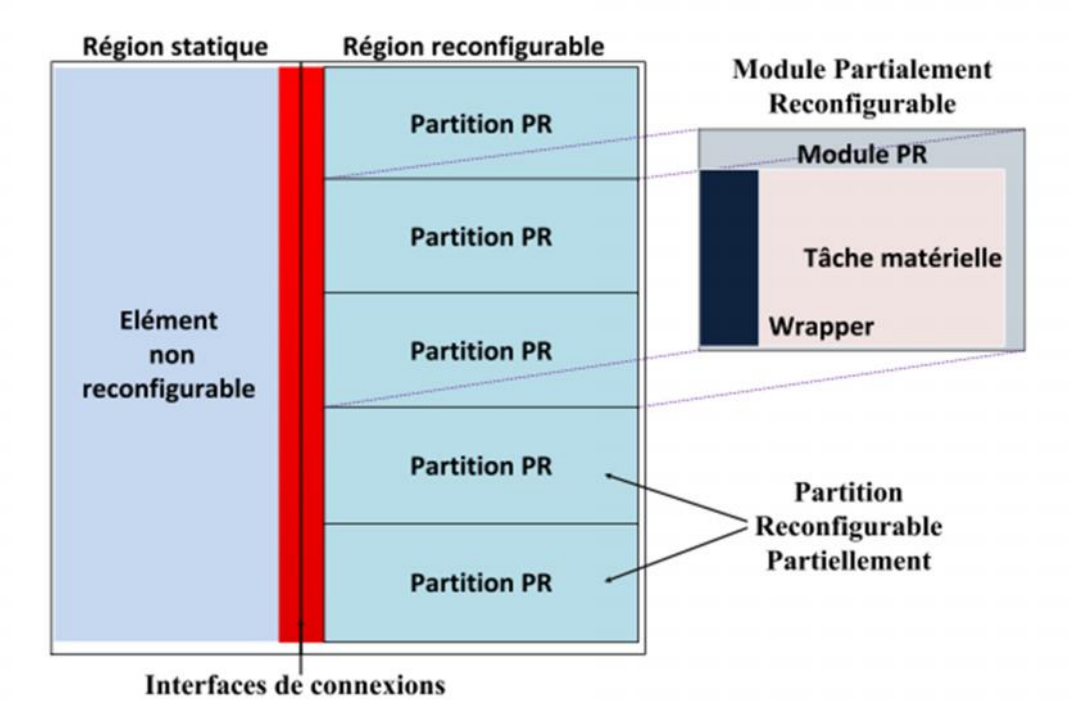


Figure 2.2 : Terminologies pour la reconfiguration.

2.5 Techniques de reconfiguration des FPGA

La reconfiguration existe depuis plusieurs années, son évolution a permis de dégager différentes méthodes. Nous allons voir dans un premier temps les moyens dont nous disposons pour utiliser la reconfiguration. Ensuite nous verrons qu'il existe plusieurs flots de conception pour mettre en œuvre la reconfiguration partielle des circuits FPGA de chez Xilinx. Puis nous établirons un comparatif sur les systèmes d'interconnexions des régions reconfigurables. Enfin, nous présenterons les différentes contraintes en fonction des technologies et des flots utilisés.

2.5.1 Moyens pour la reconfiguration partielle

La reconfiguration d'un FPGA peut être réalisée en interne sur le FPGA. Dans le cas d'une demande de reconfiguration externe au FPGA, un PC dialogue avec le FPGA. Cette communication est réalisée par l'intermédiaire du port JTAG du FPGA. Les bitstreams de reconfigurations partielles ou complets sont envoyés sur le FPGA. Cette méthode demande l'action d'un acteur extérieur et elle n'est donc pas adéquate pour un système auto-adaptatif.

La reconfiguration d'un système auto-adaptatif sur les FPGA Xilinx est mise en œuvre en se servant du "Port d'Accès de Configuration Interne" (ICAP). Un bitstream partiel est écrit dans l'ICAP, ce bitstream reconfigure alors les zones spécifiées du FPGA. La communication avec l'ICAP peut être mise en œuvre par un microprocesseur embarqué (un PowerPC ou un Microblaze).

L'ICAP fournit un accès interne à la logique de configuration du FPGA. Grâce à l'interface de l'ICAP, les données de configuration peuvent être chargées dynamiquement dans la mémoire de configuration du FPGA. Il est également possible d'effectuer une relecture des données de configuration (Readback) de la mémoire de configuration ou de lire les registres d'état de la logique de configuration avec l'interface ICAP. Comme le montre la figure 2.3, les interfaces de l'ICAP vers la mémoire de configuration lui permettent d'accéder aux ressources configurables du FPGA.

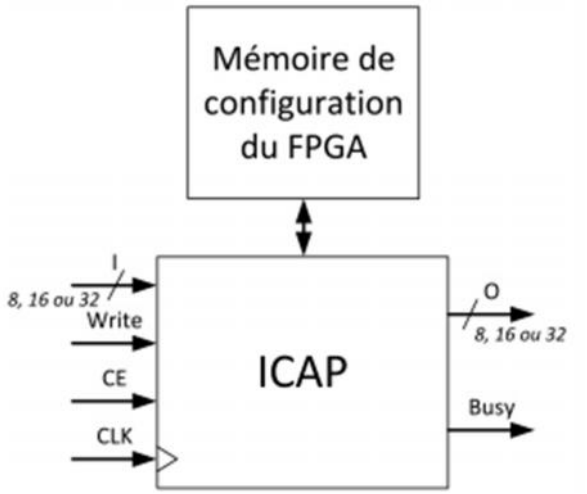


Figure 2.3 : Architecture de l'ICAP (Port d'Accès de Configuration Interne).

L'interface ICAP comprend des ports de données distinctes pour la lecture (O) et l'écriture (I) des données de configuration. La largeur des ports de données de configuration peut être configurée pour être de 8 ou 32 bits pour les Virtex-4, et de 8, 16 ou 32 bits pour les Virtex-5 et 6 [33, 34]. Xilinx précise que l'interface ICAP ne doit pas être cadencée plus rapidement que la fréquence recommandée par défaut de 100 MHz. Avec une fréquence de 100 MHz et une largeur de 32 bits d'interface de données, l'interface ICAP fournira alors une vitesse maximale de reconfiguration par défaut de 400 Mo/s (si le bitstream est stocké sur les BRAM du FPGA).

2.5.2 L'évolution des flots de conception

Dans cette partie nous allons décrire les flots de conception pour la reconfiguration partielle proposés par Xilinx du plus ancien au plus récent.

2.5.2.1 Conception modulaire de la reconfiguration partielle

La technique la plus ancienne est appelée « Module-based partial reconfiguration » (MBPR). Elle permet la reconfiguration de modules prédéfinis au préalable [35]. Le but de cette méthode est de reconfigurer des portions distinctes d'un FPGA tandis que le reste du dispositif continue de fonctionner. Cette méthode est basée sur les flots de conception modulaire. La conception modulaire est une méthode de développement conçue par Xilinx [11]. Cette méthode permet à une équipe d'ingénieurs de travailler de façon autonome sur différents modules d'une conception et de les fusionner ensuite en une seule conception pour un FPGA. Chaque module peut être conçu par un membre de l'équipe tandis que le chef d'équipe travaille toujours sur la conception de haut niveau. Cette méthode permet un gain de temps et la possibilité de modifier un module tout en laissant les autres inchangés, parce que chaque module est considéré comme un module indépendant. Par contre, la communication entre les modules et le reste de la conception doit être assurée par des bus macro placés aux frontières des modules. Le bus macro fournit une communication fixe entre les différents modules de la conception. Tous les modules de base qui seront partiellement reconfigurables devront contenir un ensemble de connexions compatible avec les bus macro. Les connexions sont réalisées et placées à la main pendant la phase de conception. Sans cette attention particulière, les communications avec les modules reconfigurables ne seraient pas possibles, car il serait impossible de garantir l'acheminement des communications entre les modules. Pour chaque reconfiguration partielle effectuée, le bus macro est utilisé pour établir un routage invariable avec les autres modules statiques ou reconfigurables, en garantissant des connexions correctes.

Les bus macro [32] permettent la communication à travers des buffers à trois états. Il faut donc veiller à ce que des précautions soient prises pour que les buffers à trois états ne soient pas en cours de modification pendant une reconfiguration. Les bus macro sont définis par une description HDL. Cette description doit faire en sorte que tous les signaux du module reconfigurable qui sont utilisés pour communiquer avec un autre module utilisent un bus macro comme le montre la figure 2.4. L'insertion des bus macro doit être

réalisée de telle sorte qu'ils ne chevauchent pas une colonne de type DSP ou type BRAM. Les bus macro sont le moyen de communication entre les modules reconfigurables et la partie statique, par conséquent toutes les connexions passent par un bus macro à l'exception des signaux d'horloge.

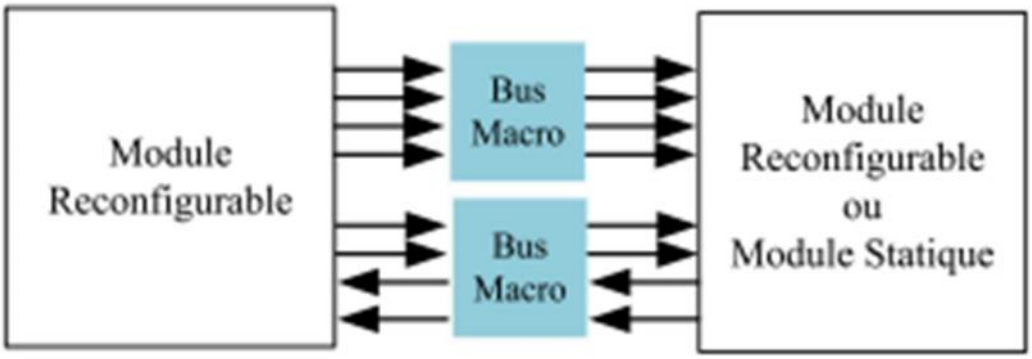


Figure 2.4 : Placement des bus macro.

L'architecture d'un bus macro montrée sur la figure 2.5 montre qu'un bus macro offre 4 bits de communication. La mise en œuvre actuelle d'un bus macro utilise huit buffers à trois états (TBUFs) reliés par plusieurs lignes qui permettent à un bit d'information de passer soit de gauche à droite ou de droite à gauche. Un bus macro utilise deux bus d'entrées (Ai et Bi) et deux bus de sorties (Ao et Bo) chacun sur quatre bits. Le bus macro utilise aussi deux bus de configuration (As et Bs) sur quatre bits pour modifier l'état des buffers à trois états.

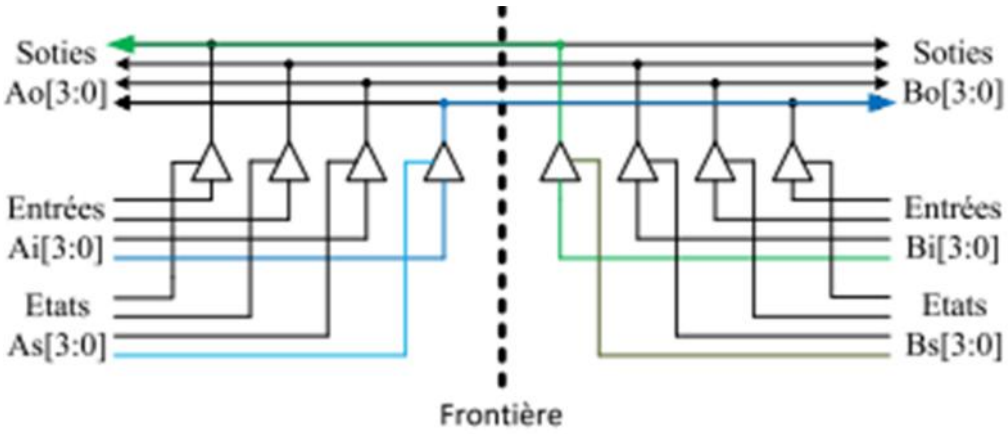


Figure 2.5: Architecture d'un Bus Macro.

Par exemple, si on souhaite que le bus macro établisse une commutation sur 1 bit

entre le module de droite et celui de gauche, on doit définir As(0) à 1 et Bs(0) à 0. Ensuite, on connecte le module émetteur sur Ai(0) et le module récepteur sur Bo(0).

La phase d'assemblage finale est le processus consistant à combiner chacun des modules individuels dans une conception FPGA complète. Le placement et le routage sont réalisés au cours de la phase d'implémentation de chaque module. Le placement et le routage sera ensuite conservée pour maintenir les connections et les performances de chaque module. Chaque module reconfigurable sera placé sur une colonne reconfigurable de l'architecture du FPGA. Le flot de reconfiguration partielle exige que le bitstream initial qui est chargé dans le dispositif FPGA soit une conception complète. Ceci est nécessaire afin que tous les modules non reconfigurables soient placés et verrouillés, et que seuls les éléments reconfigurables de la conception vont changer lors de la reconfiguration partielle.

2.5.2.2 Difference-based partial reconfiguration (DBPR)

Par la suite une nouvelle technique a vu le jour appelé « Difference-based partial reconfiguration » (DBPR) [11]. Elle peut être utilisée quand un petit changement est apporté à la conception. Elle est particulièrement utile en cas de changement d'équation des LUTs ou d'une mémoire dédiée. Le bitstream partiel ne contient que des informations sur les différences entre la structure de conception qui réside dans le FPGA et son nouveau contenu. Il existe deux méthodes de différence basée sur la reconfiguration connue (front-end et back-end). La première est basée sur la modification de la conception dans les langages de description matérielle (HDL). Il est clair qu'une telle solution exige la pleine répétition de la synthèse et des processus de mise en œuvre. Le back-end permet d'apporter des modifications au stade de la mise en œuvre du prototypage. Par conséquent, il n'est pas nécessaire de resynthétiser la conception. L'utilisation de ces deux méthodes aboutit à la création d'un bitstream partiel grâce à la comparaison de deux conceptions qui peut être utilisée pour une reconfiguration partielle du FPGA. La commutation de la configuration d'un module à un autre est très rapide, vu que les différents bitstreams peuvent être plus petits. Cette méthode ne peut pas être entièrement adaptable, car les modifications apportées à la conception ne sont pas importantes et l'intégrité du signal n'est pas assurée sur les limites des modules reconfigurables.

2.5.2.3 Early Access Partial Reconfiguration

Un flot d'accès rapide à la reconfiguration partielle a vu le jour pour pallier aux contraintes des deux flots précédents. Ce flot, nommé Early Access Partial Reconfiguration (EAPR) [12], permet de faciliter la conception d'un système reconfigurable partiellement. Il est possible avec ce flot d'adapter une région à un module partiellement reconfigurable. L'exigence du flot MBPR de placer l'ensemble des régions reconfigurables sur des colonnes est supprimée. Le flot EAPR permet désormais aux régions reconfigurables d'être de toute taille tant que sa forme est rectangulaire. La plus petite région reconfigurable qui peut être définie est une «frame», qui s'étend sur toute la hauteur d'un domaine d'horloge du FPGA.

De plus, un nouveau système de communication appelé slice bus macro a été développé par Hübner et. al. [36] comme alternatives aux bus macro (basée sur des buffers trois états). Les bus macro basés sur des slices (ou des LUT) sont représentés sur la figure 2.6, chaque bus macro se compose de deux blocs logiques configurables (CLB). Les bus macro basés sur les slices comportent un certain nombre d'avantages par rapport à ceux basés sur des buffers trois états. Le premier avantage est dû à l'évolution de la technologie des slices bus macro utilisant des LUT. Ceci est très important vu que les buffers à trois états ont disparu depuis la série 4 pour les FPGA Virtex. Ils permettent aussi d'améliorer considérablement les performances de synchronisation et simplifient le processus de construction d'une conception. De plus des fonctionnalités supplémentaires ont été ajoutées pour désactiver les signaux sortants. Cette fonction est très utile, car le comportement des signaux émanant du module n'est pas connu pendant une reconfiguration (ou si aucun module n'est présent). Comme on peut le voir sur la figure 2.6 les slices bus macro sont placés dans les modules alors que le bus macro se trouve entre les modules.

2.5.2.4 Partition based partial reconfiguration

Divers inconvénients du flot EAPR ont été résolus avec l'apparition du flot « Partition based partial reconfiguration » (PBPR) [37]. Premièrement, ce flot permet un plus grand degré de liberté sur la forme des régions reconfigurables par rapport aux flots précédents.

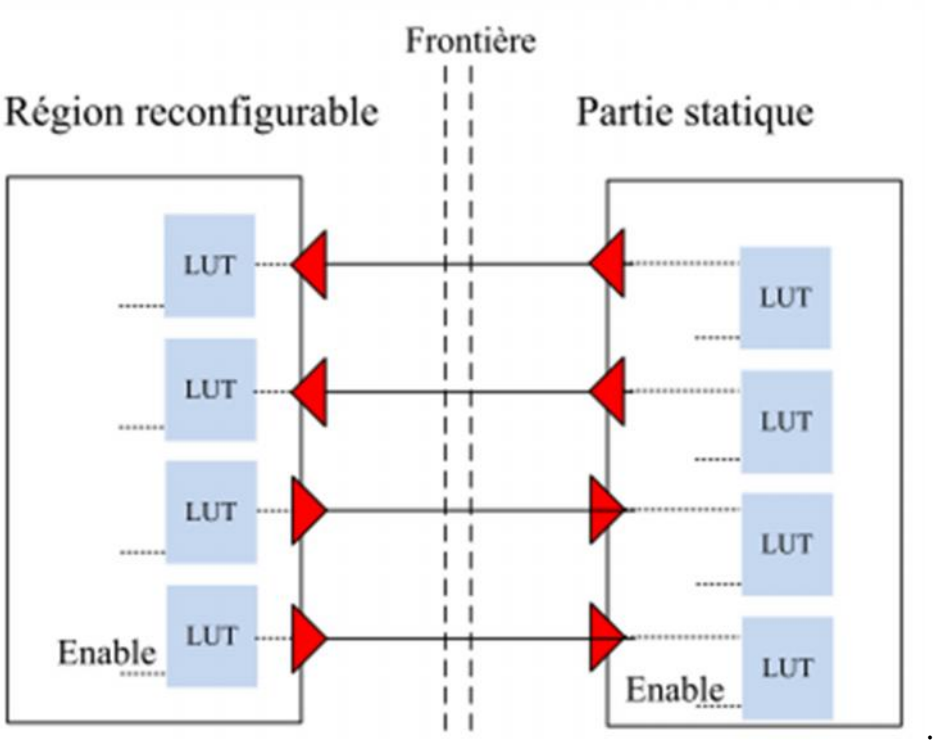


Figure 2.6: Architecture d'un Slice bus macro.

Cette liberté permet de sélectionner au mieux les ressources nécessaires. Des travaux menés dans ce cadre ont permis de valider cette nouvelle fonctionnalité du flot PBPR [38, 39]. Deuxièmement, les bus macro ont été remplacés par des pins partitions. Ces pins partitions sont nécessaires pour garantir la communication entre les parties statiques et les régions reconfigurables. Les pins partitions sont différents des bus macro, leur architecture est équivalente à une LUT sur un bit. Par rapport à l'architecture des slices bus macro l'occupation sur le FPGA a été divisée par deux, on est passé de deux LUT a une seule. Un autre avantage est que les pins partition sont automatiquement placés sur le FPGA, il n'est plus nécessaire de le faire manuellement. Comme le montre la figure 2.7, elles sont placées dans la région reconfigurable sur les slices disponibles. Une slice peut contenir plusieurs pins partitions d'entrées ou de sorties. Prenons l'exemple d'un Virtex 5, une slice peut contenir quatre pins partitions (un pin partition par LUT disponible). Une description de ce flot, qui a fait l'objet de nos travaux de recherche, sera donnée plus tard avec plus de détail dans ce chapitre.

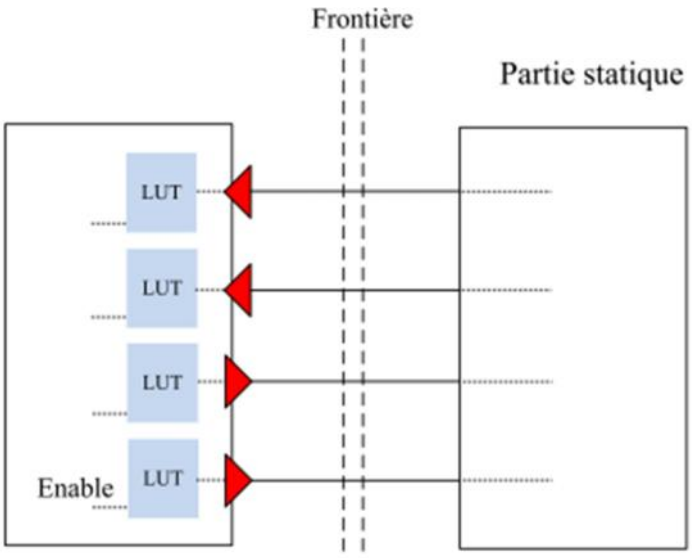


Figure 2.7 : Architecture d'un Pin Partition.

2.5.3 Comparatif sur les systèmes d'interconnexion des modules reconfigurables

Comme nous avons pu le voir précédemment, les systèmes d'interconnexion pour les modules reconfigurables ont évolué en même temps que les flots de conception (tableau 2.1). En 2002 les Bus Macro sont introduits, ils permettent de réaliser les premières communications avec les ports des modules reconfigurables. Huebner et coll. [36] propose en 2004 une évolution qui utilise des LUT et non des buffers à trois états. Cette évolution va permettre de simplifier le flot de conception et l'implémentation des interconnexions des modules reconfigurables. En 2009 Xilinx propose les pins partition qui simplifient grandement le flot de conception grâce au placement automatique des interconnexions, de plus les pins partition sont moins couteux en terme d'occupation sur le FPGA. Toutes ces interfaces de communication représentent la connexion entre le wrapper d'une IP statique et le wrapper d'une IP reconfigurable ou entre les deux wrappers de deux IP reconfigurables. Le wrapper correspond à l'interface de communication d'un IP core. Il est défini par les ports d'entrées et de sorties d'une IP. La région reconfigurable doit avoir un wrapper commun avec les IP qui seront placés sur celle-ci. Par conséquent, les wrappers des régions reconfigurables doivent être figés. Pour les différents flots présentés ci-dessus, les IP implémentées sur une même région reconfigurable doivent avoir un wrapper commun, cela signifie qu'elles auront le même nombre d'entrées et de sorties, des entrées et sorties de même taille et de même nom.

Tableau 2.1 : Comparatif des interfaces de communication.

Type	Bus Macro	Slice Bus Macro	Pin Partition
Date de création	Introduit en 2002	Proposé en 2004 [36]	Introduit par Xilinx en 2009
Architecture	2 buffers à 3 états par signal	2 LUT par signal	1 LUT par signal
Placement	Entre les modules	Dans les modules statiques et reconfigurables	Seulement dans les modules reconfigurables
Implémentation	Manuel et restreinte	Manuel	Automatique
Compatibilité FPGA	FPGA avec des buffers à trois états	Virtex 2-Pro, Virtex 4,5	Virtex 5, 6, 7
Flot de conception	MBPR / DBPR	EAPR	PBPR

2.6 Description détaillé du flot « Partition based partial reconfiguration »

Tel que discuté dans les sections précédentes, Xilinx a déployé un grand effort pour améliorer le concept de la reconfiguration partielle pour leur récents FPGA de familles Virtex 4, Virtex 5 et Virtex 6. En plus de l'avancement des technologies des FPGA, un grand travail de recherche a été investi en vue de simplifier le flot de conception des systèmes reconfigurables dynamiquement. Le nouveau flot de conception basé sur les partitions, est présenté sur la figure 2.8. Cette figure montre les phases nécessaires exigées pour accomplir la création d'une conception d'un système partiellement reconfigurable. Dans les sections suivantes, nous donnerons un aperçu sur la complexité de flot de conception associé et nous discuterons l'implication en terme de niveau d'expertise requis pour l'utilisation de la reconfiguration partielle.

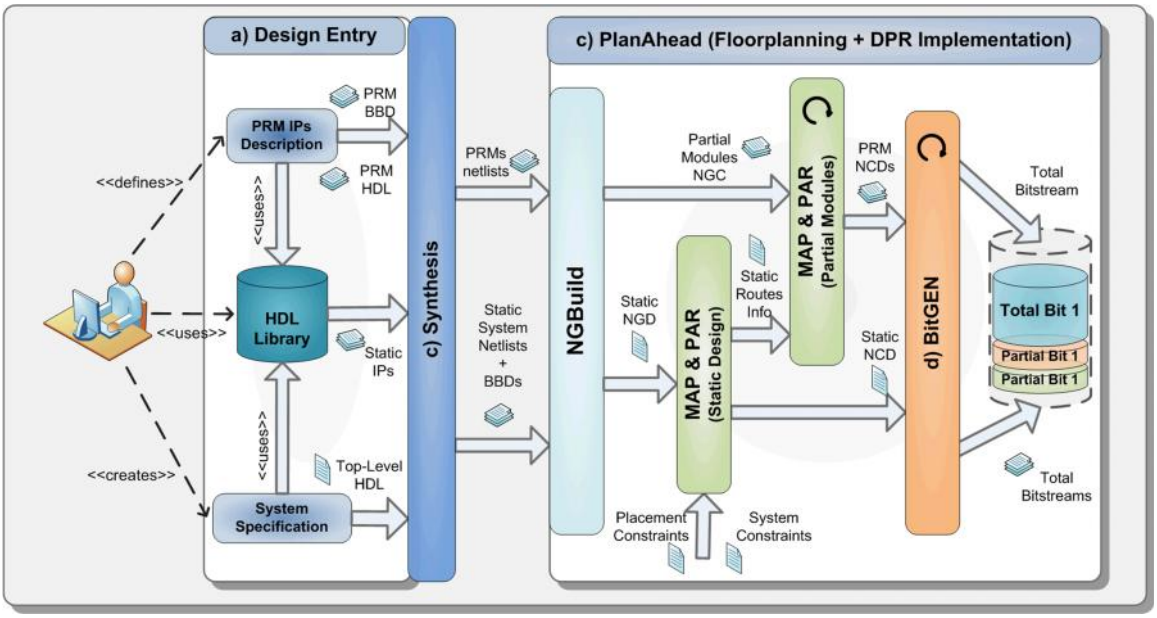


Figure 2.8: Représentation schématique du flot « Partition based partial reconfiguration »

2.6.1 Partitionnement de la conception (Design Partitioning)

Un concepteur de système commence par définir des spécifications de système sous forme de description haut niveau HDL. Le système est créé à partir d'une bibliothèque de cores IP contenant la description des différents modules du système. A cette phase, le concepteur d'une application DPR doit déjà connaître quels sont les modules qui sont susceptibles d'être dynamiquement reconfigurables. Ces modules sont connus en tant que modules partiellement Reconfigurable (PRM), et sont typiquement des netlists ou des descriptions HDL qui doivent être convertis en bitstreams partiels pour être employés par la suite pendant le processus de reconfiguration du système.

Le flot de conception DPR de Xilinx suppose que la technologie doive être en mesure d'échanger des configurations différentes dans une même partition, donc, plusieurs PRMs doivent être recueillis auprès de la bibliothèque des IP afin de définir une partition reconfigurable (RP) [37]. Un RP est un attribut d'instanciation qui définit l'instance en tant que partiellement reconfigurable. Cette instance sera ensuite observée par les étapes ultérieures de flot de conception comme étant une boîte noire dans laquelle les netlists des PRMs doivent être mappées. Les RP doivent être définies sur une section de la conception faisant partie de la logique statique tel que représenté dans la figure 2.9. Chaque partition doit être créée de manière à pouvoir accueillir les différentes

implémentations des PRMs associées. Encore, les RP doivent contenir une interface commune qui peut être utilisée par les différentes implémentations des PRMs. (côté droit de la figure 2.9).

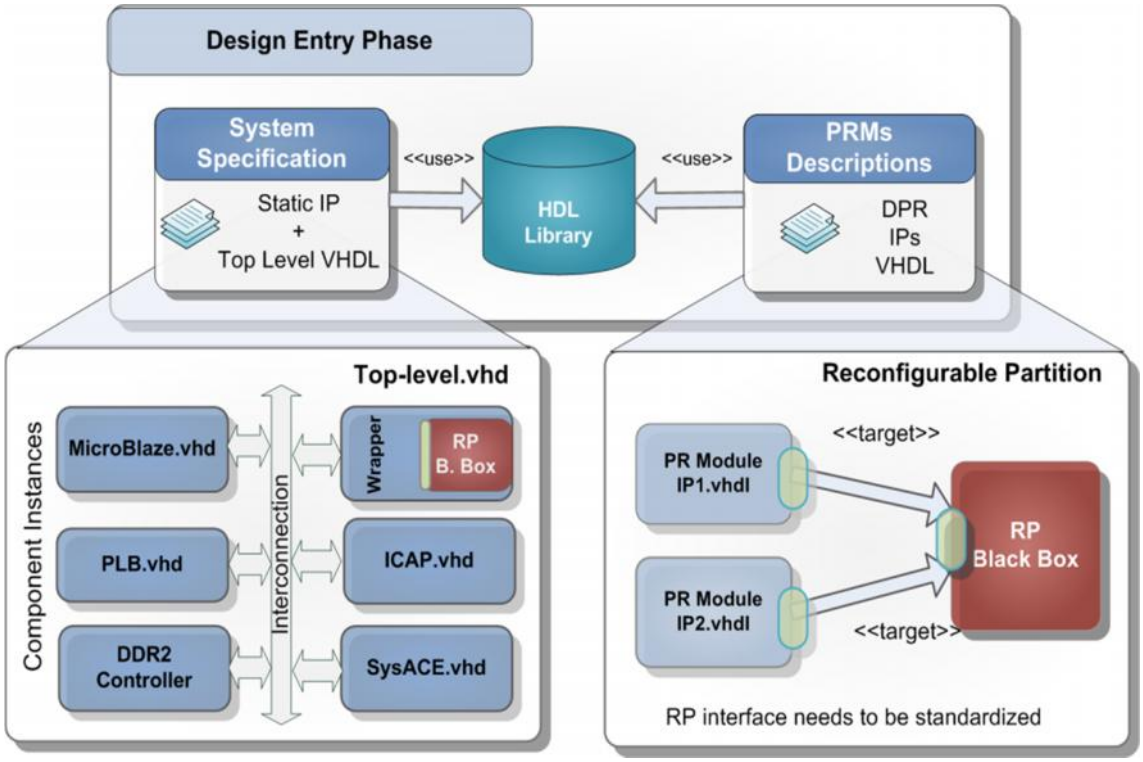


Figure 2.9: Phase de partitionnement de la conception

2.6.2 Synthèse

Pour ses derniers flots de conception DPR, Xilinx a décidé de réaliser l'implémentation du système à l'aide de l'outil PlanAhead [40]. Ce logiciel basé sur les principes d'une planification hiérarchique est destiné à la conception et l'analyse des circuits sur les FPGA de Xilinx. Il est déployé entre les phases de synthèse et de placement-routage et permet au concepteur d'analyser plus rapidement, de modifier, d'établir les contraintes et d'implémenter les conceptions. PlanAhead est un outil optionnel dans le flot de conception FPGA traditionnel, il complète la chaîne des outils ISE [41]. Il est généralement utilisé pour les conceptions nécessitant des hautes performances menant systématiquement à des solutions plus rapides et plus compactes. Il favorise également la réutilisation de la conception à travers la création de blocs de propriété intellectuelle réutilisables.

PlanAhead nous permet d'importer la hiérarchie de la conception logique sous forme de fichiers NGC ou EDIF, puis procéder à l'implémentation de la partie statique et les modules partiellement reconfigurables. Néanmoins, avant de passer à la phase d'implémentation du système, la description du système obtenu dans l'entrée de conception doit être transformée en netlists à travers une phase de synthèse, comme le montre la figure 2.8. En contraste avec le flot de conception traditionnel des FPGA où la spécification complète du système est synthétisée, la conception DPR nécessite de multiples phases de synthèse, une pour la partie statique et une pour chaque module reconfigurable dans la conception. L'utilitaire NgdBuild, dans l'étape suivante, fusionne les parties statiques et reconfigurables. La définition des partitions reconfigurables par la suite via l'outil PlanAhead désignera les interfaces entre les deux parties.

2.6.3 Planification (floorplanning)

Une fois toutes les netlists (statique et PRMs) ont été obtenues, elles seront importées par l'outil PlanAhead. Les étapes les plus importantes du flot de conception pour la reconfiguration partielle se déroulent dans cet outil. La première étape est la planification (ou floorplanning) des régions reconfigurables dans lesquelles des sections appropriées du FPGA doivent être réservées chacune pour implémenter différents PRMs. Ces derniers sont attribués à une partition RP particulière dans la phase de partitionnement, comme représenté sur le côté droit de la figure 2.10. Chaque région partiellement reconfigurable (PRR) doit être planifiée de telle sorte qu'elle puisse accueillir le plus grand PRM défini pour une partition reconfigurable. Cette action est connue sous le nom de budgétisation des ressources.

PlanAhead automatise le floorplanning de la conception et les différentes régions PRRs définies. La hiérarchie physique correspondante est alors créée et mappée dans un plan initial. PlanAhead utilise le concept de bloc physique (PBlock) comme unité de base de la hiérarchie physique. Une fois créés, les PBlocks peuvent être placés et dimensionnées sur une partie du plan du FPGA cible. Différentes estimations de la performance sont disponibles immédiatement (avant la phase de placement et routage) pour guider le concepteur à la redéfinition du plan et à l'amélioration récurrente de la conception.

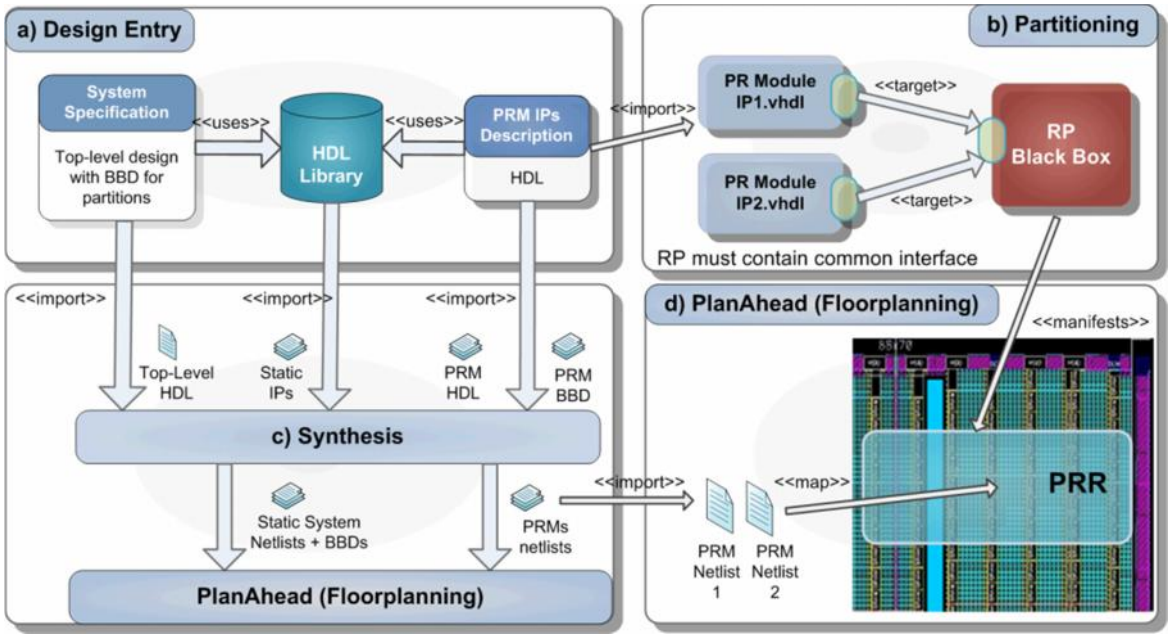


Figure 2.10 : Phase de planification de la conception

Les PRRs sont définis comme des PBlocks et les ressources incluses en termes de CLB, DSP Et BRAM peuvent être comparées avec les exigences de plus grand PRM associé afin de décider de la taille et les dimensions de la PRR optimale. Cette étape est très simple grâce à l'interface très intuitive de PlanAhead qui permet d'obtenir des informations précises sur l'utilisation des ressources pour l'exécution de la budgétisation de la logique d'une manière directe.

PlanAhead génère un fichier de contraintes (UCF) contenant les informations sur la zone réservée à chaque partition. Les composants formants la partie statique de la conception sont mappés en un seul PBlock. Chaque PRM est contraint à une section particulière dans le FPGA en utilisant les contraintes AREA_GROUP et RANGE dans le fichier UCF. AREA_GROUP est une contrainte de regroupement qui associe des éléments logiques de conception à une étiquette ou à un groupe particulier. Elle permet après la définition de partitions à délimiter la logique statique (non reconfigurable) de la logique reconfigurable, ce qui empêche de fusionner ces deux parties de la logique. La contrainte RANGE énumère les ressources (CLBs, BRAMs, blocs DSP) requises par chacun des modules reconfigurables PRM. Une fois le fichier UCF est défini, nous passons à l'étape suivante dans laquelle tous les composants statiques et reconfigurables de la plateforme seront implémentés dans le FPGA.

2.6.4 Implémentation

Les netlists importées par PlanAhead dans la phase précédente sont converties dans le format natif de Xilinx (NGD) en exécutant l'outil NgdBuild. Les netlists natives qui en résultent sont ensuite conduites, en compagnie de fichier des contraintes UCF, vers les outils MAP et PAR pour l'implémentation de la conception (figure 2.11).

A partir de la figure, on peut voir que quelques résultats de l'implémentation de la partie statique (des informations sur les routages statiques et les pins partitions) sont transmis à l'implémentation des modules PRMs qui vont être implémentés d'une façon indépendante. C'est une précaution nécessaire pour éviter d'introduire des conflits d'utilisation de ressources entre les bitstreams de configuration lorsqu'ils seront chargés successivement sur le FPGA à l'exécution. En fait, les routages de la partie statique peuvent passer complètement à travers un PRR. Ces informations sur les routages doivent être conservées afin d'éviter leurs utilisation par la suite au cours de la phase d'implémentation des PRMs.

Un exemple graphique est fourni sur les figures 2.11.c et 2.11.d. Il représente les actions MAP et PAR de la partie statique d'une conception DPR contenant deux régions reconfigurables. Comme on peut le constater sur la figure, la conception statique contient les dites partition Pins à la frontière entre la logique statique et la logique reconfigurable. Ces partitions pins sont nécessaires pour garantir que les connexions entre la logique statique et les différents PRM pour chaque RP restent identiques pour chaque configuration. Ils constituent aussi un moyen pratique pour créer des contraintes de temps sur les réseaux qui passent en provenance, ou à travers la frontière RP.

Les partitions pins sont insérées automatiquement pendant la phase d'implémentation de la logique statique et cela contrairement aux versions précédentes du flot de conception DPR, qui nécessitent d'utiliser et de placer manuellement des composants pré-routés appelés bus macros décrits précédemment. La Figure 2.11.d fait un zoom sur l'arrangement des partitions pins à base de LUTs dans des endroits fixes pour implémenter les différentes configurations des PRMs. Cela offre d'une manière efficace une interface statique aux différents modules reconfigurables.

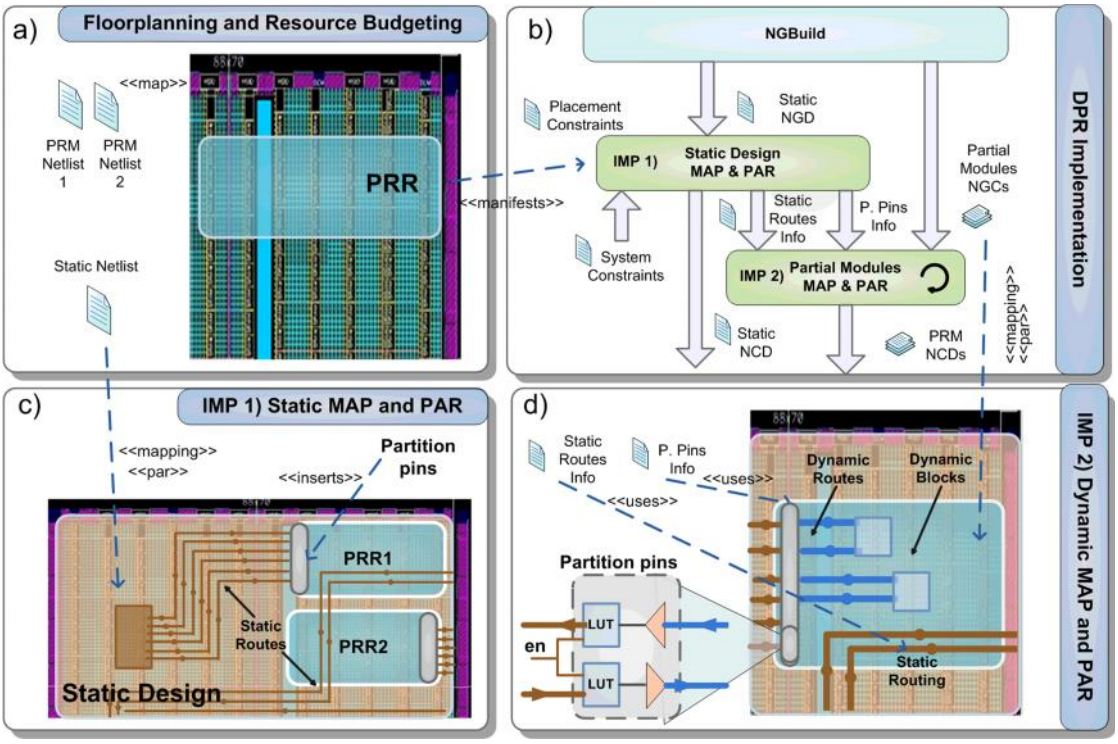


Figure 2.11: Phases d'implémentation de la conception

Le flot de conception DPR permet le concept de la reconfiguration dynamique. Cela implique que chaque combinaison des PRMs dans la conception constitue une configuration différente. A titre d'exemple, pour deux PRRs si chacune peut contenir deux PRMs différentes, il y en a au moins quatre configurations différentes, et deux de plus si un ou les deux PRR ne sont pas configurés (un bitstream vide pour chaque PRR pour réduire la consommation). Avant de procéder à l'implémentation, l'utilisateur doit définir explicitement toutes les PRMs qui doivent être mappées dans une PRR donnée tel qu'illustré dans la figure 2.11 a.

Les outils MAP et PAR implémentent alors la conception statique conjointement avec une combinaison de modules PRMs choisi par l'utilisateur (selon les besoins de l'application projetée). Comme mentionné dans le paragraphe précédent, les outils d'implémentation insèrent les partitions pins dans l'interface entre la logique statique et les partitions RP. Ces informations sont stockées dans des fichiers intermédiaires et utilisées pour les passages suivants lorsque les outils implémentent les netlists des PRMs restants. Ces outils prennent les partitions pins en tant que points de référence et masquent les routages statiques existants dans la partition pour éviter les conflits pendant le processus de reconfiguration.

2.6.5 Génération de Bitstream

Tout comme la phase d'implémentation, où chaque configuration (une combinaison de netlists statiques et celles de PRMs) doit subir la phase d'implémentation séparément, la phase de génération de bitstream suit un chemin semblable: un bitstream statique et des bitstreams partiels sont créés pour chaque passage. La figure 2.12.a montre deux configurations (deux paires de PRMs à être mappées sur deux PRRs). La phase d'implémentation du flot DPR, tel qu'illustré dans figure 2.12.b, a produit à la fois un ensemble de fichiers NCD routés pour les modules statiques et ceux partiellement reconfigurables. Ces fichiers sont utilisés pour produire la première configuration composée d'un bitstream total et deux bitstreams partiels à l'aide de l'outil Bitgen de Xilinx comme le montre la première partie de la figure 2.12.c. Les contraintes de la partie statique de la conception sont ensuite copiées pour être prises en considération dans le deuxième passage dans lequel la deuxième configuration se produit, cette fois en utilisant les deux autres modules reconfigurables.

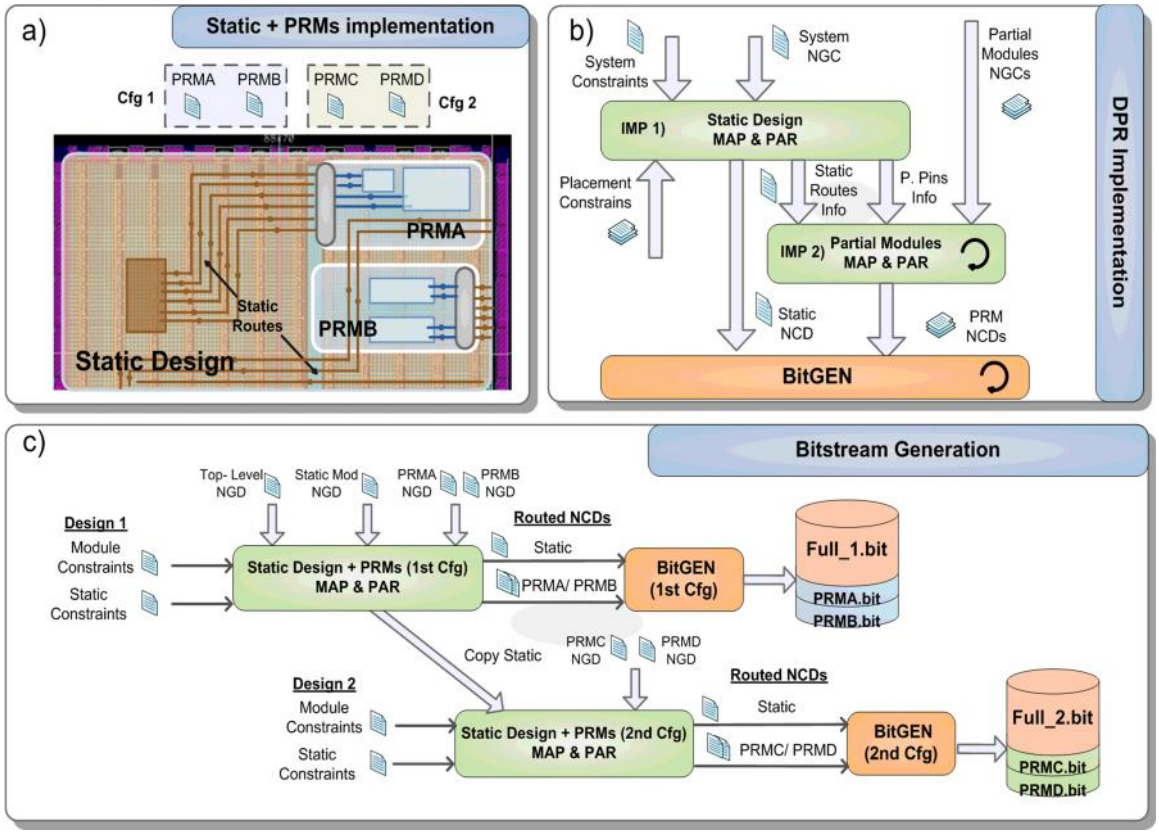


Figure 2.12: Phase de génération de bitstreams

Outre les bitstreams générés dans ces deux itérations, le processus peut aussi générer des bitstreams vierges pour chacun des PRR (non représentés sur la figure). Ces bitstreams vides servent à supprimer le contenu des PRR associées quand aucun module reconfigurable n'est nécessaire à charger.

La raison derrière la génération de plusieurs configurations totales est de permettre au concepteur de choisir la configuration initiale du FPGA entier à partir de laquelle le composant commence son exécution après sa mise sous tension. Par la suite, les bitstreams partiels pour chaque PRR vont s'échanger entre eux selon les besoins de l'application.

2.6.6 Gestion de la reconfiguration

Dans les sections précédentes, nous avons décrit comment spécifier un système pour produire un ensemble de bitstreams qui seront utilisés pour ajouter la flexibilité au système à base de FPGA à travers le processus de reconfiguration partielle. La reconfiguration peut être réalisée à la demande en utilisant des commandes à travers un agent externe ou via une connexion à distance [42]. Toutefois, la plupart des systèmes nécessitent d'être contrôlées de façon autonome, ce qui signifie que la gestion des bitstreams partiels doit être faite par un processeur [43], soit à l'extérieur à ou l'intérieur du tissu du FPGA en utilisant un processeur hard comme le PowerPC ou le plus récent, le Zynq, ou bien un processeur synthétisé comme le MicroBlaze.

L'utilisation d'un processeur facilite le processus de reconfiguration partielle, étant donné que l'application exécutée sur un FPGA autour d'un processeur utilisant un contrôleur logiciel peut être facilement adapté si des changements sont nécessaires. Cela est contrairement le cas avec un contrôleur matériel réalisé en VHDL qui exigerait beaucoup d'effort pour sa description et son adaptation [44]. De nombreux travaux ont été proposés dans la littérature pour optimiser le processus de reconfiguration (en terme de temps de reconfiguration par exemple) ou pour fournir de nouveaux services à un système d'exploitation porté par le processeur pour gérer des plateformes reconfigurables dynamiquement [45, 46]. Cet aspect sera discuté en profondeur dans le chapitre suivant. Néanmoins, la discussion suivante met l'accent sur la complexité ajoutée par le processus d'implémentation des systèmes DPR qui peut être liée principalement à l'utilisation des outils, le développement des applications ou l'exigence de nouvelles expertises.

Le développement de plateformes à base de FPGA de Xilinx autour d'un processeur a été largement simplifié en utilisant le logiciel EDK de Xilinx (Embedded Development Kit) [47], composé principalement de l'outil XPS (Xilinx Platform Studio) et l'outil SDK (Software development kit) [20]. Le premier outil est utilisé pour la création de la plateforme matérielle, tandis que le second est utilisé pour l'élaboration du code pour le processeur cible. Avec EDK, tel que présenté dans la figure 2.13, la description du système peut être obtenue à partir de XPS qui permet d'accélérer le développement de l'application qu'elle soit reconfigurable dynamiquement ou non. Cette description sera amenée vers Xilinx SDK dans lequel le développement de la partie logiciel peut commencer avant même que tous les détails de la plateforme matérielle soient finalisés.

Le développent de la partie logiciel commence par la création d'une application en utilisant le code C et un ensemble de pilotes pour communiquer avec les IPs matérielles dont la majorité sont prédéfinies dans une bibliothèque de fonctions. Le code développé et les pilotes seront compilés pour créer un fichier exécutable.

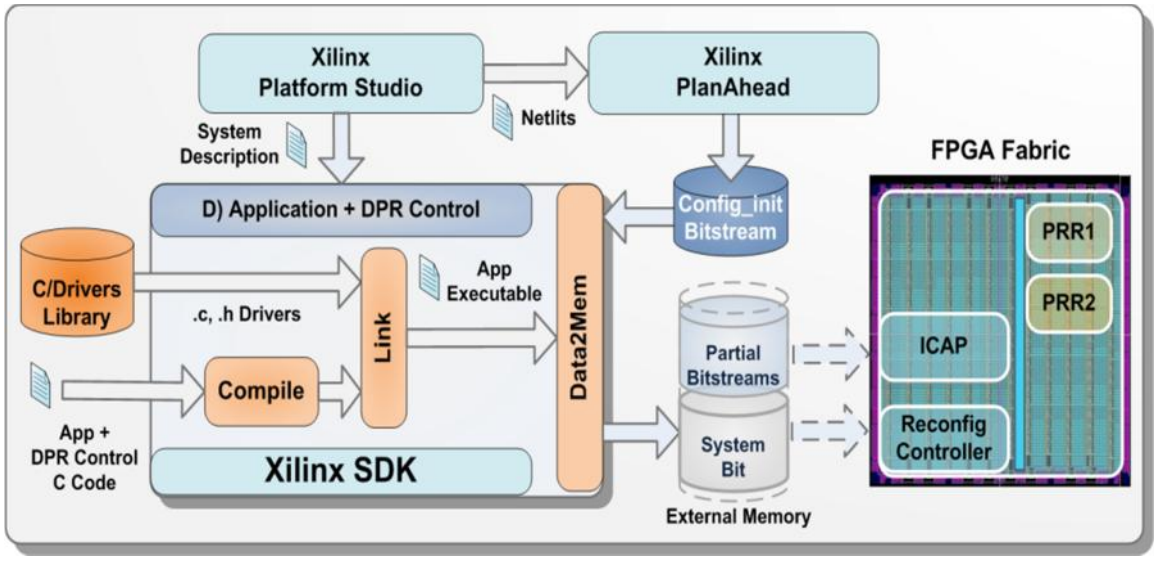


Figure 2.13 : Génération de l'application de contrôle du système DPR via les outils Xilinx

Pour un système reconfigurable dynamiquement DPR, une fois que la description du système a été créée en utilisant l'outil Xilinx PlanAhead, le bitstream de configuration généré peut alors être combiné avec le bitstream de l'application à l'aide de l'outil

Data2Mem en chargeant le fichier exécutable dans la mémoire de programme associé au processeur choisi. Ce processus permet de générer un fichier bitstream complet servant à configurer le FPGA entier à sa mise sous tension. Ce bitstream peut être stocké dans une mémoire non-volatile conjointement avec les bitstreams partiels qui seront chargés à la demande dans les régions reconfigurables associés en envoyant leurs informations de configurations vers le port ICAP [48]. Cette opération est commandée par le processeur embarqué et doit être prise en compte lors de développement de l'application.

2.7 Limitations du flot de conception pour la reconfiguration dynamique partielle

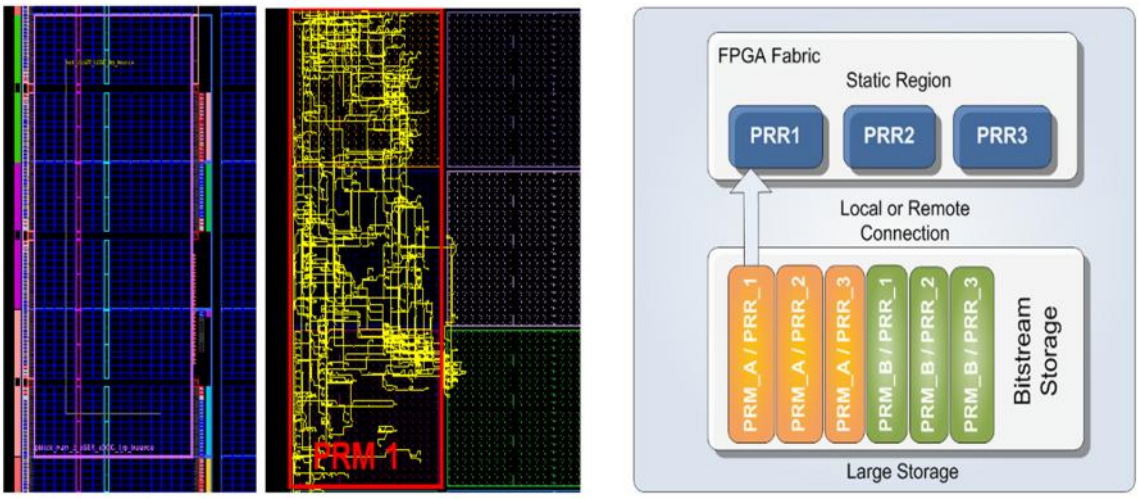
Comme mentionné dans les sections précédentes, la reconfiguration dynamique partielle a plusieurs avantages. En plus de l'augmentation de la flexibilité du système, l'utilisation des ressources est significativement réduite ce qui diminue ainsi la complexité du système à intégrer dans le FPGA et offre la possibilité d'utiliser un plus petit FPGA pour implémenter les mêmes fonctionnalités de l'application. Tout cela a un impact positif sur la consommation d'énergie du système.

Ces dernières années, un grand effort de recherche est en train d'être déployé pour étendre les capacités des SoC reconfigurables et formaliser l'utilisation de la reconfiguration partielle [7, 8]. L'idée de base est que des régions spécifiques dans le FPGA peuvent être utilisées pour placer des tâches matérielles à la demande permettant de promouvoir d'une façon efficace la notion de virtualisation des tâches matérielles. Ces dernières sont stockées sous forme de bitstreams précompilés dans une mémoire externe où ils peuvent être chargés par le processeur à la demande dans des régions où on veut remplacer leurs missions. D'autres applications pourraient avoir besoin de plusieurs instances du même module tel que les systèmes qui supportent la migration chaude [9] et les systèmes à tolérance de panne [10]. Dans le premier cas, plusieurs copies de la même fonction peuvent être potentiellement nécessaires à n'importe quel moment donné. Dans le deuxième cas, l'emplacement du même module nécessite d'être changé en fonction des conditions physiques du système.

Le flot de conception pour la reconfiguration dynamique partielle limite la reconfiguration dynamique à des zones dédiées que nous avons appelées régions partiellement reconfigurables (PRR). A la fin de flot de conception, les fichiers de configuration totale et partiels sont générés. Chaque configuration partielle cible une PRR

spécifique et permet de modifier sa fonction pendant l'exécution sans interrompre le fonctionnement général du système.

Les bitstreams partiels permettent des temps de reconfigurations plus rapides et des tailles des bitstreams plus petites par rapport aux bitstreams complets traditionnels (le bitstream nécessaire pour reconfigurer le composant FPGA entier) puisque les bitstreams partiels ne contiennent que des données de configuration spécifiques à la PRR cible. Prenant l'exemple illustré dans la figure 2.14.a, le module PMR1 doit être placé dans la région PRR1. Cette situation implique que pour chaque région PRR on doit avoir pour chaque module reconfigurable PRM associé un bitstream partiel unique. Un accroissement de l'utilisation de la mémoire aura lieu donc dans les applications où la même fonction (PRM) a besoin d'être placé dans plusieurs PRRs. Dans de tels scénarios, plusieurs bitstreams partiels quasi identiques (la même fonctionnalité avec emplacement différente) doivent être stockés et transférés au gestionnaire de reconfiguration. La figure 2.14..b illustre une section d'un système PR contenant deux PRM et trois bitstreams partiels par PRM (six bitstreams partiels au total). Cette redondance est nécessaire pour le placement dynamique des PRM mais malheureusement elle nécessite encore plus de capacités de stockage et des moyens de communication qui engendrent un surcoût temporel significatif en matière d'accès mémoire et de cycles processeur.



(a) (b)

Figure 2.14 : (a) Exemple d'un PRM placé dans un PRR.
(b) Un système DPR sans relocation des bitstreams

En vue de trouver des solutions à cette limitation et d'améliorer les performances du processus de la reconfiguration dynamique partielle, plusieurs approches ont été proposées et se présentent, dans leur majorité, comme des extensions du flot de conception de la reconfiguration dynamique. Une de ces approches est la relocation des bitstreams partiels. Le processus de relocation cherche à étendre la plaçabilité d'un bitstream en effectuant des modifications sur son contenu

Les modifications apportées affectent principalement les informations liées aux adresses de configuration pour changer son emplacement. Elles consistent à déplacer les coordonnés de la partition prévue vers d'autres régions dans l'espace reconfigurable du FPGA. Plusieurs approches de relocation ont été proposées dans la littérature réalisant tous une manipulation en ligne du bitstream. Les modules chargés de relocation sont implémentées soit en matériel soit en logiciel. Les deux variantes sont généralement efficaces mais présentent les deux des inconvénients. En premier lieu, les approches logicielles exécutées par le processeur embarqué introduisent un temps de manipulation qui peut être considéré excessif dans les applications critiques ou en temps réel. En second lieu, les approches de relocation implémentées en matériel sont plus rapides sur le plan exécution, mais elles ont besoin de plusieurs ressources logiques coûteuses pour effectuer toutes les modifications nécessaires du bitstream.

2.8 Conclusion

Ce chapitre a été consacré à la présentation du concept de la reconfiguration dynamique partielle et les flots de conception associés introduits par Xilinx pour ses circuits FPGA. Ces flots permettent de développer des applications avec des tâches matérielles qui peuvent être allouées et désallouées dans le FPGA sans interrompre le système.

Comme nous l'avons mentionné, Ces dernières années, un intérêt particulier à surmonter le problème de plaçabilité des tâches dans l'espace du FPGA a conduit à la notion de relocation de bitstream partiel. Le principe est d'effectuer des manipulations sur un seul bitstream pour le faire correspondre aux différentes partitions ce qui permet de réduire considérablement le besoin en mémoire de stockage pour les applications qui nécessitent d'avoir plusieurs instances d'un même module. Toutefois, l'implémentation des approches déjà proposées trouve des difficultés accrues. Elles consomment

généralement assez de ressources quand elles sont implémentées en matériel et introduisent un surcoût en temps d'exécution lorsqu'elles sont implémentées en logiciel. Notre contribution à travers cette thèse vise le développement d'une nouvelle méthodologie de relocation qui permet d'accélérer considérablement le processus de relocation et cela sans introduire aucune incidence sur les ressources du système. Dans les deux chapitres suivants, nous allons exposer, en premier lieu un état de l'art dans le domaine de relocation et les solutions proposées dans la littérature. Puis nous allons décrire en détail notre méthodologie de relocation et son interaction avec le nouveau flot de conception pour la reconfiguration dynamique partielle basé sur le principe des partitions que nous avons décrit précédemment.

Chapitre 3

TECHNIQUE DE RELOCATION DES BITSTREAMS PARTIELS

CHAPITRE 3

TECHNIQUE DE RELOCATION DES BITSTREAMS PARTIELS

3.1 Introduction

Comme nous avons mentionné dans le chapitre précédent, la reconfiguration dynamique a été introduite ces dernières années comme un moyen permettant de virtualiser les tâches matérielles dans les systèmes à base de FPGA [2]. Cependant, ce n'était qu'avec l'introduction de la technologie de reconfiguration dynamique partielle (RDP) par Xilinx que ces systèmes sont devenus une réalité. Dans les systèmes RDP, les modules reconfigurables dynamiquement peuvent être implémentés comme des modules matériels indépendants qui peuvent être chargés dans le composant en cas de besoin en remplaçant d'autres fonctionnalités. Toutefois, Les flots de conception de la reconfiguration dynamique partielle que nous avons décrits dans le chapitre précédent limitent l'emplacement des modules partiellement reconfigurables (PRM) dans des régions reconfigurables prédéfinies dans lesquels ces PRM seront explicitement placées. La communauté des chercheurs et les créateurs de systèmes sur puce ont trouvé cette méthode trop restrictive pour la mise en œuvre des systèmes complexes avec plusieurs fonctionnalités; les applications doivent être donc soigneusement planifiées au préalable ce qui aboutit à des systèmes avec une flexibilité limitée.

En vue de trouver des solutions à ces limitations et d'augmenter les capacités du processus de la reconfiguration dynamique partielle, plusieurs approches ont été proposées et se présentent dans leur majorité comme des extensions du flot traditionnel de conception de la reconfiguration dynamique. Une de ces approches est la relocation des bitstreams partiels. Le processus de relocation cherche à étendre la plaçabilité d'un bitstream en effectuant des modifications sur son contenu. Ces modifications affectent principalement les informations liées aux adresses de configuration et consiste à déplacer les coordonnés de la partition prévue vers d'autres régions dans l'espace reconfigurable du FPGA. La relocation peut être bien utilisée dans les applications qui nécessitent une implémentation dite en échelle de plusieurs cœurs semblables pour effectuer une sorte de traitement intensif pour améliorer la performance et réduire le temps de chargement [49, 50].

Le présent chapitre est organisé comme suit : en premier lieu, nous allons passer en revue les différents services qui peuvent être offerts à une architecture reconfigurable dynamiquement pour améliorer sa flexibilité. Ces services peuvent être gérés sous l'égide d'un système d'exploitation porté sur un processeur embarqué dans le système. Par la suite, nous discuterons le service de relocation en se focalisant tout d'abord sur les aspects technologiques et architecturaux des FPGA qui limitent l'application de la relocation dans le flot de conception de la reconfiguration partielle. Ensuite nous présenterons les principales approches de relocation proposées dans la littérature et leurs limites en termes de ressources déployées par le relocateur, du temps nécessaire pour effectuer la relocation, de flot de conception DPR déployé et d'autres contraintes. Ensuite, nous introduirons brièvement notre nouvelle méthodologie de relocation et les améliorations apportées pour lever ces contraintes. Nous conclurons ce chapitre par la description des défis posés par le nouveau flot de conception de la reconfiguration partielle basé sur le principe des partitions, et comment il limite l'applicabilité des approches de relocation précédents. La description détaillée de notre méthodologie de relocation va faire l'objet du quatrième chapitre.

3.2 Services offerts pour les systèmes reconfigurables dynamiquement

De nombreuses applications industrielles ont déjà trouvé le jour en exploitant le concept de la reconfiguration partielle dans plusieurs domaines tels que le radio logiciel [4], la biométrie [5], l'industrie automobile [6] et bien d'autres. Ces applications sont généralement implémentées sur des architectures reconfigurables de type SoC, ce qui leur permet d'exploiter les avantages de la flexibilité du matériel reconfigurable et la programmabilité du processeur qui contrôle le SoC. Les FPGA actuels peuvent contenir maintenant un ou plusieurs processeurs directement incorporés dans le silicium où encore des processeurs logiciels qui peuvent être synthétisés et ajoutés dans la plateforme. Il devient ainsi tout à fait envisageable de mettre en œuvre des architectures SoC hybrides dans lesquelles certaines tâches seraient implémentées matériellement et gérées suivant le modèle des tâches logicielles.

Similairement aux tâches logicielles, les tâches matérielles ont aussi besoin d'être gérées, ordonnancées, préemptées et placées dans la logique reconfigurable. Ces tâches ne sont plus considérées comme périphériques esclaves attachées à un bus et contrôlées par

des pilotes logicielles. Elles sont maintenant plus actives et peuvent demander, tout comme les tâches logicielles, des primitives de synchronisation, l'accès au bus et à la mémoire et peuvent être dynamiquement crées et terminées. Tous ces comportements des tâches matérielles dans un système reconfigurable doivent être gérés par un système d'exploitation (OS). En conséquence, un OS destiné aux systèmes reconfigurables doit être en mesure de fournir des services supplémentaires pour gérer l'ensemble des traitements dans la plateforme reconfigurable. Ces services qui peuvent être supportés par l'OS sont résumés dans la figure 3.1.

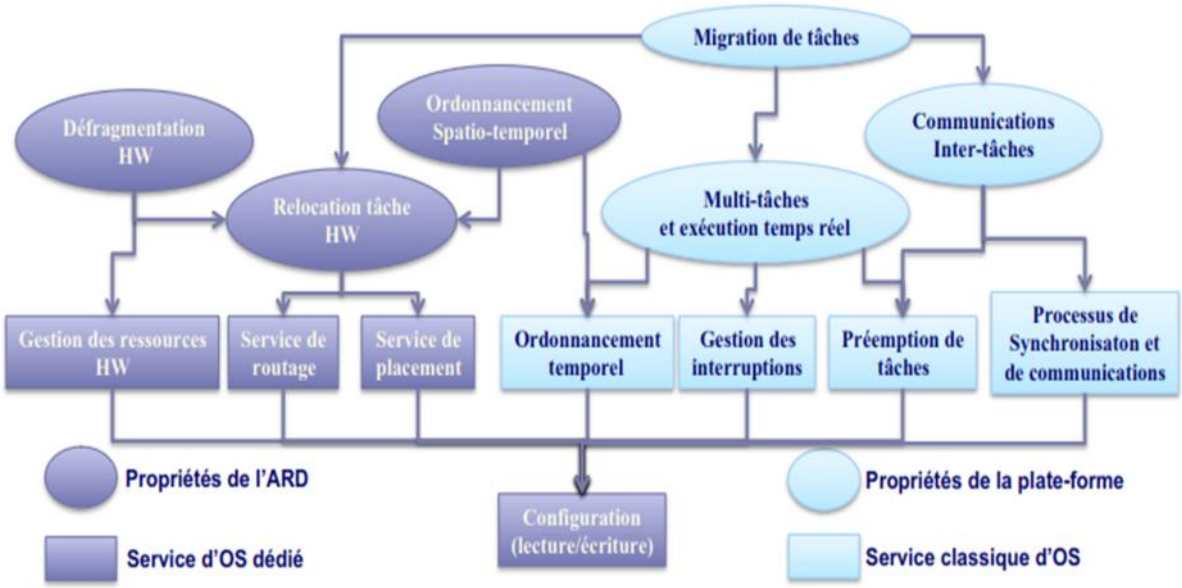


Figure 3.1 : Services d'un OS supportant la gestion d'une Architecture reconfigurable dynamiquement

En plus des propriétés classiques que l'on attend, un OS gérant une architecture reconfigurable nécessite la gestion des tâches matérielles d'une manière dynamique. L'OS peut être amené à gérer plusieurs instances d'une même tâche. En effet, certaines tâches peuvent être instanciées en tâches logicielles et/ou matérielles. Le choix du type pour l'exécution d'un traitement est réalisé dynamiquement par l'OS.

Quelle que soit la stratégie choisie, lors de l'instanciation d'une tâche matérielle, il faudra alors décider d'un placement de la tâche dans la matrice du FPGA. Afin de répondre

aux besoins de performance et d'optimisation, ce dernier doit être réalisé de manière dynamique et flexible. L'ordonnancement spatio-temporel, doit alors gérer l'exécution des tâches non seulement dans le domaine temporel, mais aussi dans le domaine spatial. Ce nouveau type d'ordonnanceurs doit alors gérer les différents types de tâches en fonction de contraintes de temps réel, mais aussi de disponibilité de ressources. Ce service de placement peut aussi offrir un service plus évolué si l'on souhaite assurer la répartition optimale des charges de calcul au sein du SoC reconfigurable. Pour remplir cet objectif, les tâches ainsi que leurs supports d'exécution doivent supporter une éventuelle préemption par l'OS.

La gestion des ressources est aussi un point clé pour la gestion d'une architecture reconfigurable. Une tâche matérielle doit être implémentée de manière la plus efficace possible, donc en utilisant les ressources adéquates. La notion de ressources dans le modèle de SoC reconfigurable est une notion dynamique car elle évolue au fil des reconfigurations effectuées par l'OS. Cette propriété a un impact sur le placement des tâches. L'allocation dynamique de tâches au sein de l'architecture peut entrainer une fragmentation des ressources (de manière similaire à une mémoire dans laquelle une succession d'allocations et de désallocations sont effectuées). La propriété de défragmentation permet alors de compacter les tâches au sein de l'architecture afin de récupérer la forme rectangulaire la plus grande possible.

Un service plus avancée de l'OS pourrait être aussi la relocation qui permet à une tâche matérielle déjà placée et configurée d'être relogée dans une autre région de la logique reconfigurable en utilisant d'autres ressources. Les services précédents s'appuient tous sur la propriété de relocation de tâches, qui permet de déplacer une tâche matérielle au sein de l'architecture reconfigurable. Cette propriété est nécessaire pour l'implémentation de la migration de tâche de matériel vers matériel. Après relocation, Cette tâche va continuer à remplir la même fonction de l'endroit où il a été préempté.

Les propriétés offertes par la reconfiguration dynamique nécessitent donc de revisiter certains services d'OS (figure 3.1). Des services spécifiques comme la gestion des ressources matérielles, la gestion de la reconfiguration et les services de relocation et de placement sont nécessaires pour l'implémentation des propriétés précédentes. Cependant, d'autres services nécessitent d'être adaptés afin de prendre en compte les spécificités d'un SoC reconfigurable (communication, ordonnancement, préemption et migration de

tâches,…). Cette liste non exhaustive est, à notre sens, minimale. La conception d'une plateforme reconfigurable dynamiquement incluant un OS est alors une tâche complexe nécessitant la mise en œuvre d'une méthodologie adaptée.

3.3 Le Principe de Relocation

Nous avons indiqué que dans le flot de conception traditionnel de la reconfiguration partielle de Xilinx, les bitstreams partiels obtenus pour implémenter une fonctionnalité d'un module PRM sont associés à une région reconfigurable (PRR) fixe respectant leurs frontières physiques. Par conséquent, le bitstream ne peut pas être utilisé directement pour configurer un autre PRR dans un autre endroit du FPGA même si les ressources sous-jacentes et la géométrie des PRR sont les mêmes.

Ces dernières années, un grand intérêt a été montré par la communié scientifique pour surmonter ce problème. Cela a conduit à l'élaboration d'une série de techniques et d'approches qui peuvent être communément nommées Relocation des Bitstream partiels (PBR). Le principe de base consiste à effectuer la manipulation et la modification d'un bitstream afin de pouvoir le placer dans un PRR différent. Les bitstreams partiels tels que décrits contiennent des informations sur leur positions dans la structure du FPGA. En manipulant ces informations, un PRM peut être mis dans un PRR différent si et seulement si certaines conditions sont remplies. On doit avoir tout d'abord la même distribution des ressources dans les PRR source et destination et on doit éviter également que le routage statique soit en conflit avec les ressources de routage dans les modules reconfigurables (PRM). La figure 3.2 représente le principe de relocation et les différents types qui ont été proposés.

Le système reconfigurable de la figure 3.2 contient trois régions PRRs., deux modules PRMs sont sélectionnés pour partager chacune des trois régions. Pour cela on a besoin d'un bitstream pour chaque module. Le relocateur, le module résidant dans la partie statique du FPGA chargé de la relocation du bitstream, reçoit l'information du bitstream et procède à sa modification selon les informations liées à la PRR de destination.

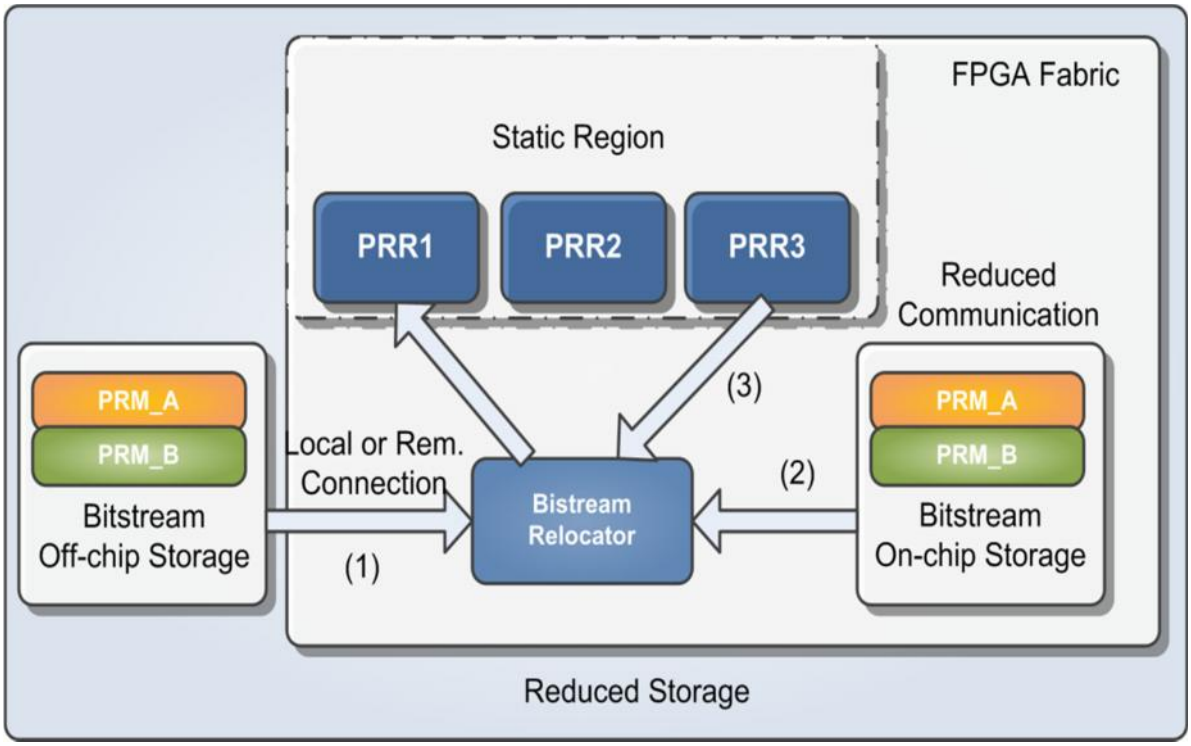


Figure 3.2 : Différents scénarios de Relocation des Bitstreams.

Dans la relocation des bitstreams nous nous sommes intéressés à l'optimisation de la mémoire de stockage, il y en a plusieurs scénarios pour atteindre cet objectif. Traditionnellement, les bitstreams sont stockés dans une mémoire externe où même à l'extérieur du système (scénario 1 dans la figure), cette approche est la plus envisageable mais elle peut compliquer l'accès au bitstream si les ressources du FPGA utilisées sont limitées (ressources nécessaires pour implémenter le contrôleur mémoire). La seconde approche consiste à stocker le bitstream dans la mémoire locale BRAM du FPGA (scénario 2). Cette approche est justifiée seulement quand quelques modules ont besoin d'être reconfigurés. Finalement, dans quelques systèmes, une technique appelé Relecture où Readback [51] peut être déployée pour extraire un bitstream directement à partir d'une PRR active et le reloger directement dans une autre PRR (scénario 3).

Les travaux de recherche menés jusqu'à présent dans le domaine de relocation des bitstreams sont concentrés toujours autour de ces principes illustrés dans la figure 3.2. La majorité d'entre eux réalisent la relocation en considérant une distribution homogène des ressources logiques dans le FPGA. Les différentes approches de la relocation existants peuvent être classées selon les critères suivants :

(a) Endroit du processeur qui manipule le bitstream: sur puce ou hors-puce **:**

Les approches de relocation se différent dans le sens où quelques approches d'entre eux réalisent la relocation en hors-puce (de l'extérieur du FPGA). Evidemment. Cet aspect évite l'utilisation d'une mémoire sur carte pour stocker les bitstreams, mais il complique le protocole de communication car il nécessite une connexion à distance augmentant ainsi la surcharge de configuration. D'autres approches embarquent le processeur de relocation (relocateur) dans le composant FPGA. Dans ce cas, différents moyens de stockage des bitstreams peuvent être utilisés.

(b) Type du processeur sur puce : matériel ou logiciel **:**

Le relocateur peut être implémenté soit en matériel soit en logiciel. La première approche implique la création d'un accélérateur matériel spécialisé qui interagit directement avec le bitstream et le port de configuration ICAP. La seconde approche utilise un processeur embarqué (i.e. MicroBlaze) pour retrouver les bitstreams partiels et les modifier. Les avantages de l'accélérateur matériel sont multiples : il libère le processeur des tâches répétitives consommant du temps et accélère le processus de relocation à l'inverse d'une solution logicielle qui peut créer un étouffement entre la mémoire externe et le port ICAP.

(c) Stockage des bitstreams dans une RAM sur puce ou dans une mémoire Flash hors puce**:**

Si le processus de relocation doit être réalisé sur puce, le bitstream peut être stocké dans une mémoire externe ou bien dans une mémoire interne de type BRAM. Egalement, il peut être lu à partir d'une autre région PRR à travers le processus appelé Readback. La décision dépend fortement de l'application mais la majorité des approches utilisent une mémoire hors puce puisque, pour les applications autonomes, un support de stockage non volatile doit être utilisé.

(d) Le flot de conception déployé pour la reconfiguration partielle**:**

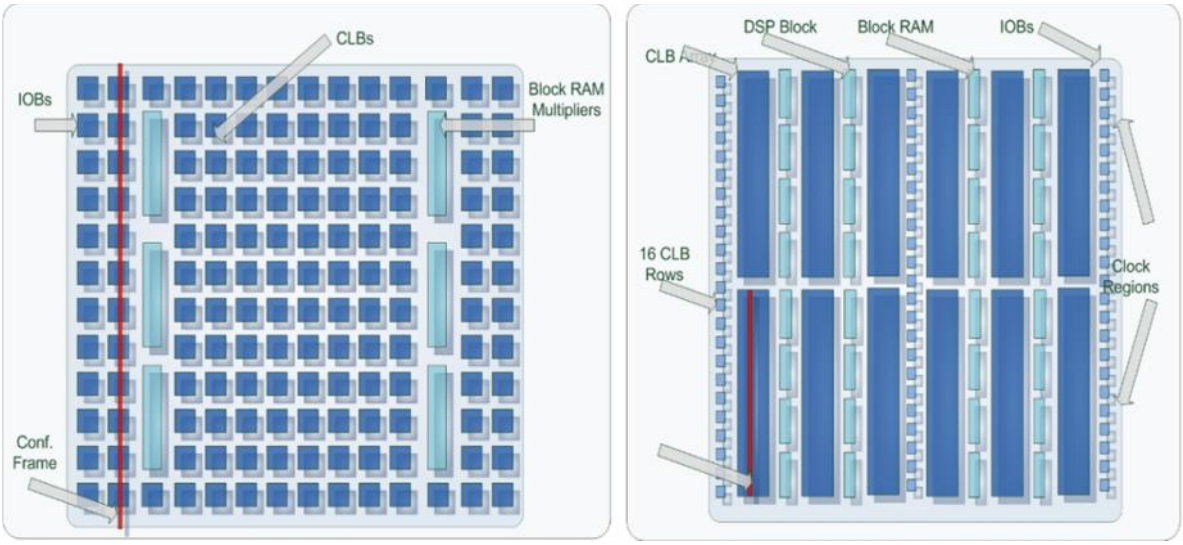
Le flot de conception choisi pour construire le système reconfigurable est d'une grande importance puisque chaque version a ses contraintes particulières permettant à l'utilisateur de limiter les routages statiques dans les modules reconfigurables. L'utilisation

de la contrainte de routage n'existe pas dans les nouvelles versions du flot, chose qui a engendré plus de complications dans les méthodologies de relocation.

3.4 Description de la structure de configuration d'un FPGA

Dans cette section, nous offrons une brève description de l'architecture d'un FPGA et sa configuration, informations nécessaires pour comprendre le concept de relocation par manipulation des bitstreams. Les familles Virtex 4, Virtex 5 et Virtex 6 sont les plus récentes générations des FPGA de Xilinx contenants une large variété d'éléments. Les blocs de base ont vu une amélioration par rapport à ceux qui existaient dans les anciens produits Virtex (Virtex, Virtex-II et Virtex-II Pro) mais ils assurent toujours une compatibilité avec les conceptions existantes. Les principaux types de blocs présents dans les différentes familles sont : CLB, DSP, BRAM, IOB et CLOCK.

Tous ces éléments programmables à l'intérieur des composants Virtex sont contrôlés par des cellules mémoire volatiles qui doivent être configurées sous tension. Ces cellules mémoire sont appelées communément mémoire de configuration. Les architectures des Virtex 4, Virtex 5 et Virtex 6 ont une mémoire de configuration arrangée en bancs (frames) qui sont quadrillés à travers le composant. Ces bancs sont les plus petits segments adressables de l'espace de la mémoire de configuration et toutes les opérations doivent agir donc sur ces bancs de configurations comme le montre la figure 3.3.b.



(a) Virtex II (b) Virtex 4

Figure 3.3 : Bancs de configuration dans deux familles de FPGA de Xilinx

Cette architecture diffère des précédentes familles Virtex qui sont configurées en bancs couvrant toute la hauteur du composant comme le montre la figure 3.3.a. Dans ce cas, les modules reconfigurables doivent également couvrir la pleine hauteur de ces composants.

3.4.1 Adressage des bancs de configuration :

L'adresse de la rangée (row) identifie une rangée dans le FPGA. Le composant est divisé en deux sections, haute et basse ; les rangées sont numérotées de 0 jusqu'au maximum dans chaque moitié du FPGA, commençant à partir du centre jusqu'au top pour la section haute et jusqu'au bas pour la section inferieure tel qu'illustré dans la figure 3.4. Pour les Virtex 4,5 et 6, une rangée est constituée d'un empilement de blocs de base avec une ligne d'horloge HCLK passant au milieu. Chaque rangée est divisée en même nombre de colonnes où chaque colonne correspond à un des blocs de configuration (CLB, DSP, block RAM, IOB, etc.).

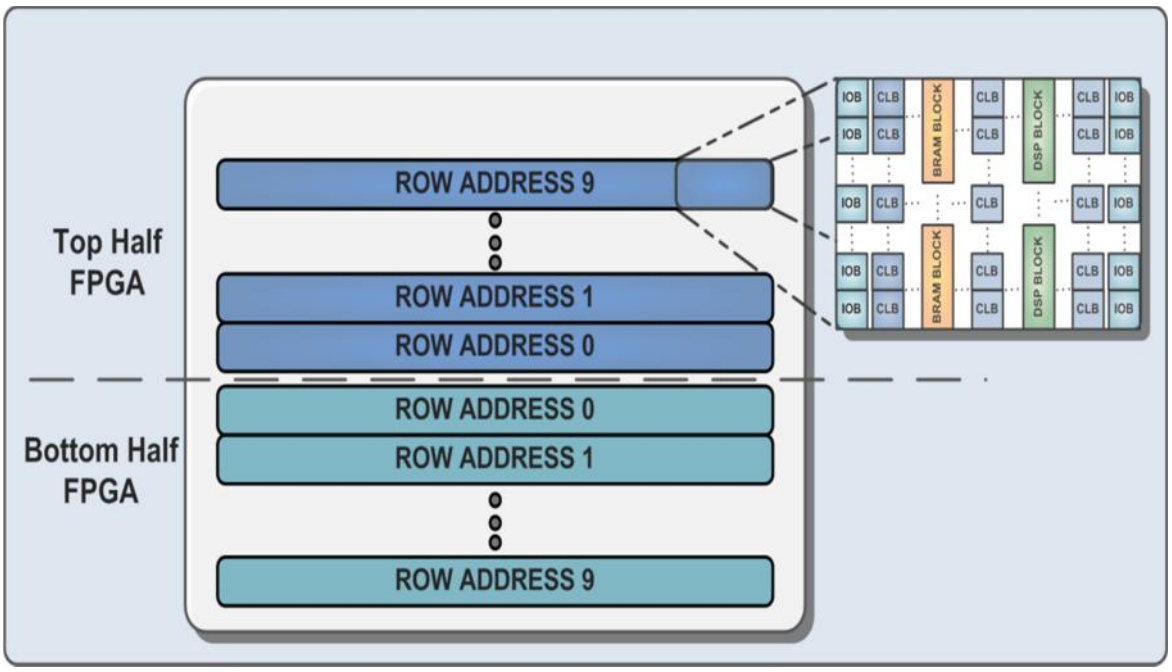


Figure 3.4 : Organisation des ressources dans un FPGA:

Le Tableau 3.1 montre le nombre de blocs dans les différentes colonnes d'une rangée pour les FPGA de type Virtex 4, 5 et 6. L'adresse majeure est numérotée de gauche à droite et commence par la valeur zéro.

Tableau 3.1 : Nombre de blocs par colonne dans une rangée d'un FPGA

Type de colonne	Virtex4	Virtex5	Virtex 6
CLB	16	20	40
DSP	8	8	16
BRAM	4	4	8
IOB	16	20	40

Il y a deux séquences d'adresses majeures par rangée, la première attribue une adresse majeure utilisée pour sélectionner chaque colonne pour une configuration normale alors que la deuxième séquence attribue une adresse majeure pour chaque colonne de bloc RAM utilisé pour accéder aux bancs spécifiant les données contenues dans les blocs RAM. En conséquence, les blocs RAM ont deux adresses majeures : une pour accéder à leur configuration normale, et l'autre pour accéder à leur contenu. Chaque colonne contient un certain nombre de bancs accessibles à partir des adresses mineures. Le nombre de bancs dans une colonne dépend du type de bloc et du type de colonne. Le tableau 3.2 indique le nombre de bancs (adresses mineures) par colonne pour chaque type de bloc dans un Virtex4. Les bancs sont numérotés aussi de gauche à droite, en commençant par zéro.

Tableau 3.2 : Nombre de bancs par colonne

Type de colonne	Nombre de bancs
CLB	22
DSP	21
Interconnexion BRAM	20
Contenu BRAM	64
IOB	30
Clock	2

3.4.2 Registres de configuration

Les bitstreams partiels et totaux pour les Virtex 4, 5 et 6 sont constitués d'un flot de paquets contenants diverses commandes et données de configuration. Toutes les commandes sont exécutées par lecture ou écriture sur des registres de configuration. Il existe deux types d'entête de commande dans un bitstream. Le type 1 comprend un

compteur de mots définissant le nombre de mots de données qui suit la commande. Si le compteur est à 0, le mot suivant doit être l'entête de type 2 qui contient un nombre étendu de mots pour les paramètres suivants.

Dans le tableau 3.3 ci-dessous est représenté le format de type 1 d'entête de commande composé, de gauche à droite: de trois bits indiquant le type d'en-tête, qui sont 001 pour le type 1 et 010 pour le type 2 ; de deux bits définissant l'opération (Opcode), qui sont 00 pour NOP, 01 pour l'opération de lecture et 10 pour l'opération d'écriture sur le registre ; de quatorze bits pour l'adressage de registre, même si seulement cinq d'entre eux sont utilisées; de 2 bits réservés pour une utilisation future et de onze bits contenant le compteur de mots.

Tableau 3.3 : Format de type 1 d'entête de commande dans un bitstream

Type d'entête	Opcode	Registre	Réservé	Compteur de mots
[31:29]	[28:27]	[26:13]	[12:11]	[10:0]
001	xx	RRRRRRRRRxxxxx	RR	xxxxxxxxxxx

Le paquet de données de type 2 doit suivre un paquet de type 1 quand son compteur de mots est égal à zéro, et est utilisé lorsque le nombre de mots de données dépasse 2048 mots. Le format de l'entête de type 2 est indiquée dans le tableau 3.4. Aucune adresse de registre n'est présente dans l'entête, car elle est contenue dans le précédent entête de type 1. Les premiers trois bits signalent le type de l'entête qui est 010, les deux bits suivants sont réservés pour une future utilisation et les 27 bits restants contiennent la valeur du compteur de mots.

Tableau 3.4 : Format d'entête de type 2

Type d'entête	Opcode	Compteur de mots
[31:29]	[28:27]	[26:0]
010	RR	Utilisés tous

Le tableau 3.5 récapitule les différents registres existants dans les plans de configuration des FPGAs de type Virtex 4, 5 et 6. Une explication succincte de quelques registres sera donnée pour comprendre le processus de configuration.

Le registre CRC (Cyclic Redundancy Check) contient la valeur de vérification des données de configuration servant à détecter les erreurs durant la configuration. Une fois les paquets de données de configuration sont générés durant le processus de génération du bitstream, l'outil calculera une valeur de contrôle de redondance cyclique CRC à partir de ces données. A la fin, Le bitstream généré va contenir une commande de vérification du CRC, suivie de la valeur calculée du CRC. Durant la configuration du bitstream dans le FPGA, le composant recalcule la valeur de CRC des données de configuration entrantes. Si cette nouvelle valeur ne correspond pas à la valeur de CRC précalculéé et inclue dans le bitstream, le composant indique une erreur et le processus de configuration s'arrête.

Tableau 3.5: Registres de Configuration

Nom de Reg.	Addr.	Description
CRC	00000	CRC register
FAR	00001	Frame Address Register
FDRI	00010	Frame Data Register Input
FDRO	00011	Frame Data Register Output
CMD	00100	Command register
CTL0	00101	Control register 0
MASK	00110	Masking register for CTL0 and CTL1
STAT	00111	Statusregister
LOUT	01000	Legacy Output Register
COR0	01001	Configuration Option Register 0
MFWR	01010	Multiple Frame Write Register
CBC	01011	Initial CBC value register
IDCODE	01100	Device ID register
AXSS	01101	User Bitstream Access Register
COR1	01110	Configuration Option Register 1
CSOB	01111	Used for daisy chain parallel interface
WBSTAR	10000	Warm Boot Start Address Register
TIMER	10001	Watchdog Timer Register
BOOTSTS	10110	Boot History Status Register
CTL1	11000	Control Register 1

Les FPGA de type Virtex 4, 5 et 6 sont subdivisés en deux moitiés, haute et basse. Les bancs dans la moitié basse constituent une sorte de miroir de leurs homologues dans la moitie haute à l'exception des rangées HCLK qui contiennent des horloges régionales ou globales. Tous les bancs des FPGA de type Virtex 4 par exemple ont une longueur fixe et identique de 1312 bits (41 mots de 32 bits). Un banc peut configurer à la fois 4 blocs BRAM, 32 IOB ou 4 blocs DSP. L'adresse du banc peut être directement écrite ou bien auto incrémenté à chaque fin de banc.

Les bancs reconfigurables sont les entités minimales pour bâtir une configuration partielle. Même si une configuration plus petite à un banc est choisie alors le banc entier sera configuré. Chaque banc de configuration dans le FPGA a une adresse unique de 32 bits appelé FAR (Frame Address Register), composé de cinq paramètres : le type de bloc (Block type) ; l'indicateur haut/bas (Top/Bottom indicator) ; l'adresse du rangée (Row address) ; l'adresse majeure ou adresse de colonne (Major address) et l'adresse mineure (Minor address). L'ordre de ces paramètres dépend de type de FPGA cible comme le montre le tableau 3.6 suivant :

Tableau 3.6 : Composition du registre FAR

Type d'adresse	Index de bits		
	Virtex 4	Virtex 5	Virtex 6
Bit Haut/Bas	[22]	[20]	[20]
Type de bloc	[21:19]	[23:21]	[23:21]
Adresse de rangée	[18:14]	[19:15]	[19:15]
Adresse de colonne	[13:6]	[14:7]	[14:7]
Adresse mineure	[5:0]	[6:0]	[6:0]

Le registre FDRI est utilisé pour configurer les données de Banc à l'adresse spécifiées dans le registre FAR. C'est un registre à décalage dans lequel les données sont chargées avant de les transférer vers la mémoire de configuration. La nouvelle configuration est écrite dans le FPGA en chargeant le registre CMD avec la commande d'écriture de configuration WCFG à condition qu'au moins deux mots de configuration

soient programmés dans le registre FDRI. Le registre FDRO est utilisé pour lire les données de configuration depuis le FPGA dans un processus appelé relecture ou Readback. Cette opération est réalisée en chargeant le registre CMD avec la commande RCFG puis en adressant la FDRO avec une commande de lecture. Le CMD contient les commandes de configuration qui sont utilisées pour gérer la logique de contrôle de la configuration. La commande présente dans le registre CMD est exécutée à chaque fois que le FAR est chargé avec une nouvelle valeur. Les autres registres indiqués dans le tableau 3.5 ont chacun une fonction particulière pour réussir l'opération de configuration. Pour plus d'informations, le lecteur est invité à consulter les guides d'utilisation de configuration des FPGA de Xilinx de type Virtex 4, 5 et 6 [33, 34, 51].

3.4.3 Structure d'un Bitstream

Un bitstream est un fichier binaire qui contient toutes les données de configuration qui seront chargées dans le composant FPGA à travers l'une des interfaces de configuration disponibles. Ce fichier de configuration est composé d'une entête générale contenant des informations sur le type du FPGA cible, la date de création du fichier et sa taille en octets. Ces informations préliminaires sont suivies d'une série de mots de 32 bits contenant des commandes de configuration, des données additionnelles et des blocs de synchronisation. Les commandes sont composées par une entête de commande de type 1 et éventuellement une deuxième entête supplémentaire de commande de type 2 suivi d'un paquet de paramètres en nombre égal au compteur de mots contenus dans les entêtes précédents. Les commandes de groupe CMD ont un aspect particulier, en fait, ils n'ont pas de paramètres et sont constitués par une entête ordinaire qui leur identifie en tant que commande suivi d'un mot qui contient le code de la commande à exécuter. Ces informations initiales sont suivies par les paramètres effectifs, en nombre égal à celui indiqué dans le champ compteur de mots. En plus de ces commandes, deux mots particuliers existent, appelés Dummy Word et Sync Word, qui marquent le début réel du bitstream.

Le tableau 3.7 montre la structure d'un bitstream exemple qui comprend les commandes les plus intéressantes pour le processus de reconfiguration. Deux types de registres ont un intérêt particulier pour la relocation des bitstreams partiels. Le premier est le registre FAR (Frame Address Register) indiquant où est situé un paquet de données de configurations entrantes à l'intérieur du FPGA. Le second est le registre de contrôle de

redondance cyclique (CRC), dont son but est de protéger le FPGA contre toute corruption du fichier bitstream qui pourrait engendrer des erreurs de configuration pouvant endommager le composant.

Tableau 3.7 : Structure d'un Bitstream

Mot de 32bits (Hex)	Designation
...	Entête du fichier Bitstream
...	
...	
...	
...	
FFFFFFFF	Dummy Word
AA995566	Sync Word
...	...
30008001	Entête de Type 1, Ecrire 1 mot à CMD
00000007	Reset CRC
...	...
30002001	Entête de Type 1, Ecrire 1 mot à FAR
Xxxxxxxx	Valeur de FAR
30008001	Entête de Type 1, Ecrire 1 mot à CMD
00000001	Ecriture de Configuration
...	...
30004000	Entête de Type 1, Ecrire 0 mot à FDRI
50000400	Entête de Type 2, Ecrire 1024 motsàFDRI
xxxxxxxx	Mot de données 1
...	...
xxxxxxxx	Mot de donnéesn
xxxxxxxx	Mot de donnéesn+1
xxxxxxxx	Dernier mot de données(1024)
...	...
30000001	Entête de Type 1, Ecrire 1 mot à CRC
xxxxxxxx	Valeur de CRC
...	...

La relocation d'un bitstream partiel est accompli via la manipulation des registres FAR. Connaissant à priori la plage d'adresses de chaque configuration d'une région PRR, un bitstream partiel donné ciblant un PRR choisi peut être modifiée dynamiquement en mettant à jour chaque registre FAR dans le fichier. Cette mise à jour des registres FAR produira un changement dans le contenu du fichier bitstream, les registres CRC doivent donc être également recalculés dynamiquement pour protéger l'intégrité du bitstream et réussir sa reconfiguration.

La relocation fournit donc un moyen pour éliminer la redondance des fichiers bitstreams partiels ; Elle manipule un seul fichier permettant de placer ce bitstream dans n'importe quel PRR et éliminant ainsi les supports redondants de stockage et de communication.

3.5 Les approches de Relocation proposées dans la littérature

Dans cette section, nous présentons quelques approches de relocation proposées dans la littérature. Un classement de ces approches sera donné à la fin en se basant sur les critères de classifications citées dans la section 3.3.

3.5.1 REPLICA2Pro

Le filtre de relocation REPLICA2Pro [52] est un module matériel conçu pour la relocation des tâches matérielles dans des FPGA de type Virtex-II/PRO par la manipulation des registres FAR dans leurs bitstreams correspondants. L'organisation de la mémoire de configuration de Virtex-II/PRO est basée sur des bancs couvrant toute la hauteur du FPGA. Le processus de relocation peut être conduit alors uniquement dans le sens horizontal.

Les entrées du filtre sont le bitstream partiel de la tâche et la région de destination définie par l'adresse de colonne du CLB se trouvant le plus à gauche dans cette région. La sortie principale du filtre est un bitstream de destination qui implémente la tâche dans la région désirée. L'adresse de colonne de destination peut être déterminée par un algorithme de gestion de ressource qui prend en compte la taille de la tâche et l'allocation courante des ressources.

Le filtre REPLICA2Pro est inséré dans le chemin de configuration entre la mémoire de stockage des bitstreams et le gestionnaire de configuration. Toutes les manipulations effectuées se font durant le processus de configuration. Le filtre, tel que présenté dans la figure 3.5, est constitué de quatre blocs fonctionnels.

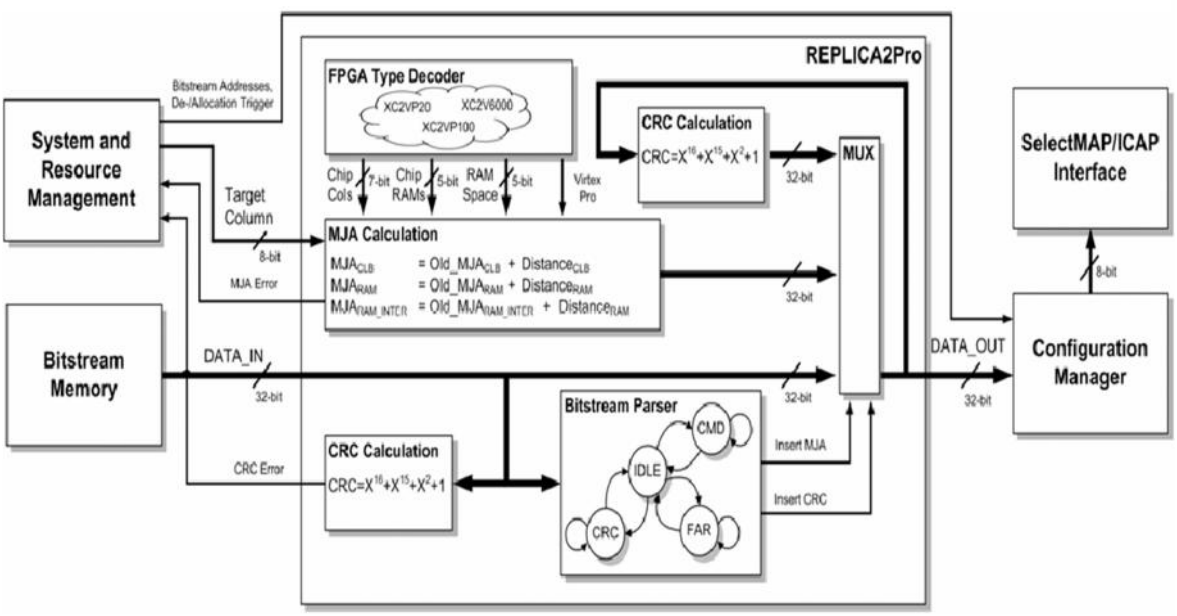


Figure 3.5 : Le filtre de relocation REPLICA2Pro

Le premier bloc (Bitstream Parser) sert à examiner le bitstream en décodant tout d'abord tous les mots du fichier. Il détecte le type du FPGA cible puis il génère les signaux de contrôle nécessaire pour les autres blocs. Il détermine également les positions des registres FAR et CRC dans le bitstream qui vont subir le changement. Le deuxième bloc est le décodeur de type qui permet, à partir du type du FPGA détecté, de générer quelques paramètres spécifiques tels que le nombre de colonne CLB incluses dans le composant, le nombre de colonne RAM/multiplicateur, et l'espace entre deux RAM. Ces paramètres seront utilisés par le bloc de calcul des adresses majeures (MJA Calculation). Ce bloc est le module effectif qui réalise le processus de relocation en calculant toutes les nouvelles adresses FAR du bitstream. Le dernier bloc (CRC calculation) sert à calculer la nouvelle valeur de CRC du bitstream, ceci est nécessaire puisque le filtre the REPLICA2Pro a manipulé les adresses FAR. Les nouvelles valeurs de ce registre doivent être incluses dans un nouveau calcul du CRC du bitstream.

3.5.2 BiRF

L'approche BiRF [53] est un autre filtre matériel de relocation qui cible, en plus des FPGA Virtex-II/PRO, les séries Virtex 4 et Virtex 5 de Xilinx. Le bloc diagramme de BiRF est présenté dans la figure 3.6 ci-dessous.

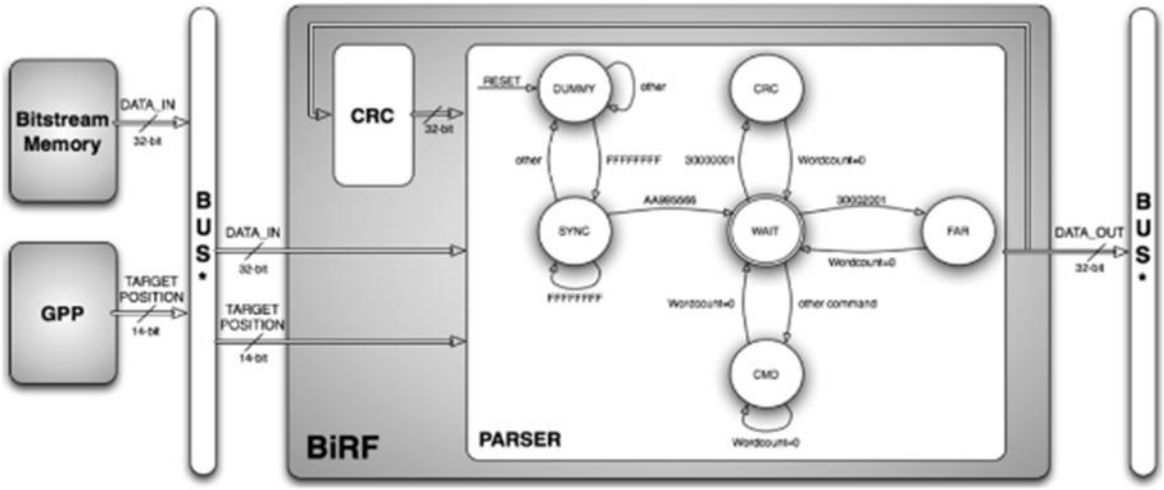


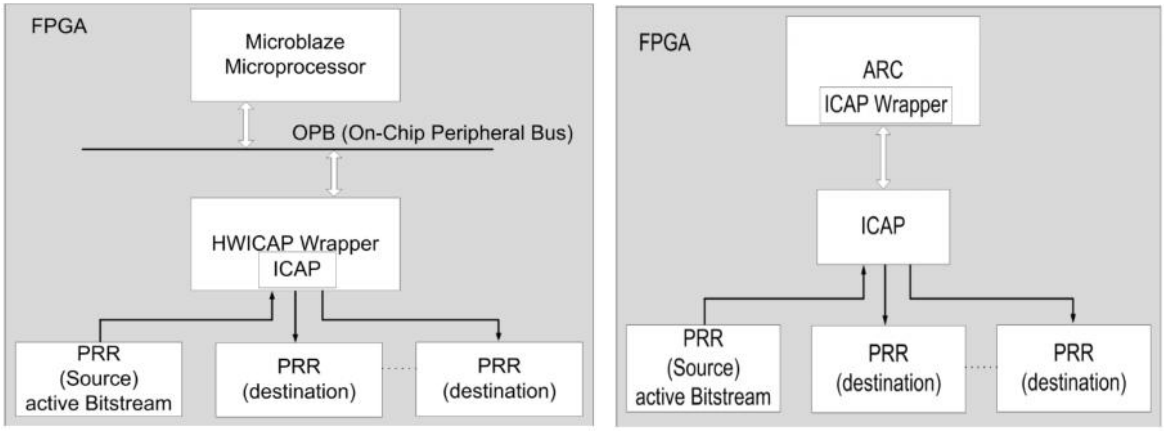
Figure 3.6 : Le filtre de relocation BiRF

Il est constitué d'une unité logique de calcul de CRC dans le bitstream modifié et une machine d'état FSM qui examine le bitstream et effectue les manipulations nécessaires pour permettre sa relocation. Les principales entrées du filtre sont le fichier de configuration dans la forme d'une séquence de mots de 32 bits et l'adresse utilisée pour spécifier l'endroit où le bitstream sera déplacé. Le bitstream pris de la mémoire ainsi que l'adresse de destination donnée par le processeur GPP sont acheminés vers le filtre à travers le bus. La sortie de filtre est le bitstream manipulé envoyé mot par mot vers le même bus. Ce dernier peut être l'un des bus embarqués OPB ou PLB de Coreconnect dans lequel Le filtre BiRF est connecté en tant qu'esclave à travers une interface personnalisée dite IPIF. Il existe également une version logicielle de BIRF [54] exécuté par le processeur.

3.5.3 ARC

R. Kallam et al. [55] proposent une approche intéressante, une technique de relocation d'une PRR à une autre, basée sur le principe de relecture de bitstream

(Readback). Cette méthode lit des données de bancs de configuration directement à partir d'une PRR active (source) et les reloge à la volée dans une PRR de destination. Cela permet d'accélérer la relocation et élimine le besoin d'enregistrer une copie temporaire de bitstream. Deux versions de relocateur ont été réalisées. Une première version matérielle appelée ARC [55] (pour Accelerated Relocation Circuit) (figure 3.7.a) et une deuxième solution logicielle exécutée par le processeur Microblaze [56] (figure 3.7.b). Les deux versions ciblent les FPGAs de type Virtex 4.



(a) la version matérielle, ARC (b) la version logicielle

Figure 3.7 : Le filtre de relocation par relecture de bitstream

3.5.4 Carver et al.

L'approche de relocation proposée par Carver et al. [57] cible la famille Virtex 4 de Xilinx. Il manipule le bitstream via un algorithme exécuté par le processeur embarqué Microblaze. C'est un programme relativement facile à implémenter toutefois il présente quelques limitations dans son exécution. Premièrement, il a besoin d'une mémoire BRAM pour stocker la copie du bitstream relogeable. En plus, le programme n'effectue pas le calcul du CRC. Les auteurs proposent comme alternative de désactiver la vérification de CRC dans le bitstream ce qui peut engendrer des erreurs pendant la configuration si les calculs des nouvelles valeurs de registres FAR lors de la relocation ne se font pas correctement.

3.5.5 Berker et al.

Les auteurs dans [58] proposent un nouveau pilote pour le corps HWICAP pour les FPGA de type Virtex 4. En plus de la fonction de configuration, le pilote analyse le bitstream reçu de la PRR source, identifie et modifie ses adresses FAR et reloge le bitstream dans la PRR de destination désirée. Ce pilote est exécuté par le processeur Microblaze de Xilinx et peut réaliser, sous certaines conditions, la relocation entre deux zones non identiques en termes de distribution de ressources. En plus, une accélération supplémentaire peut être obtenue avec l'aide des circuits matériels qui peuvent communiquer directement avec le port ICAP pour réaliser la relocation. Ils lisent la configuration à partir de la région source dans la mémoire de configuration et procède à sa relocation via une exécution logicielle par le processeur communicant plus rapidement avec la couche matérielle HWICAP.

3.6 Classification des approches de relocation étudiées

Les approches de relocation proposées dans la littérature qui sont implémentées en matériel sont efficaces en termes d'accélération de processus de relocation mais elles nécessitent des ressources logiques additionnelles pour construire le relocateur, spécialement leur module de recalcul du CRC qui consomme le plus de ressources. Les approches logicielles de relocation proposées réalisent également la relocation d'une façon rentable. Ce sont des bonnes solutions pour les FPGA de faibles ressources logiques, Cependant, ils sont inadaptées pour les applications en temps réel puisqu'elles introduisent un temps d'exécution excessif pouvant entrainer des déficits d'ordonnancement de leurs tâches. Le tableau 3.8 récapitule les principales caractéristiques de toutes les approches de relocation mentionnées ci-dessus qui sont basées tous sur une manipulation en ligne du bitstream.

Afin de rendre le processus de relocation moins couteux en matière de temps pris par le relocateur pour manipuler le bitstream et en terme de taux d'utilisation des ressources logiques pour implémenter le relocateur, nous avons proposé une nouvelle méthodologie de relocation que nous avons appelée OORBIT (pour offline/online relocation of bitstream). Elle est basée sur une combinaison de deux méthodes de manipulation du bitstream, en hors ligne et en ligne.

Tableau3.8 : Classification des approches de relocation étudiés

Référence	Relocateur	Type de traitement	Type de relocation	Version du Flot PR
REPLICA2PRO [52]	En-ligne	HW	Normal	9.2
BiRF [53,54]	En-ligne	HW/SW	Normal	9.2
ARC [55,56]	En-ligne	HW/SW	PRR-PRR	9.2
Carver et al. [57]	En-ligne	SW	Normal	8.2
Becker et al. [58]	En-ligne	SW	Normal	9.2

Pour la phase hors ligne, nous avons développé un outil qui permet en première phase d'analyser le fichier bitstream généré par le flot de conception traditionnel de la reconfiguration partielle. Toutes les adresses de configurations dans le fichier seront identifiées et par la suite toutes les régions relogeables possibles seront spécifiées. En second lieu, nous procédons au calcul de nouvelles adresses de configurations pour chaque allocation possible du bitstream. Toutes ces nouvelles informations liées à la relocation seront ensuite ajoutées au bitstream original sous forme d'un complément de données. Le bitstream modifié supportant la relocation sera utilisé durant l'exécution de l'application par notre module développé pour la relocation en ligne qui sera exécuté par le processeur. Il consiste principalement à modifier les anciennes adresses de configuration du bitstream original par ceux précalculés en hors ligne correspondants à la nouvelle allocation désirée. Cette solution permet d'accélérer le processus de relocation d'une façon spectaculaire et cela sans introduire aucune incidence sur les ressources logiques du système. Notre méthodologie sera décrite en détail dans le prochain chapitre.

3.7 Défis posés par le nouveau flot de conception DPR basé sur les partitions

Les discussions dans la section précédente couvrent les différentes approches de relocation vis-à-vis de l'implémentation des modules relocateurs et les types de relocation supportés. Cependant, la classification inclut également le flot de conception déployé du fait que les résultats d'implémentation sont fortement dépendants des contraintes appliquées dans les phases de placement et de routage dans les parties statiques et dynamiques de la conception.

En premier lieu, le nouveau flot de conception DPR, utilise des partitions pins à la place des anciens bus macros pour interconnecter la partie statique avec les modules reconfigurables. Contrairement aux anciens flots déployés, le placement de ces pins dans les différentes régions PRR se fait actuellement d'une façon automatique ce qui entraine souvent une distribution non uniforme de ces pins dans les différentes régions reconfigurables. Pour réussir le processus de relocation, nous devons procéder de sorte que les placements relatifs de partitions pins doivent être uniformisés dans chaque région reconfigurable du système. En conséquence, Une nouvelle fonction d'uniformisation de ces pins a été élaborée et ajoutée à notre nouvelle méthodologie de relocation.

Egalement, dans les précédents flots de conception de la reconfiguration dynamique, on avait la possibilité d'interdire le routage des lignes statiques à travers les régions reconfigurables PRR en utilisant la contrainte ROUTING="CLOSED". Les différentes approches citées dans le tableau 3.8 réalisaient la relocation en utilisant cette contrainte avec un placement contrôlé des anciens bus macros. Cette procédure permettait de réussir la relocation et éliminait toute possibilité d'avoir un conflit de routage qui puisse provoquer un court-circuit dans le FPGA. Cette contrainte de routage n'existe pas dans le nouveau flot de conception DPR basé sur les partitions. En conséquence, même si les partitions pins auront la même forme et placés dans les mêmes endroits que ceux des bus macros, il n'est pas sûr de contraindre les routages statiques dans les interfaces des partitions.

Pour résoudre ce problème de routage et garantir la relocation, nous proposons dans le présent travail une nouvelle procédure qui consiste à intégrer une nouvelle contrainte d'interdiction au niveau de chaque CLB se trouvant aux parois de chaque partition reconfigurable. Cette action permet d'intégrer, pendant la phase de routage de la partie statique, des sortes de bloqueurs qui vont empêcher que les routages statiques traversent les partitions reconfigurables. Une nouvelle fonction intégrant ce concept de bloqueurs a été également élaboré et ajouté à notre méthodologie de relocation qui sera décrite en détail dans le prochain chapitre.

Chapitre 4

LA NOUVELLE METHODOLOGIE DE RELOCATION PROPOSEE

CHAPITRE 4

LA NOUVELLE METHODOLOGIE DE RELOCATION PROPOSEE

4.1 Introduction

Nous avons présenté dans le chapitre précédent les principales approches de relocation proposées dans la littérature basées tous sur une manipulation en ligne du bitstream. Quelques modules de relocation sont implémentés en matériel et des autres sont implémentés en logiciel. Les deux variantes sont généralement efficaces mais présentent les deux des inconvénients. En premier lieu, les approches logicielles exécutées par le processeur embarqué introduisent un temps d'exécution, lors de manipulation de bitstream, qui peut être considéré excessif dans des applications critiques ou en temps réel. En second lieu, les approches de relocation implémentées en matériel sont plus rapides en terme d'exécution mais ont besoin de nombreuses ressources logiques coûteuses pour leurs implémentation en vue d'effectuer toutes les modifications nécessaires pour produire un bitstream relogeable. Afin de rendre le processus de relocation moins coûteux en termes de temps de manipulation de bitstream et de taux d'utilisation des ressources, nous proposons une nouvelle méthodologie que nous appelons OORBIT (pour Offline/Online Relocation of Bitstream). Cette nouvelle méthodologie permettra, comme nous allons le montrer dans le présent chapitre, d'accélérer amplement le processus de relocation et cela sans introduire aucune ressource matérielle supplémentaire pour l'implémenter. Ces performances rendent OORBIT le plus performant dans le domaine.

Dans le présent chapitre, nous allons présenter la méthodologie complète d'OORBIT. Nous décrivons tout d'abord l'outil hors ligne d'OORBIT, en se focalisant principalement sur les deux phases d'analyse et de relocation du bitstream relogeable. La section suivante décrit la manière avec laquelle notre outil de relocation hors-ligne est utilisé dans le contexte de nouveau flot de conception PR basé sur les partitions. Nous détaillons comment OORBIT interagit avec les différents outils utilisés dans le nouveau flot et comment exploite les fichiers intermédiaires pour faire réussir le processus de relocation. Par la suite, nous expliquons la façon avec laquelle ce fichier bitstream relogeable sera utilisé par le module en ligne d'OORBIT pour effectuer la relocation. En

fin de ce chapitre, nous présentons un ensemble d'expériences exécutées en vue d'évaluer et de valider notre approche de relocation. Nous montrons également l'importance de certains paramètres en particulier le temps de relocation et la quantité des ressources nécessaires pour implémenter le composant de relocation dans le système. Ces informations seront utilisées pour comparer notre approche avec les autres approches de relocation étudiées.

4.2 Présentation de notre nouvelle méthodologie de relocation

OORBIT est la nouvelle méthodologie de relocation que nous proposons, basée sur une combinaison de deux approches en hors ligne et en ligne. Pour la phase hors-ligne, nous avons développé un outil qui nous permet, en premier lieu, de procéder à l'analyse des bitstreams partiels obtenus par le flot de conception DPR. Toutes les adresses de configuration FAR seront identifiées et par conséquent toutes les régions de relocation possibles seront spécifiées dans le composant. En deuxième phase, l'outil permet de calculer les nouvelles adresses FAR et les valeurs de vérification CRC pour chacune des partitions possibles obtenues. Ces informations seront ensuite ajoutées au bitstream original sous la forme d'un complément de données en bas de fichier. Le nouveau bitstream généré supportant la relocation sera utilisé plus tard lors de l'exécution de l'application à travers notre module de relocation en ligne qui est une simple routine logicielle exécutée par le processeur embarqué. L'opération de relocation en ligne du bitstream consiste à modifier les anciennes adresses FAR et les valeurs du CRC dans le bitstream original par celles précalculés en hors ligne correspondant à la nouvelle partition souhaitée.

4.2.1 Description de l'outil hors ligne de relocation

L'outil hors ligne d'OORBIT a été créé pour rendre les modules matériels relogeables en manipulant leurs bitstreams partielles correspondants. L'outil développé permet d'effectuer une série d'opérations sur les fichiers bitstreams partiels. Il se compose de deux phases principales qui sont :

4.2.1.1 L'analyse du bitstream

L'analyse du bitstream est la première phase de l'outil (voir la partie supérieure de la figure 4.1). Elle permet d'effectuer une série de fonctions sur le fichier bitstream créé par

le flot de conception DPR de Xilinx [37]. La première fonction accède au fichier de configuration et extrait de son en-tête plusieurs paramètres de configuration. Les deux principaux paramètres d'intérêt sont la référence du FPGA cible et la taille des données de configuration incluses dans le fichier.

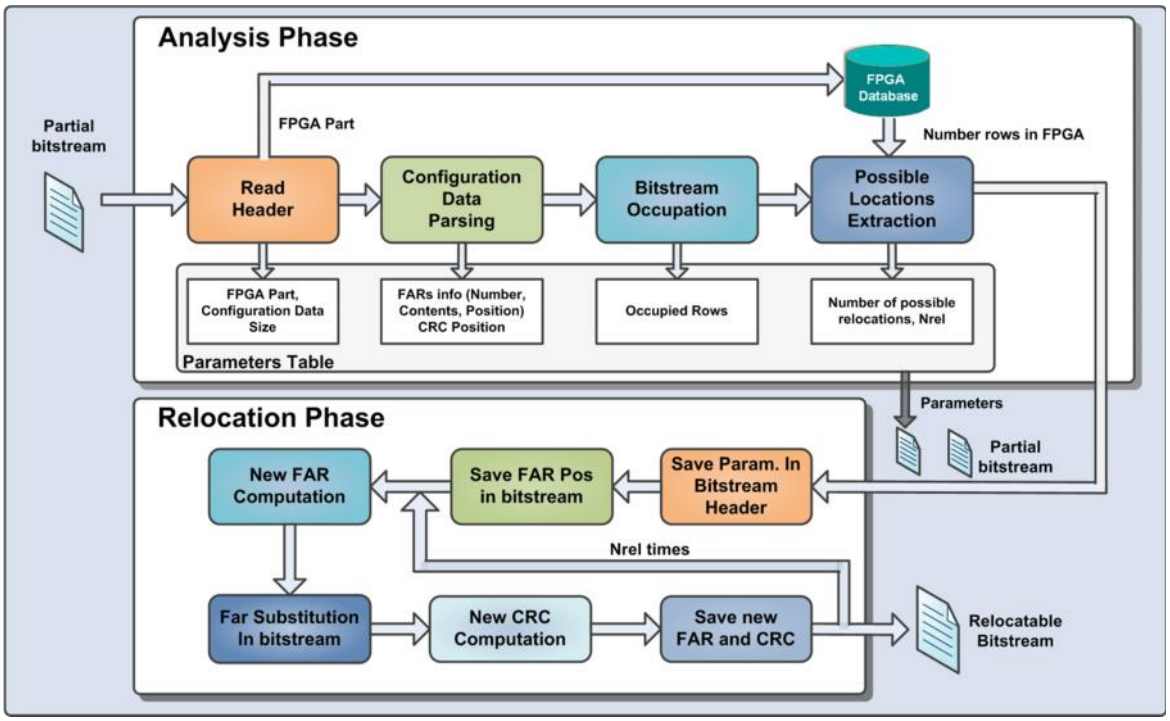


Figure 4.1 : Les différentes opérations de l'outil de relocation hors ligne du bitstream.

La seconde fonction dans cette première partie est l'analyseur du bitstream qui sert essentiellement à décoder tous les mots utilisés dans la configuration. Elle est capable de distinguer entre les commandes et les données, les commandes d'écriture ou de lecture dans le bitstream. Elle cherche principalement les commandes d'écriture des registres FAR et CRC puisque les mots qui les suivent vont subir un changement lors du processus de relocation. Premièrement, les positions et les contenus de ces mots et le nombre de registres FAR (N_{FAR}) sont identifiées dans le fichier. Par la suite, le contenu de chaque registre FAR est décodé en vue de déterminer les cinq champs formant l'adresse du banc de configuration à savoir l'indicateur haut/bas, le type de bloc, l'adresse de la rangée, l'adresse majeure et l'adresse mineure. Cette fonction d'analyse de bitstream supporte toutes les références des FPGA de type Virtex 4, 5 et 6. Pour retrouver proprement ces champs de configurations, la référence du FPGA est prise en compte puisque l'ordre de ces champs diffère d'un type de FPGA à un autre.

Après l'analyse du bitstream, la prochaine fonction détermine le nombre de rangées occupées par le bitstream partiel dans le FPGA. Elle vérifie tout d'abord les adresses des rangées incluses dans le registre FAR analysé. Cette information est nécessaire pour identifier la surface de configuration occupée par le bitstream. Par la suite, une fonction de recherche a été développée à travers une routine de base de données pour déterminer le nombre de rangées disponibles dans le FPGA cible. Elle utilise comme entrée la référence du FPGA déjà extraite de l'entête du fichier. Cette routine de base de données indexe tous les FPGA de type Virtex 4, 5 et 6 et leurs nombres de rangées correspondants. La dernière fonction dans cette phase d'analyse est destinée à déterminer le nombre de relocations possibles (N_{REL}) dans l'espace reconfigurable du FPGA dans lesquelles le bitstream peut être replacé. N_{REL} est calculé en fonction du nombre de rangées occupées par la configuration et de celles disponibles dans le FPGA cible.

4.2.1.2 Processus de relocation

Après la phase d'analyse du bitstream, la phase de relocation est chargée de créer un bitstream partiel relogeable à travers une série de manipulations (voir la partie inférieure de la figure 4.1). La première étape est de sauvegarder les paramètres calculés N_{FAR} et N_{REL} dans l'entête du fichier bitstream comme nouveaux paramètres de configuration à la place du paramètre temps qui peut être ignoré sans affecter l'intégrité du bitstream. Ces deux paramètres seront exploités plus tard par notre outil de relocation en ligne lors du fonctionnement du FPGA. L'opération suivante consiste à sauvegarder en bas de fichier toutes les positions des registres FAR et CRC précédemment identifiées qui seront certainement les mêmes pour toutes les relocations possibles. Ces positions seront utilisées comme adresses d'accès aux mots qui doivent être modifiés lors du processus de relocation en ligne.

Les prochaines fonctions, dans cette phase de relocation hors-ligne, vont procéder au calcul des nouveaux paramètres de configuration pour les différentes relocations possibles déterminées durant la phase d'analyse du bitstream. La première fonction, qui est considérée comme la plus importante dans la phase de relocation, sert à calculer les nouvelles adresses FAR de bitstream. Elle permet de créer pour chaque relocation possible les nouvelles adresses des bancs de configuration qui vont substituer les anciennes dans le bitstream. Notre méthodologie tient compte de la symétrie verticale présente dans les

nouveaux FPGA de type Virtex4, 5 et 6 de Xilinx. Pour cette considération, seulement les deux champs, l'indicateur haut/bas et l'adresse du rangée, parmi les cinq formant l'adresse du banc, seront manipulés pendant le calcul de chaque nouveau paramètre FAR. Ce calcul tient compte aussi des différences qui existent dans la composition des registres FAR pour les différents types de FPGA. Cette fonction consiste principalement à décaler verticalement la région du bitstream par une rangée pour chaque relocation possible commençant du haut vers le bas. Une fois toutes les nouvelles valeurs des registres FAR seront calculées pour la première relocation, ils vont remplacer leurs correspondants situés dans le bitstream original dans les positions identifiées précédemment. Cette substitution effectuée en hors ligne est faite en vue de calculer la nouvelle valeur de vérification CRC du bitstream pour cette relocation. Les FPGAs de Xilinx utilisent le mécanisme CRC, comme nous l'avons expliqué dans le précèdent chapitre, pour détecter les erreurs dans le bitstream durant la configuration. Cette valeur de vérification est calculée pendant le processus de génération du bitstream et encore durant le transfert des données vers le FPGA. Puisque la relocation procède à un changement dans une partie du bitstream (modification des registres FAR), la valeur du CRC doit être recalculée pour éviter les erreurs et permettre au processus de reconfiguration de s'accomplir avec succès. Pour le calcul du CRC, le polynôme utilisé par Xilinx pour les nouvelles familles de FPGA est présenté comme suit :

$$x32 + x28 + x27 + x26 + x25 + x23 + x22 + x20 + x19 + x18 + x14 + x13 + x11 + x10 + x9 + x8 + x6 + 1.$$

Cet algorithme de calcul à 32 bits a été implémenté dans notre outil de relocation hors ligne. Il permet d'analyser le nouveau bitstream mot par mot et procède au calcul de la nouvelle valeur du CRC en utilisant les données de configuration associées à certaines commandes d'écriture dans le bitstream.

Pour le reste, la même procédure sera appliquée : pour chaque relocation possible, on calcule en premier lieu ses nouvelles adresses FAR et on les substitue par la suite avec leurs homologues dans le bitstream original pour calculer la nouvelle valeur de CRC. Une fois les calculs seront faits pour toutes les relocations possibles, tous ces nouveaux paramètres de configuration déterminés (incluant même les paramètres du bitstream original) seront enregistrés à la fin du fichier comme le montre la figure 4.2 suivante.

Figure 4.2: Nouvelle organisation du bitstream effectuée par notre approche de relocation.

L'outil de relocation en hors ligne a été développé sous le langage de programmation Borland Delphi sous forme de modules séparés permettant chacun de traiter une étape de notre approche. Le programme intègre également d'autres fonctions supplémentaires, qui vont être décrites par la suite, servant à interagir avec le nouveau flot de conception basé sur les partitions. Elles procèdent à l'uniformisation de placement des partitions pins dans les régions reconfigurables et à l'intégration de nouvelles mesures pour contraindre le routage statique dans les partions reconfigurables. Tous les algorithmes de manipulation du bitstream ont été implémentés avec une grande précaution non seulement pour garantir l'exactitude des résultats mais encore pour assurer l'intégrité du bitstream qui sera chargé dans le FPGA. La figure 4.3 donne un aperçu sur l'interface graphique de l'outil.

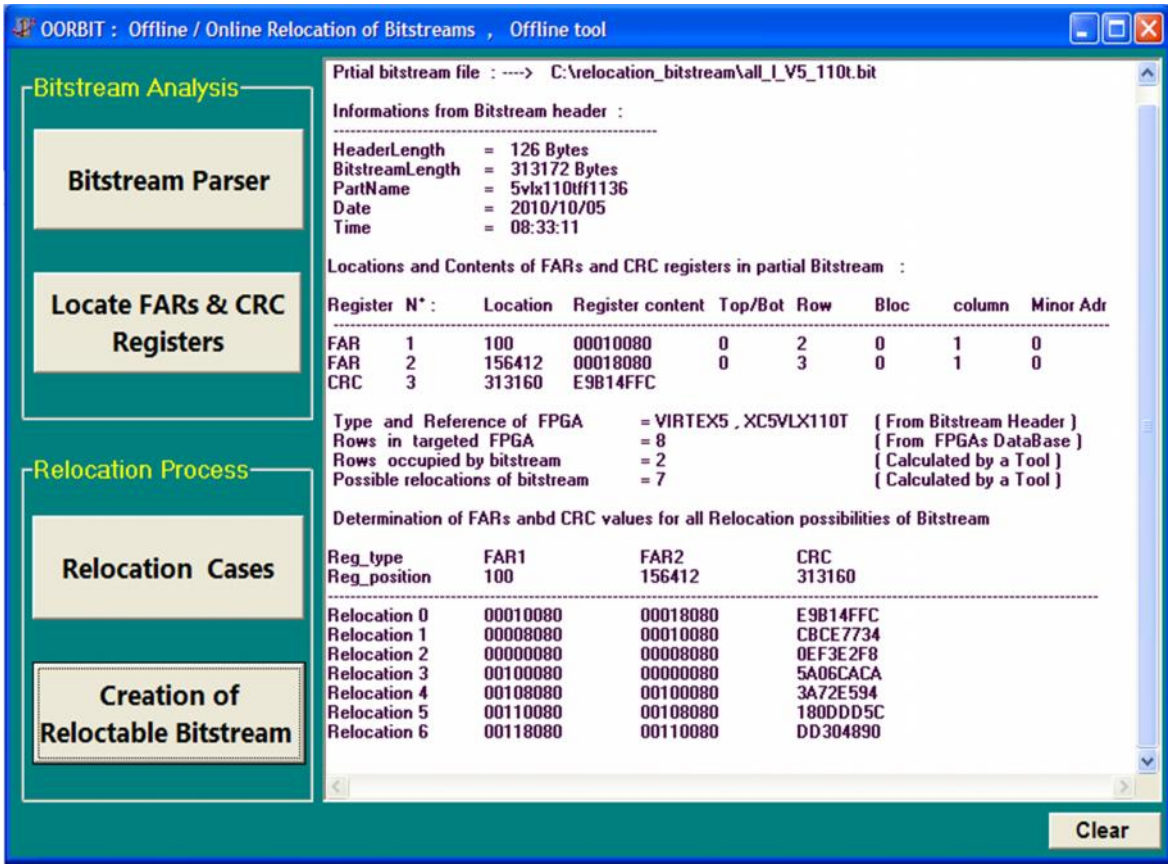


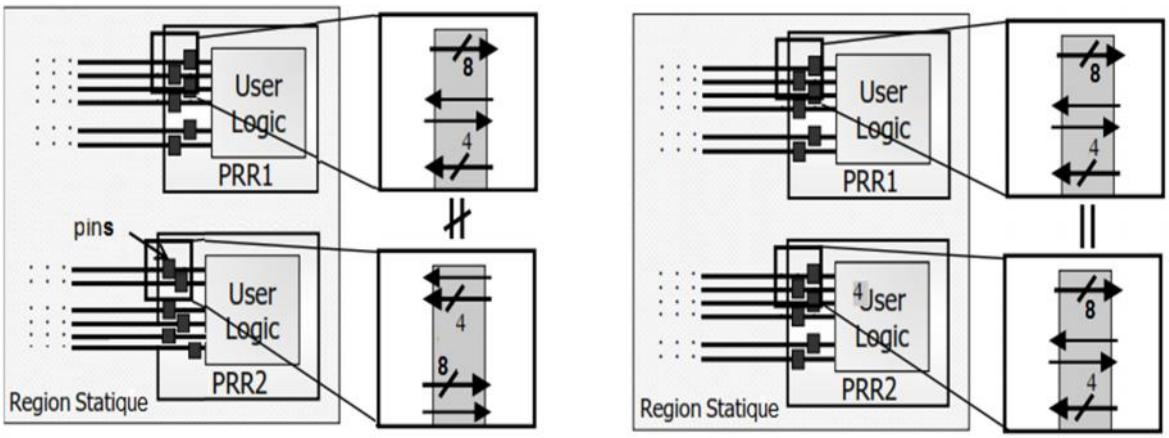
Figure 4.3: Aperçu sur l'interface graphique de l'outil hors ligne d'OORBIT

4.2.2 Nos actions apportées pour supporter la relocation dans le nouveau flot DPR

4.2.2.1 Uniformisation de placement des partitions pins

Dans le nouveau flot de conception DPR, le placement des partitions pins dans les différentes régions PRR se fait d'une manière automatique; l'ordre et les positions de ces pins peuvent changer d'une région à une autre comme le montre l'exemple de la figure 4.4.a. Pour réussir le processus de relocation, nous devons garantir que toutes les interconnexions entre tous les modules PR et la partie statique soient faites de la même manière. Pour cela, les placements relatifs des partitions pins doivent être uniformisés dans toutes les régions reconfigurables du système tel qu'illustré dans la figure 4.4.b.

Lors de la conception du système, les différentes partitions reconfigurables sont définies sans aucune contrainte sur le positionnement initial des partitions pins. Après la phase de routage dans le flot de conception, les informations sur le placement de ces pins peuvent être extraites à partir du fichier de description de circuit généré (NCD).



(a) Placement automatique (b) Placement uniformisé

Figure 4.4 : Exemple d'uniformisation de placement des partitions pins

La commande native PR2UCF génère un fichier de contraintes utilisateurs de type UCF qui regroupe les informations sur le placement des pins pour chacune des régions reconfigurable existantes dans le système. Si on veut obtenir ces informations à partir d'une conception A, il faut exécuter, au niveau de la ligne de commande de l'outil PlanAhead, la commande suivante :

Pr2ucf Conception_A.ncd –o pin_partition_Conception_A.ucf

La figure 4.5 présente un exemple du fichier UCF généré. Les partitions pins sont implémentées en utilisant des LUT contenu dans un slice qui est un composant d'un bloc configurable CLB. Les coordonnées de ces pins sont donc liées aux coordonnées des slices où sont implémentés. En vue d'harmoniser le placement des pins dans toutes les régions reconfigurables, on a élaboré une fonction d'uniformisation qui traite avec le fichier UCF généré.

Cette fonction, qui a été intégrée dans notre outil de relocation hors-ligne, consiste au départ à maintenir les pins de la première partition reconfigurables se trouvant en haut du FPGA dans leurs positions initiales et les prendre comme référence. Par la suite, on recopie la disposition de ces pins dans toutes les autres régions et on procède au changement de coordonnées de la manière suivante : les coordonnées en X seront maintenues telles qu'elles sont pour toutes les partitions puisque la relocation se fera uniquement dans le sens vertical. Les coordonnées en Y des pins vont subir un décalage avec un pas à chaque passage d'une partition à une autre. Ce pas correspond au nombre de

CLB se trouvant dans une colonne d'un rangée de configuration du FPGA. Sa valeur diffère d'un type de FPGA à un autre comme le montre le tableau 3.1 du précédent chapitre. Elle est de 16 CLB pour le Virtex 4, de 20 CLB pour le Virtex 5 et de 40 CLB pour le Virtex 6.

```
# Location constraints for Reconfigurable Module Partition Pins

PIN "dct_0/dct_0/USER_LOGIC_I/rp_instance.Reset" LOC = SLICE_X38Y108;
PIN "dct_0/dct_0/USER_LOGIC_I/rp_instance.ain(0)" LOC = SLICE_X40Y110;
PIN "dct_0/dct_0/USER_LOGIC_I/rp_instance.ain(1)" LOC = SLICE_X40Y110;
PIN "dct_0/dct_0/USER_LOGIC_I/rp_instance.ain(10)" LOC = SLICE_X40Y108;
PIN "dct_0/dct_0/USER_LOGIC_I/rp_instance.ain(11)" LOC = SLICE_X40Y108;
                                        ⋮
PIN "dct_1/dct_1/USER_LOGIC_I/rp_instance.Reset" LOC = SLICE_X39Y97;
PIN "dct_1/dct_1/USER_LOGIC_I/rp_instance.ain(0)" LOC = SLICE_X39Y98;
PIN "dct_1/dct_1/USER_LOGIC_I/rp_instance.ain(1)" LOC = SLICE_X37Y99;
PIN "dct_1/dct_1/USER_LOGIC_I/rp_instance.ain(10)" LOC = SLICE_X37Y98;
PIN "dct_1/dct_1/USER_LOGIC_I/rp_instance.ain(11)" LOC = SLICE_X37Y98;
                                        ⋮
```

Figure 4.5 : Extrait du fichier UCF généré par la commande PR2UCF

Une fois l'uniformisation est finalisée, toutes ces nouvelles informations seront ajoutées au fichier principal des contraintes du système (system.ucf) pour permettre au flot de conception DPR de refaire la phase le routage avec le nouvel arrangement uniformisé des partitions pins dans le FPGA.

4.2.2.2 Nouvelle technique pour contraindre le routage

Nous avons vu dans le précédent chapitre que toutes les approches de relocation qui ont été proposées dans la littérature utilisaient la contrainte ROUTING="CLOSED, inclue dans tous les anciens flots de Conception DPR, pour interdire aux lignes de routage statiques de traverser les régions reconfigurables. Cela permettait de supporter le processus de relocation et de protéger également le FPGA de tout éventuel court-circuit dû à un conflit de routage. Dans le nouveau flot de conception DPR déployé et pour des raisons que nous ignorons jusqu'à présent, cette contrainte de routage n'existe plus dans la liste de contraintes. Le fait que nous sommes les premiers à notre connaissance qui traitent avec la relocation dans le contexte de nouveau flot de conception DPR, un grand effort a été déployé en vue de trouver une solution efficace à ce problème de routage.

Après un examen profond de toutes les contraintes [59] supportées par le flot, nous avons remarqué que la contrainte nommée PROHIBIT, qui intervient au niveau d'un slice dans un CLB, permet d'interdire l'utilisation de ce slice dans la logique ou pour le routage. Le slice avec la contrainte en question peut jouer donc le rôle d'un bloqueur de routage dans l'endroit où se trouve. Ce concept de bloqueurs a été exploité dans notre nouvelle méthodologie de relocation en vue de réaliser des sortes de barrières de routage entre la région statique et les partitions reconfigurables. L'insertion de ces bloqueurs est effectué par l'application de la contrainte PROHIBIT au différents slices des CLB se trouvant aux parois de chaque partition reconfigurable du système. La fonction que nous avons élaborée dans ce contexte, permet tout d'abord de recenser tous ces slices avec leurs coordonnées correspondantes en exonérant de cette liste les slices où se trouvent les partitions pins. Par la suite, on utilise l'expression normalisée de la contrainte pour chaque slice classé tel que présenté dans la figure 4.6.

```
                                    ⋮
CONFIG PROHIBIT = "SLICE_X6Y47"; # generated_by_OORBIT
CONFIG PROHIBIT = "SLICE_X6Y46"; # generated_by_OORBIT
CONFIG PROHIBIT = "SLICE_X6Y45"; # generated_by_OORBIT
CONFIG PROHIBIT = "SLICE_X6Y44"; # generated_by_OORBIT
CONFIG PROHIBIT = "SLICE_X6Y43"; # generated_by_OORBIT
                                    ⋮
```

Figure 4.6 : Exemple d'application de la contrainte PROHIBIT

Cette nouvelle liste de contrainte sera ajoutée au fichier UCF de système, conjointement avec les nouvelles positions uniformisées des partitions pins, pour refaire uniquement la phase de routage de la partie statique. Lors de l'implémentation des modules reconfigurables, toutes les lignes de contrainte PROHIBIT, nouvellement ajoutées dans le fichier UCF, doivent être effacé pour libérer les slices correspondants et permettre de les utiliser pour implémenter ces modules.

4.2.3 L'outil de relocation hors ligne dans le contexte de nouveau flot de conception pour la reconfiguration partielle

Le but de cette section est de définir l'interaction de notre outil de relocation hors ligne avec le flot standard de conception de la reconfiguration partielle de Xilinx. L'outil

proposé coexiste avec le flot de conception et peut être vu comme une extension de ses fonctions. Les entrées à l'outil sont fondamentalement l'information de la phase d'exécution produite par l'outil PlanAhead de Xilinx et les bitstreams partiels générés. Le flot original demeure sans changement comme le montre la figure 4.7; une étape intermédiaire sert à obtenir les informations sur les positions des pins partitions, ces derniers vont être uniformisés par notre outil pour assurer qu'ils soient placés de la même manière pour toutes les partitions reconfigurables définies. Ils seront réinjectés à nouveau dans le fichier des contraintes UCF du système. Ce dernier va inclure également les endroits des bloqueurs que nous avons élaborés pour interdire l'inclusion des lignes de routage statique dans les partitions reconfigurables

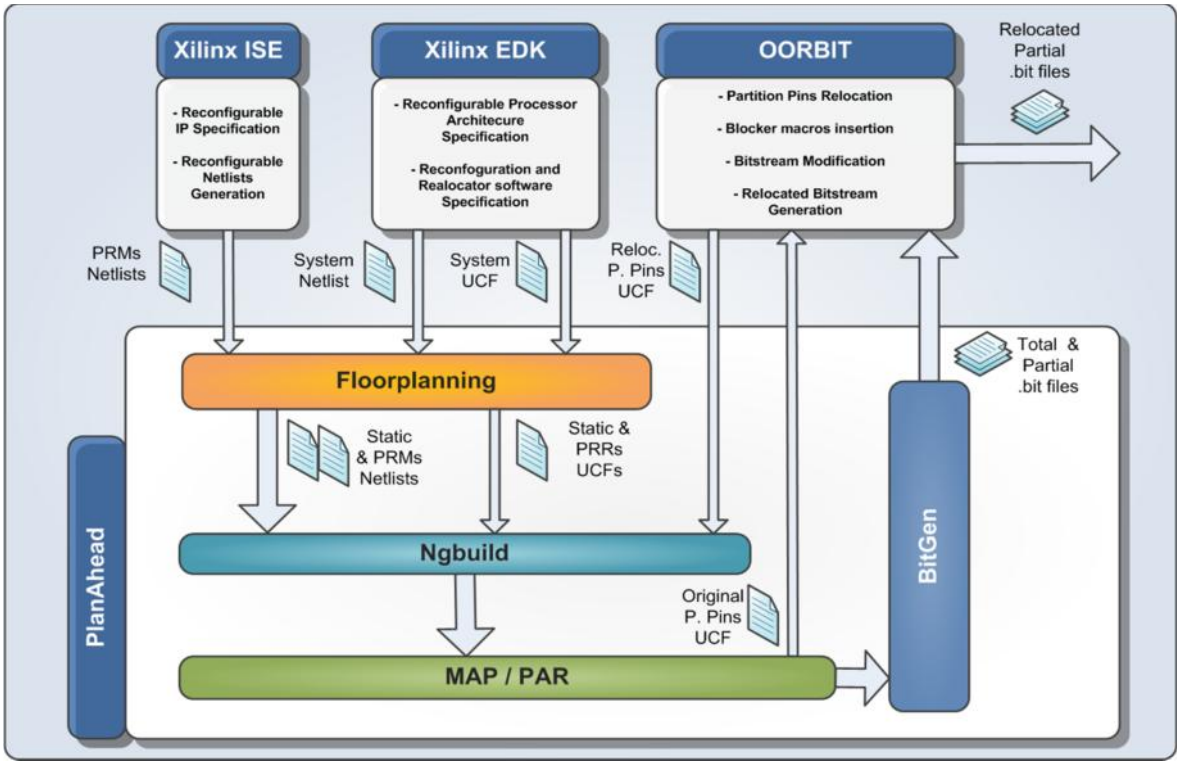


Figure 4.7 : Extension du flot de conception DPR avec le module de relocation hors ligne d'OORBIT

En conséquence, le flot de conception DPR produira de cette manière des bitstreams partiels parfaitement exploitables par notre outil de relocation hors ligne en vue de générer des bitstreams relogeables prêts à être utilisés par la suite par notre module de relocation en ligne.

4.2.4 Description du module de relocation en ligne

Le module de relocation en ligne (relocateur) est un programme software exécuté par le processeur embarqué (i.e. Le Microblaze). Il exploite les fichiers bitstreams générés par l'outil de relocation hors ligne d'OORBIT. Une étape d'initialisation est nécessaire au début de l'application en vue d'accélérer l'opération de relocation. Elle consiste à transférer tous ces fichiers depuis la mémoire de sauvegarde non volatile (i.e. Compact Flash) vers une mémoire SDRAM. De cette manière, l'accès aux données de configuration et leurs manipulations n'est pas seulement une opération facile mais plus rapide. Pendant l'opération de transfert, on extrait de chaque bitstream leurs principaux paramètres de configuration à partir de l'entête du fichier; spécialement le nombre de relocations possibles (N_{REL}), le nombre des adresses FAR (N_{FAR}) et la taille des données de configuration. Le module de relocation est exécuté suite à un appel de placement. Il reçoit comme entrées les paramètres de configuration cités et le nouvel endroit désiré du bitstream. Le relocateur lit alors les paramètres de relocation enregistrées en bas de fichier qui décrit les positions et les contenus des registres FAR et CRC correspondants à la partition de relocation désirée. Il les substitue avec leurs équivalents dans la configuration de base comme illustré dans la figure 4.2 et avisera le processeur pour commencer le processus de reconfiguration du bitstream relogé dans la partition désirée via le port de configuration ICAP intégré dans le FPGA.

4.3 Banc expérimental de la nouvelle méthodologie de relocation.

Dans la présente section, nous présentons la plateforme reconfigurable que nous avons créée pour tester et valider notre nouvelle méthodologie de relocation. Son architecture présentée dans la figure 4.8, est réalisée autour du processeur softcore Microblaze, qui sera implémenté dans une série de FPGAs de type Virtex 4, Virtex 5 et Virtex 6. La présence du processeur dans la conception répond au moins à deux nécessités. Initialement, La configuration statique sera programmée dans le FPGA au début de fonctionnement alors que les bitstreams partiels vont être chargés selon le besoin pendant l'exécution de l'application. Le processeur peut fournir dans ce cas le meilleur mécanisme pour gérer efficacement le processus de reconfiguration. En second lieu, l'utilisateur peut planifier les reconfigurations suite à des évènements spécifiques dans l'application. La

présence du processeur dans le système peut faciliter considérablement cette tâche de synchronisation.

La partie statique de l'architecture proposée comporte le processeur Microblaze, avec son mémoire système, connecté à une série de périphériques via un bus de communication de type PLB. Ces périphériques donnent un support aux différentes tâches de système. Le module ACE permettra de charger les bitstreams partiels à partir de la mémoire compact Flash (CF). Le contrôleur DDR2 gère les opérations de lecture/écriture à partir d'une mémoire externe de type DDR2/SDRAM. Dans cette mémoire, les bitstreams partiels relogeables seront transférés à partir de la mémoire CF en vue d'accélérer leurs éventuelles relocations et reconfigurations. Le module UART fournit un standard de communication entre l'application et l'utilisateur via une interface de communication comme HyperTerminal de Windows.

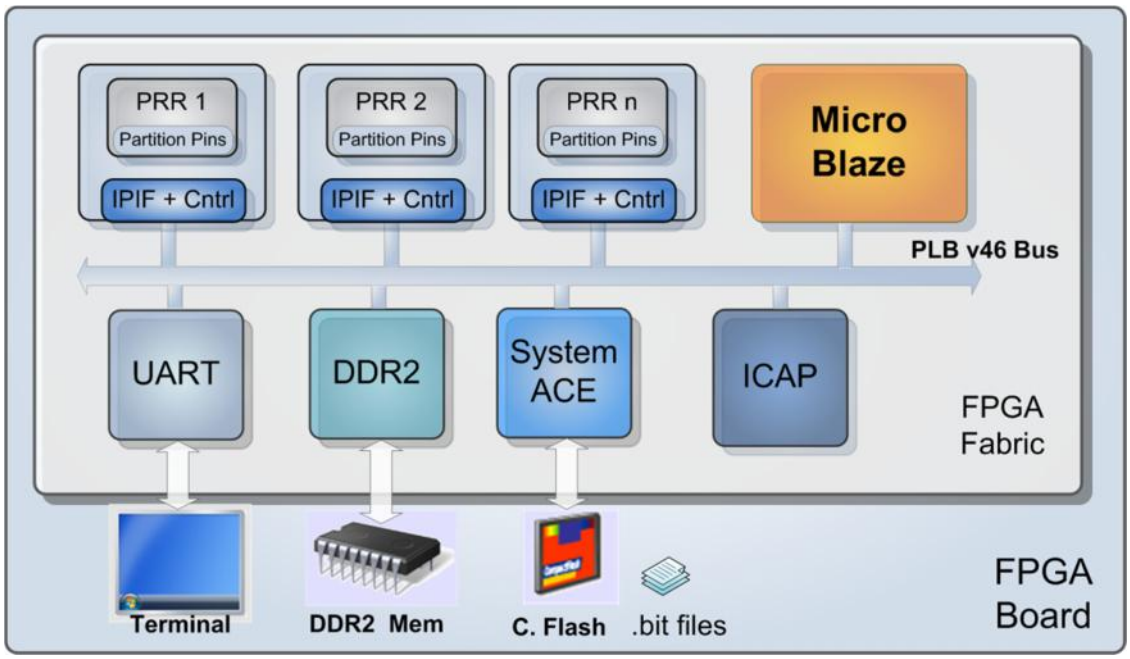


Figure 4.8 : Plateforme de validation de la nouvelle approche de relocation proposée

Les composants partiellement reconfigurables sont présentés par les modules en gris dans la figure 4.8. Le composant de configuration ICAP, commandé par le processeur, permet de reconfigurer le FPGA avec les bitstreams partiels des différents modules reconfigurables du système. Ces modules sont séparés de la logique statique à travers des partitions pins décrites précédemment. Ces pins n'ont pas besoin d'être contrôlées par le

processeur comme le cas des anciens Bus Macros, mais on doit seulement garantir que la logique soit initialisée après chaque reconfiguration pour éviter tout comportement imprévu. L'initialisation peut être réalisée par le processeur directement après le chargement du bitstream partiel par l'activation du signal Bus2IP_reset connecté à la partition PRR reconfigurée.

Le système reconfigurable de la figure 4.8 a été synthétisé dans l'environnement de développement EDK de Xilinx en définissant les régions PRR comme boites noires dans les fichiers Top Level du système. Les netlists de tous les modules PRM, qui vont intégrer le système, seront synthétisés indépendamment en utilisant l'environnement ISE de Xilinx. Après la phase de synthèse, tous les fichiers de conception seront importés vers l'outil PlanAhead pour compléter l'implémentation de notre système. On va définir tout d'abord les différentes régions reconfigurables PRR dans le FPGA puis on va leur associer leurs modules PRMs définis. Pour notre cas de relocation, l'ensemble de modules reconfigurables du système seront également associés à la première région en vue de générer tous les bitstreams partiels liés à cette région. Cela facilite leur traitement plus tard par notre outil de relocation en hors ligne pour les rendre relogeables. Après une première phase de placement-routage, on extrait de la conception les informations liées aux coordonnées des régions reconfigurables ainsi que les positions de partitions pins initialement implémentées. Ces informations seront utilisées par notre outil pour uniformiser l'emplacement de ses pins et pour insérer également les bloqueurs de routage avant de passer à la phase finale de placement et de routage. La figure 4.9 illustre l'organisation des modules PRR dans le FPGA. Chaque PRR défini occupe un rang entier dans le côté gauche du composant. Ce côté est choisi puisque il comporte à la fois les modules BRAM et DSP et présente en plus une distribution plus homogène des ressources logiques.

4.4 Résultats et discussions

Dans cette section, nous présentons un ensemble d'expériences menées en vue d'évaluer et de valider notre approche de relocation. Nous montrons également l'importance de certains paramètres en particulier le temps de relocation et la quantité des ressources nécessaires pour implémenter le composant de relocation en ligne dans le système.

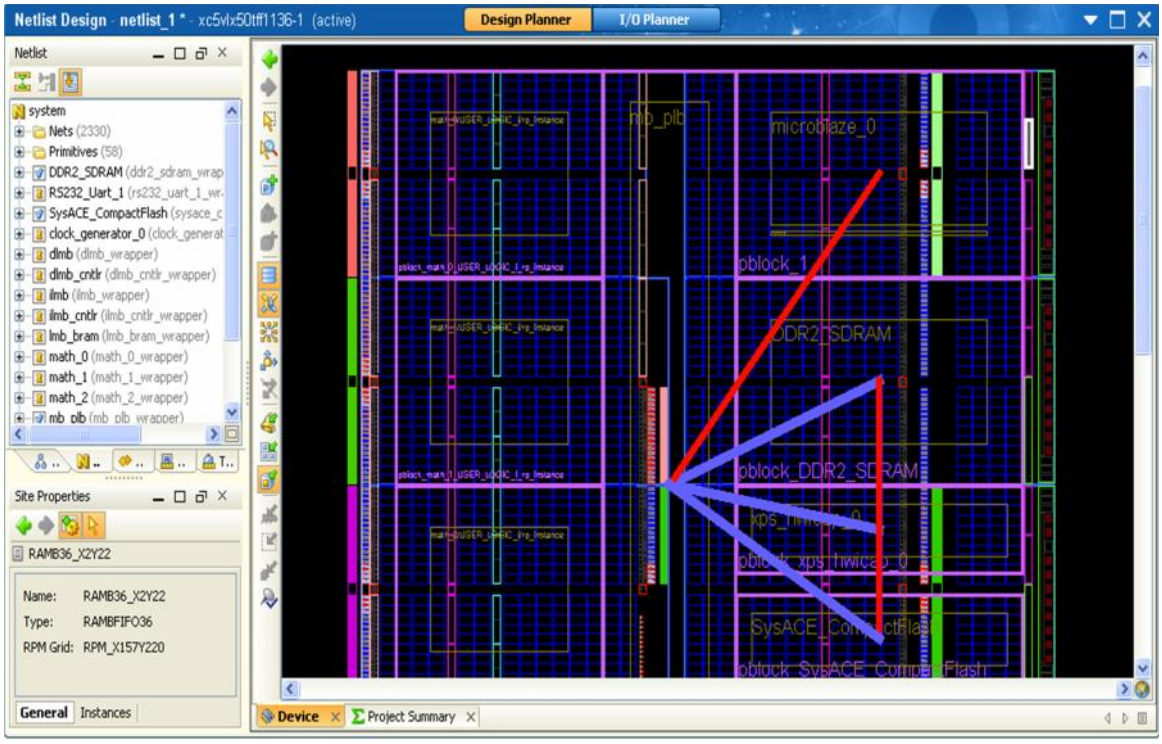


Figure 4.9 : Emplacement physique des régions PRR et des autres modules statiques dans le FPGA

Tout d'abord, une estimation du temps de relocation est donnée au moyen d'un calcul analytique sur plusieurs instances d'un module reconfigurable. Ces informations seront utilisées pour comparer notre approche avec ceux déjà étudiées en utilisant trois modules reconfigurables comme référence. Ces modules vont servir pour montrer l'impact de temps de relocation sur la performance d'un système dit scalable dans lequel un module sera implémenté simultanément sur plusieurs régions reconfigurables. Finalement nous discutons l'aspect de consommation des ressources dans les différentes approches de relocation et les avantages apportés par notre méthodologie.

4.4.1 Estimation de temps de relocation.

Chaque bitstream partiel est utilisé pour configurer une portion du FPGA composée d'un nombre défini de rangées. Deux adresses FAR au maximum sont assignées à chaque rangée dans le bitstream. La première adresse est utilisée pour accéder aux bancs de configuration normaux alors que la seconde adresse accède au contenu des BRAM. Cette dernière est utilisée seulement si les BRAM sont configurés dans la rangée. Le calcul de temps, qui prend le module de relocation pour modifier un bitstream original enregistré

dans la mémoire, est relativement simple. Il correspond aux nombres de cycles utilisé pour modifier tous les registres FAR et CRC présents dans le bitstream par ceux précalculés et enregistrés en bas de fichier bitstream correspondant à la nouvelle partition. Dans notre cas, nous avons utilisé une mémoire extérieure de type DDR2-SDRAM pour enregistrer les bitstreams relogeables. Le temps de relocation nécessaire pour retrouver et modifier les registres concernés est évalué par la formule suivante:

$$T_{relocation} = T_{read_{pos}} + T_{read_{regs}} + N_{regs} \times T_{write_{reg}} \quad (1)$$

T_{read_pos} est le temps nécessaire pour lire les positions des registres FAR et CRC à partir du bas de fichier bitstream, T_{read_regs} est le temps pris pour lire les nouvelles valeurs des registres FAR et CRC, N_{regs} est le nombre de registres à modifier et T_{write_reg} est le temps nécessaire pour écrire la nouvelle valeur de chaque registre dans le bitstream.

Le temps de transfert nécessaire pour réaliser une opération de lecture/écriture à partir de la mémoire DDR2 en mode rafale (Burst mode) est égal au temps de latence de la mémoire (estimé à 19 cycles d'horloge) plus un cycle d'horloge par mot de 32 bits. Le tableau 4.1 fournit une estimation des temps de relocation minimal et maximal d'un bitstream partiel en fonction du nombre de rangées qu'il occupe dans le FPGA à une fréquence de fonctionnement de 100 Mhz.

Table 4.1 : Temps de relocation en fonction du nombre de rangées occupées

Nombre de rangées occupés	Nombre Min des Registres modifiés	Nombre Max des Registres modifiés	Temps Min (µs)	Temps Max (µs)
1	2	3	0.82	1.04
2	3	5	1.04	1.48
3	4	7	1.26	1.92
4	5	9	1.48	2.36
5	6	11	1.70	2.80
6	7	13	1.92	3.24
7	8	15	2.14	3.68
8	9	17	2.36	4.12
9	10	19	2.58	4.56
10	11	21	2.80	5.00
11	12	23	3.02	5.44

4.4.2. Etude comparative entres les approches de relocation.

En vue de ressortir les performances de notre approche en termes de temps de relocation, nous avons pris le même exemple étudié dans [60], avec 3 modules reconfigurables implémentés sur la même plateforme (Virtex 4 fonctionnant à 100 Mhz).

Ces modules sont le transformé en cosinus discret (DCT), le transformé en ondelettes discret (DWT) et le convertisseur de l'espace de couleurs (CSC). Le processeur réalise la relocation de bitstream et envoie les nouvelles données de configuration vers le module HWICAP en mode rafale qui va les transférer immédiatement vers le composant ICAP qui réalise la reconfiguration à la vitesse de 100MB/s [48].

Dans le tableau 4.2 nous fournissons les ressources utilisées, la taille de bitstream et le nombre de rangées occupées par chacun de ces modules dans le FPGA. Nous comparons aussi les temps de relocation de chaque bitstream entre l'approche de relocation ARC [55], les deux versions matérielle et logicielle de l'approche BIRF [53] et notre approche OORBIT. Nous donnons enfin dans la dernière colonne le temps de reconfiguration de chaque bitstream.

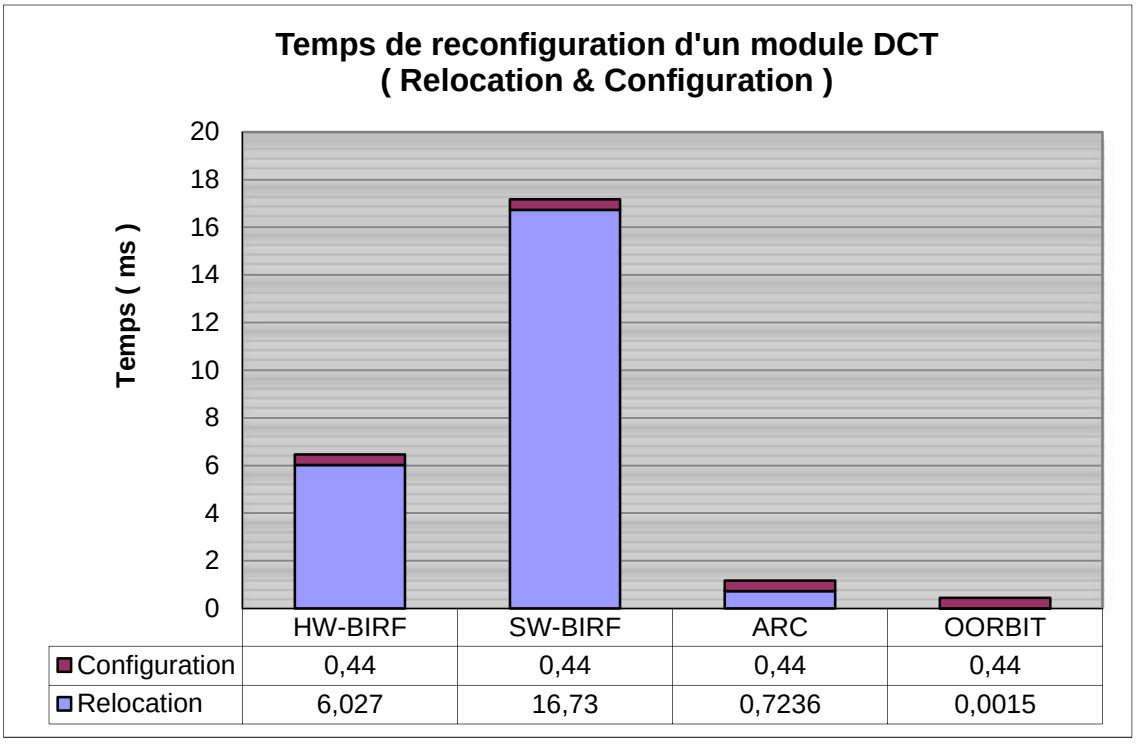
En terme de temps de relocation, nous observons une accélération moyenne avec OORBIT de plus de 3000 et 8000 fois respectivement par rapport à HW-BiRF et SW-BiRF. Cette grande accélération dans le processus de relocation apportée par notre relocateur est due principalement au fait que pour les deux versions de BIRF, un temps excessif a été investi pour recalculer la valeur de CRC. Ce temps est éliminé dans notre approche puisque les valeurs de CRC sont précalculées en hors ligne pour toutes les partitions possibles du bitstream dans le FPGA.

Tableau 4.2 : Comparaison des temps de relocation entre OORBIT et les autres approches

Cas	Ressources FPGA (Virtex 4VLX25)				Bitstream		Temps de relocation (ms)				Temps pour configurer (ms)
	LUT	**FF**	**DSP**	**BRAM**	**KB #**	**Rangée utilisés**	**HW BIRF [53]**	**SW BIRF [54]**	**ARC [55]**	**OORBIT**	**HWICAP [48]**
DCT	**1419**	**1636**	**8**	**8**	**44**	**2**	**6.0270**	**16.7300**	**0.7236**	**0.0015**	**0.44**
DWT	**940**	**389**	**0**	**4**	**47**	**1**	**6.4383**	**17.8707**	**0.4060**	**0.0010**	**0.47**
CSC	**318**	**438**	**1**	**12**	**17**	**3**	**2.3287**	**06.4638**	**0.4033**	**0.0019**	**0.17**

Similairement, les résultats obtenus avec notre approche montrent encore une accélération moyenne de plus de 300 fois par rapport à la méthode de relocation dite ARC basée sur le principe de Readback, cela est attribué au temps additionnel pris par cette dernière pour lire tous les bancs de configuration avant de passer à la relocation.

En Outre, On remarque que les autres approches de relocation étudiées ont dégradé la vitesse de reconfiguration à 7.3 MB/s et à 2.6 MB/s respectivement pour HW-BiRF et SW-BiRF et à moins de 30 MB/s en moyenne pour ARC. Avec OORBIT, le temps de relocation d'un bitstream est presque négligeable comparé à son temps de reconfiguration; il est plus de 100 fois moins en moyenne. La vitesse maximale de reconfiguration d'ICAP qui est de 100MB/s est donc atteinte. La figure 4.10 montre bien ses performances en comparant les différentes approches en termes des temps de relocation et de reconfiguration d'un module DCT utilisé comme référence. Les temps de reconfiguration demeurent relativement constants, alors que les temps de relocation varient selon l'approche. Nous pouvons observer que les performances d'OORBIT surpassent de loin les autres approches. Ce facteur est largement considéré dans beaucoup d'applications exécutées en temps réel exigeant une commutation rapide des tâches IP via un processus de reconfiguration partielle.



	HW-BIRF	SW-BIRF	ARC	OORBIT
Configuration	0,44	0,44	0,44	0,44
Relocation	6,027	16,73	0,7236	0,0015

Figure 4.10 : Temps de relocation et de configuration d'un module DCT

4.4.3 Discussion sur les systèmes largement scalables

La discussion précédente s'est focalisée sur le temps de relocation requis pour un seul module. Cependant, les techniques de relocation sont mieux exploitées par les applications dans lesquelles plusieurs implémentations de la même tâche sont nécessaires. La discussion qui suit vise à souligner l'importance de réduire autant que possible le temps de relocation ce qui présente d'ailleurs l'objectif de notre outil de relocation OORBIT. Examinons par exemple les applications dans lesquelles la scalablité fonctionnelle est une nécessité. Quelques exemples prises de l'état de l'art des modules scalables sont : le transformé en ondelettes discret (DWT) et le transformé en cosinus discret (DCT) de plusieurs tailles. Une approche directe pour créer des modules scalables de tailles variables serait d'implémenter la même tâche dans plusieurs endroits avec différentes conditions d'exécution et de surface et de charger le plus approprié dans le système selon les ressources de matériel disponibles.

Parmi les architectures appropriées qui implémentent des solutions scalables et qui utilisent la reconfiguration dynamique partielle, on trouve les réseaux systoliques [49, 50] qui sont les plus couramment employés. L'arithmétique distribuée fournit les solutions scalables pour effectuer des opérations arithmétiques, alors que les architectures systoliques peuvent exécuter des tâches à calcul intense dans un large champ d'applications.

A titre d'exemple, les auteurs dans [50] présentent une architecture scalable à base d'un FPGA pour un calcul DCT. Ils achèvent la scalablité en effectuant les opérations 2D-DCT pour différentes zones, de 1×1 à 8×8. Leur architecture scalable est ajustée à travers le processus de reconfiguration partielle pour exécuter différents types de codage zonal de DCT. En outre, les composants de DCT peuvent être reconfigurés en vue de réduire la précision des coefficients de DCT particulièrement quand on augmente le paramètre de quantification pour réaliser un taux de compression élevé.

L'application telle que discuté préalablement exige le placement de multiples implémentations d'un même module dans le FPGA. L'utilisation de flot traditionnel de conception de la reconfiguration partielle DPR exigera le stockage d'un bitstream pour chaque implémentation dans la mémoire. Si nous prenons le cas d'implémentation du module DCT fourni dans le tableau 4.2, nous aurons un besoin en mémoire pour le

stockage des bitstreams de 44KB X 8 PRRs. Le temps total de reconfiguration pour les 8 PRR sera de 3.52ms (8x 0.44ms). L'utilisation de la relocation des bitstreams permettra de réduire l'espace de stockage à la taille d'un seul bitstream qui est 44 KB plus quelques mots supplémentaires ajouté au bitstream par notre outil de relocation.

Egalement, l'objectif principal de toutes les méthodologies de relocation proposées, est de réduire le temps de processus de relocation de bitstream avant sa reconfiguration. Si nous prenons les données comparatives du tableau 4.2, nous pouvons observer que le temps de relocation réalisé par les deux versions de BIRF dépasse de loin le temps de configuration pour les 8 modules (51.74 ms et 137.36 ms pour les deux versions HW et SW respectivement comparées à 3.52 ms). La méthode ARC est légèrement meilleure dans cet égard, mais son inconvénient est de traiter seulement avec les captures de configurations par relecture d'ICAP et pas avec les fichiers bitstreams. Dans beaucoup d'applications utilisant la reconfiguration dynamique partielle, le temps de configuration lui-même pourrait être considéré comme pénalité ; donc la présence d'un temps additionnel est exorbitante. Nous pouvons voir que l'outil OORBIT présente un temps de relocation très bas presque négligeable devant le temps de reconfiguration. Une des nouveautés de l'approche est que ceci est réalisé sans aucune incidence sur les ressources matérielles du système ce qui montre bien encore que notre méthodologie a clairement surpassé les approches précédente.

4.4.4 Discussion sur les ressources utilisées

Concernant l'utilisation de ressources, le module en ligne d'OORBIT est exécuté par le processeur et n'exige aucune ressource additionnelle comparée aux autres approches de relocation exécutées en matériel. Notre relocateur a besoin seulement de quelques lignes de programme pour exécuter la relocation du bitstream, qui consiste à substituer quelques registres dans la mémoire (variant de 2 à 23 quel que soit le type du FPGA) par d'autres situés dans la même mémoire et cela sans effectuer aucun traitement. Suivant le tableau 4.3, l'existence du processeur dans le système n'est pas considérée comme une ressource consommée par le relocateur puisqu'il est déjà présent dans toutes les approches de relocation matérielles et logicielles étudiées. Dans ces systèmes reconfigurables de type SoC permettant une auto-reconfiguration, la présence du processeur est essentielle pour

exécuter la partie logicielle de l'application et pour contrôler correctement le processus de reconfiguration.

En comparaison avec les approches de relocation mentionnées dans le tableau 4.3, les performances en termes de vitesse de relocation et de taux d'utilisation des ressources sont largement meilleures en employant notre relocateur. Les ressources employées par les approches matérielles peuvent devenir contraignantes spécialement pour les petits composants de FPGA où la reconfiguration partielle est employée principalement en tant que moyen d'optimisation de ressources. Pour les approches logicielles exécutées par le processeur, contrairement à notre approche, le temps élevé nécessaire pour exécuter la relocation réduira d'une façon significative la vitesse de reconfiguration et par conséquent les performances du système.

Tableau 4.3 Comparaison de ressources utilisées par les approches de relocation

Relocateur	Type	Utilisation de ressources (Virtex 4 FPGA)			Processeur	Vitesse de Reconfiguration (*Relocation & Configuration)*
		LUT	FF	BRAM		
BiRF [54]	SW	0	0	0	Microblaze/PPC	2.6 MB/s
BiRF [53]	HW	555	175	1	Microblaze/PPC	7.3 MB/s
Carver et al. [57]	HW	2047	1574	31	Microblaze	0.32 MB/s
ARC [55]	SW	0	0	0	Microblaze	5.34 MB/s
ARC [56]	HW	1072	686	1	Microblaze	30 MB/s
Online-OORBIT [61]	**SW**	**0**	**0**	**0**	**Microblaze**	**≈ 100 MB/s**

4.5 Conclusion

Dans ce chapitre, Nous avons présenté en détail notre nouvelle méthodologie développée pour la relocation des tâches matérielles que nous avons appelé OORBIT (pour Offline-Online Relocation of Bitstream) et son interaction avec le nouveau flot de conception basé sur le principe des partitions. En comparaison avec les autres approches que nous avons présentées, nous remarquons que notre relocateur en ligne n'a besoin d'aucune ressource matérielle supplémentaire puisque aucun d'autre traitement n'est nécessaire vu que tous les calculs des adresses de configurations ont été déjà effectués en hors-ligne pour les différentes positions possibles des bitstreams. Parallèlement, une

accélération très significative du processus de relocation a été réalisée par notre méthodologie du principalement à la simplicité de l'opération qui consistera dans notre cas à effectuer une sim0ple substitution de valeurs par d'autres pré calculés et présents dans le même fichier bitstream à reloger. Une fois le bitstream est relogé pour la nouvelle partition de destination, il sera immédiatement transmis vers le gestionnaire de configuration pour sa reconfiguration dans la région ciblée. En conclusion, ces exploits réalisées par OORBIT le rend le plus performant dans le domaine.

CONCLUSION

CONCLUSION

La reconfiguration dynamique partielle se présente comme une nouvelle solution technologique de grand intérêt pour les systèmes embarqués. Cette technique offre la possibilité de changement de l'architecture d'une partie de l'application pendant que le reste du système continue son fonctionnement normal. Ceci peut être très utile dans plusieurs cas d'application qui nécessitent des traitements adaptatifs ou une sélection de traitement selon le déroulement de l'application.

Un intérêt très particulier est actuellement consacré à cette technologie et aux opportunités offertes par les nouvelles générations des circuits FPGA. Les travaux dans ce domaine concernent des aspects liés d'une part à l'application de la RDP et d'autre part aux moyens méthodologiques et aux outils permettant de faciliter la mise en œuvre des architectures reconfigurables. Ces architectures, bien qu'elles évoluent d'une façon continue, souffrent encore d'un manque d'efficacité sur le placement des tâches matérielles dans le FPGA. L'idée de base est que des régions spécifiques dans le FPGA peuvent être utilisées pour placer des tâches matérielles à la demande permettant de promouvoir d'une façon efficace la notion de virtualisation des tâches matérielles. Ces dernières sont stockées sous forme de bitstreams précompilés dans une mémoire externe en général où ils peuvent être chargés par le processeur à la demande dans des régions où on veut remplacer leurs missions.. En conséquence, un système d'exploitation destiné aux systèmes reconfigurables doit être présent pour fournir des services pour la gestion des tâches matérielles dans le système. Un service plus avancée de l'OS pourrait être aussi la relocation qui permet à une tâche matérielle déjà placée et configurée d'être relogée dans une autre région de la logique reconfigurable en utilisant d'autres ressources. Le processus de relocation cherche à étendre la plaçabilité d'un bitstream en effectuant des modifications sur son contenu. Après relocation, Cette tâche peut continuer à remplir la même fonction de l'endroit où il a été préempté. Plusieurs approches de relocation ont été proposées dans la littérature et se présentent dans leur majorité comme des extensions du flot traditionnel de conception de la reconfiguration dynamique. Toutefois ces approches présentent quelques inconvénients en relation avec leur implémentation en termes de temps d'exécution et de taux de ressources utilisés par le relocateur en vue d'effectuer les modifications nécessaires pour produire un bitstream relogeable.

Dans cette thèse, nous avons développé une nouvelle méthodologie de relocation que nous avons appelée OORBIT (pour Offline /Online Relocation of Bitstreams) basée sur une manipulation conjointe en hors ligne et en ligne des bitstreams. Pour la phase hors ligne, nous avons développé un outil qui permet en première phase d'analyser le fichier bitstream généré par le flot de conception traditionnel de la reconfiguration partielle. Toutes les adresses de configurations sont identifiées et par la suite toutes les régions relogeables possibles sont spécifiées. En second lieu, nous procédons au calcul de nouvelles adresses de configurations pour chaque allocation possible du bitstream. Toutes ces nouvelles informations liées à la relocation sont ensuite ajoutées au bitstream original sous forme d'un complément de données. Le bitstream modifié supportant la relocation est utilisé durant l'exécution de l'application par notre module développé pour la relocation en ligne qui est exécutée par le processeur. Il consiste principalement à modifier les anciennes adresses de configuration du bitstream original par ceux précalculés en hors ligne correspondants à la nouvelle allocation désirée. Cette solution a permis d'accélérer le processus de relocation d'une façon spectaculaire

Avec OORBIT, le bitstream relogeable réalisé garde son intégrité suite à l'ajout des données de relocation. Il peut être directement exploitable durant une reconfiguration. En plus, les données de relocation ajoutées au bitstream, qui ne dépassent pas au maximum les 186 octets, n'ont aucune influence sur la taille du bitstream initial qui dépasse dans la majorité des cas des dizaines de Kbits. Egalement, le processus de relocation en ligne s'est limité seulement à quelques opérations de substitution d'une vingtaine de mots au maximum. Le temps de relocation est équivalent au temps de lecture et écriture de mémoire de ces mots. En Comparaison avec les autres approches étudiées, on a observé une accélération moyenne avec OORBIT de plus de 3000 fois et 8000 fois par rapport respectivement aux deux versions matérielle et software de l'approche BIRF et une accélération moyenne de plus de 300 fois par rapport à l'approche ARC. En Outre, On a remarqué que les autres approches de relocation étudiées ont dégradé la vitesse de reconfiguration à 7.3 MB/s et à 2.6 MB/s respectivement pour HW-BIRF et SW-BIRF et à moins de 30 MB/s en moyenne pour ARC. Avec OORBIT, le temps de relocation d'un bitstream est presque négligeable comparé à son temps de reconfiguration; il est plus de 100 fois moins en moyenne. La vitesse maximale de reconfiguration d'ICAP qui est de 100MB/s est donc atteinte. Cette grande accélération rend OORBIT le plus performant dans le domaine.

Encore et contrairement aux autres approches, OORBIT ne nécessite aucune ressource matérielle pour l'implémenter. Il requiert uniquement quelques lignes de programme pour réaliser des simples lectures/écritures de mémoire sans effectuer aucun calcul. En plus de ces avantages, notre approche OORBIT supporte tous les FPGA de types 4, 5 et 6 et demeure le seul à notre connaissance qui traite avec le contexte du nouveau flot de conception de la reconfiguration partielle de Xilinx basé sur le concept de partitions. Ce nouveau flot a éliminé quelques contraintes existantes dans les précédents flots qui ont été utilisés pour éviter qu'un routage statique traverse les régions reconfigurables. Par conséquent, des traitements supplémentaires ont été assuré par notre nouvelle approche, lors de la phase de conception, pour empêcher ce routage en vue de garantir l'intégrité du FPGA et de réussir le processus de relocation du bitstream. Ces traitements ont été formalisés et ajoutées à notre méthodologie de relocation.

Une poursuite possible de ces travaux est d'étendre le concept de la relocation pour couvrir encore les tâches logicielles. Actuellement, la relocation consiste à faire migrer une tâche matérielle d'une partition à une autre. Nous envisageons de réaliser la migration des tâches entre le matériel et le logiciel. Pour cela nous devons réaliser deux versions pour chaque tâche dans le système, l'un en software et l'autre en matérielle. Le premier objectif sera d'arriver à basculer le traitement d'une tâche entre les deux supports d'exécution. La problématique qui peut survenir est la manière avec laquelle on doit transférer le contexte de la tâche à migrer entre deux supports dont leurs manières d'exécution sont complètement différentes.

REFERENCES

REFERENCES

1. P. Manet, “An Evaluation of Dynamic Partial Reconfiguration for Signal and Image Processing in Professional Electronics Applications”, EURASIP Journal on Embedded Systems, Vol 2008, (2008).

2. K. Compton, S. Hack, “Reconfiguration Management, Chapter 4 in Reconfigurable Computing The Theory and Practice of FPGA-based Computing”, Morgan Kaufmann, (2008).

3. A. Donlin, “Applications, Design Tools and Low Power Issues in FPGA Reconfiguration”, Chapter 22 in Designing Embedded Processors A Low Power Perspective, (Springer, 2007), 513-541.

4. M. Uhm, “Making the Adaptivity of SDR and Cognative Radio Affordable: Going Beyond flexibility and adaptability in FPGAs”, Xilinx DSP Magazine, (May 2006), 25-27.

5. F. Fons, M. Fons, “Making Biometrics the Killer App of FPGA Dynamic Partial Reconfiguration”, Xcell Magazine, (2010)

6. M. Hübner, J. Becker, “Seamless Design Flow for Run-Time Reconfigurable Automotive Systems”, DATE Workshop on Future Trends in Automotive Electronics and Tool Integration, (March 2006).

7. J. Hagemeyer, B. Kettelhoit, M. Koester, and M. Porrmann, “Design of homogeneous communication infrastructures for partially reconfigurable FPGAs”, in Proc of the International Conference on Engineering of Reconfigurable Systems and Algorithms ERSA ’07, CSREA Press, (2007).

8. A. Oetken, S. Wildermann, J. Teich, D. Koch, “A Bus-based SoC Architecture for Flexible Module Placement on Reconfigurable FPGAs”, (FPL 2010).

9. A. Gupte, P. Jones, “Hotspot Mitigation Using Dynamic Partial Reconfiguration for Improved Performance, Reconfigurable Computing and FPGAs”, International Conference on Reconfigurable Computing and FPGAs, (2009), 89-94.

10. D. Montminy, R. Baldwin, P. Williams, and B. Mullins, “Using relocatable bitstreams for fault tolerance, Adaptive Hardware and Systems”, Second NASA/ESA Conference (August 2007), 701–708.

11. Xilinx Corporation, “Two Flows for Partial Reconfiguration Module Based or Difference Based”, Xilinx XAPP290, (2004).

12. Xilinx Corporation, “Early Access Partial Reconfiguration User Guide”, Xilinx UG208, (2006).

13. Xilinx Corporation, Site web: Http: //www.xilinx.com/.

14. Altera Corporation, Site web: Http: //www.altera.com/.

15. J. Rose, R.J. Francis, D. Lewis, and P. Chow, "Architecture of field programmable gate arrays: the effect of logic block functionality on area efficiency", IEEE Journal of Solid-State Circuits, 25(5), (Oct 1990), 1217–1225

16. I. Kuon and J.Rose, "Quantifying and exploring the gap between FPGAs and ASICs", Springer, (2010).

17. Katherine Compton and Scott Hauck, "Reconfigurable computing: a survey of systems and software", ACM Computer Survey., 34(2),(June 2002), 171–210

18. A. Donlin, "Chapter 22: Applications, design tools and low power issues in FPGA reconfiguration", Springer Netherlands, (2007), 451–501

19. Xilinx Corporation, "Command Line Tools User Guide", Xilinx UG628 , (2012).

20. Xilinx Corporation, "Embedded System Tools Reference Manual", Xilinx UG111, (2012).

21. J. Becker et al., "Dynamic and partial FPGA exploitation", Proceedings of the IEEE, 95(2) , (2007), 438–452

22. Christopher Claus, Rehan Ahmed, Florian Altenried, and Walter Stechele, "Towards rapid dynamic partial reconfiguration in video-based driver assistance systems". In Proceedings of the 6th international conference on Reconfigurable Computing: architectures, Tools and Applications, ARC'10, Springer, Berlin, (2010), 55–67

23. J. Strunk, T. Volkmer, K. Stephan, Wolfgang Rehm, and H. Schick, "Impact of runtime reconfiguration on design and speed, a case study based on a grid of runtime reconfigurable modules inside a FPGA". In Parallel Distributed Processing, IPDPS 2009. IEEE International Symposium, (May 2009), 1-8

24. Christophe Bobda, "Introduction to Reconfigurable Computing: Architectures, Algorithms, and Applications", Springer Publishing Company, Incorporated, 1st edition,(2007)

25. Ting LIU, "Optimisation par synthèse architecturale des méthodes de partitionnement temporel pour les circuits reconfigurables", Thèse de doctorat, Université Henri Poincaré, (Novembre 2008).

26. P. Bertin," Mémoires actives programmables : conception, réalisation et programmation", Thèse de doctorat, Université de Paris 07, (1992).

27. J.E. Vuillemin et al., "Programmable active memories: reconfigurable systems come of age", Very Large Scale Integration (VLSI) Systems, IEEE Transactions on, 4(1), (March 1996), 56-69

28. M. Franzmier et al., “Hardware accelerated data analysis”, In Parallel Computing in Electrical Engineering, PARELEC 2004, International Conference on, (Sept 2004), 309-314

29. K.M.G. Purna et D. Bhatia, “Temporal partitioning and scheduling data flow graphs for reconfigurable computers”, IEEE Computers, 48(6), (June 1999). 579 –590

30. H. AMANO, “A survey on dynamically reconfigurable processors”, IEICE Transactions on Communications, 89(12), (2006)

31. K. Compton, S. Hauck, “Reconfigurable Computing: A Survey of Systems and Software”, ACM Computing Surveys, Vol. 34, No. 2, (June 2002), 171-210.

32. Patrick Lysaght, On Blodget and Jeff Mason, “Invited paper: Enhanced architectures, design methodologies and cad tools for dynamic reconfiguration of Xilinx FPGAs”, Thilo (1), (2006), 1-6

33. Xilinx Corporation, “Virtex-6 FPGA Configuration User Guide”, Xilinx UG360, (2010).

34. Xilinx Corporation, “Virtex-5 FPGA Configuration User Guide”, Xilinx UG091, (2010).

35. Xilinx Corporation, “Development System Reference Guide”, chapter 4: Modular Design. (2002).

36. M. Huebner, T. Becker and J. Becker, “Real-time LUT-based network topologies for dynamic and partial FPGA self-reconfiguration”, In Proceedings of the 17th symposium on Integrated circuits and system design, SBCCI’04, ACM NY, USA, (2004), 28–32

37. Xilinx Corporation, “Partial Reconfiguration User Guide”, Xilinx UG702, (2012)

38. N. Marques et al., “Partially reconfigurable entropy encoder for multi standards video adaptation”, In Consumer Electronics (ISCE’2011), IEEE 15th International Symposium on, (June 2011), 492-496.

39. F. Duhem et al, “Multi-standards video adaptation using a fast reconfiguration manager”, Microprocessors and Microsystems journal (MICPRO), (2012).

40. Xilinx Corporation, “PlanAhead User Guide”, Xilinx UG632, (2012).

41. Xilinx Corporation, “ISE in depth Tutorial”, Xilinx UG695, (2012).

42. S. Neuendorffer, “Chapter 12: FPGA Platforms for Embedded Systems”, Morgan Kaufmann, (2010).

43. O. Blodget et al., “A self-reconfiguring platform”, In Proceedings of Field Programmable Logic and Applications, (2003), 565-574.

44. O. Blodget et al., “A light weight approach for embedded reconfiguration of FPGAs”. In Design, Automation and Test”, in Europe Conference and Exhibition, (2003), 399-400.

45. H. Kwok-Hay So and R.W. Brodersen, “Improving usability of FPGA-based reconfigurable computers through operating system support”, In Field Programmable Logic and Applications, FPL ’06, International Conference on, (Aug 2006), 1-6

46. M.D. Santambrogio, V. Rana, and D. Sciuto,”Operating System support for online partial dynamic reconfiguration management”, In Field Programmable Logic and Applications, FPL’2008. International Conference on, (Sept 2008), 455-458

47. Xilinx Corporation, “EDK Concepts, Tools, and Techniques”, Xilinx UG683, (2012).

48. Xilinx Corporation, “LOGICORE IP XPS HWICAP”, Xilinx v5.00a, (July 2010).

49. J. Huang, J. Lee, Y. Ge, “An array-based scalable architecture for DCT computations in video coding”, In the Proceedings of the International Conference on Neural Networks and Signal Processing, (June 2008), 451-455.

50. J. Huang, M. Parris, J. Lee, R. F. DeMara, “Scalable FPGA Architecture for DCT Computation using Dynamic Partial Reconfiguration”, In the Proceedings of the International Conference on Engineering of Reconfigurable Systems and Algorithms ERSA, Las Vegas, U.S.A, (July 2008).

51. Xilinx Corporation, “Virtex-4 FPGA Configuration Guide”, Xilinx UG071, (2009).

52. H. Kalte and M. Porrmann, “REPLICA2PRO: Task Relocation by Bitstream Manipulation in Virtex-II/Pro FPGA”s, in Proceeding of the 3rd Conference on Computing Frontiers, (2006). 403–412

53. S. Corbetta, M. Morandi, M. Novati, M. Santambrogio, D. Sciuto, and P. Spoletini, “Internal and external bitstream relocation for partial dynamic reconfiguration”, Very Large Scale Integration (VLSI) Systems, Vol. 17, no. 11, IEEE Transactions, (Nov 2009), 1650-1654.

54. S. Corbetta, F. Ferrandi, M. Morandi, M. Novati, M. D. Santambrogio, and D. Sciuto, “Two novel approaches to online partial bitstream relocation in a dynamically reconfigurable system”, in Proc. IEEE Computer. Society, (May 2007), 457-458.

55. R. Kallam, A. Sudarsanam, and A. Dasu, “Accelerated Relocation Circuit”, IET Electronics Letters, (2009)

56. A. Sudarsanam, R. Kallam, and A. Dasu, “PRR-PRR Dynamic Relocation”, IEEE Computer Architecture Letters, Vol. 8, No. 2, (Jul-Dec 2009).

57. J. Carver, R. Pittman, and A. Forin, “Relocation and Automatic Floorplanning of FPGA Partial Reconfiguration Bitstreams”, Microsoft Research, Technical Report no.MSR-TR-2008-111, (Aug 2008).

58. T. Becker, W. Luk, and P. Cheung, “Enhancing Relocatability of Partial Bitstreams for Run-time Reconfiguration”, Field Programmable Custom Computing Machines, 15th Annual IEEE Symposium, (Apr 2007), 35–44.

59. Xilinx Corporation, “Constraints Guide”, Xilinx UG625, (2010).

60. A. Sreeramareddy, R. Kallam , A. R. Dasu, A. Akoglu, “Self-reconfigurable architecture for reusable systems system with Accelerated Relocation (SCARS-ARC)”, Parallel & Distributed Processing, Workshops and PhD Forum (IPDPSW’10), IEEE International Symposium , (2010).

61. Maamar Touiza, Gilberto Ochoa, El-Bay Bourennane , Abderrezak Guessoum, Kamel Messaoudi “A Novel Methodology for Accelerating Bitstream Relocation in Partially Reconfigurable Systems”, Journal of Microprocessors and Microsystems (MICPRO), Hardware Embedded Systems, Elsevier Edition, in press, (July 2012), http://dx.doi.org/10.1016/j.micpro.2012.07.004

APPENDICES