Partitionnement d'applications à temps d'exécution variables sur architectures reconfigurables.

GHAFFARI Fakhreddine *[&] ; AUGUIN Michel* ; BENJEMAA Maher****

**Laboratoire d'Informatique, Signaux et Systèmes de Sophia-Antipolis, les Algorithmes bat. Euclide, 2000, route des Lucioles BP 121, 06903 Sophia-Antipolis Cedex*

***Laboratoire de recherche ACEM. Ecole Nationale d'Ingénieurs de Sfax, BPW 3038 Sfax Tunisie.*

Ghaffari@i3s.unice.fr, auguin@i3s.unice.fr, Maher.Benjemaa@enis.rnu.tn

1. Introduction

L'analyse de mouvement dans les séquences d'images est un domaine de recherche active à l'heure actuelle en raison de son importance dans de nombreuses applications : télésurveillance, compression pour les télécommunications ou l'archivage, diagnostic médical, météorologie, contrôle non destructif, robotique mobile, etc. Cependant, ces applications requièrent des puissances de calcul adaptées.
Le problème s'accentue lorsque nous voulons travailler sur des systèmes embarqués qui seront soumises à des contraintes de temps réel, de consommation de surface,…

Intégrer une telle puissance de traitement dans un système embarqué implique généralement des architectures hétérogènes parallèles pour rechercher les meilleurs compromis performances / taille / consommation. Dans ce cas, des nouvelles méthodologies de conception qualifiées de conception conjointe logiciel/ matériel (Codesign) sont nécessaires. L'approche codesign consiste alors à définir l'ensemble des sous tâches d'une application à intégrer et à effectuer leur répartition sur des cibles logicielles ou matérielles. L'intérêt majeur de cette méthodologie réside dans la recherche d'une adéquation application / architecture satisfaisant les nombreuses contraintes de conception telles que le coût, les performances, la surface, la consommation, les temps de conception et de développement (Time To Market), l'évolutivité,… La conception efficace de ces systèmes hétérogènes nécessite une approche globale dans laquelle les parties matérielles et logicielles sont conçues en parallèle et de façon interactive [3rd]. Une des phases clés de l'approche Codesign est le partitionnement logiciel/ matériel ; qui consiste à distribuer les fonctionnalités d'une spécification système sur un ensemble des unités d'une architecture cible en effectuant le choix pour une implémentation logicielle, matérielle ou mixte des différentes parties du système.
Face à la complexité du problème de partitionnement logiciel / matériel, plusieurs approches ont adopté des méthodes manuelles pour affecter chaque tâche vers l'entité correspondante sur l'architecture. D'autres préfèrent que le partitionnement soit semi-automatique, il s'agit de combiner à la fois la conception manuelle et la conception automatique. L'approche dans ce cas, suppose que le concepteur commence la synthèse à partir d'une spécification initiale des fonctions d'un système et ayant en tête une solution d'architecture. Toutes les transformations de raffinement sont réalisées de manière automatique mais toutes les décisions de raffinement sont prises par le concepteur qui peut utiliser son savoir faire et son expérience pour converger vers une solution efficace.
La troisième catégorie d'approche est celle qui utilise une méthode automatique ; dans ce cas un algorithme d'optimisation, qui prend en compte tous les paramètres du problème, sera adopté. Souvent des heuristiques seront privilégiées dans ce dernier cas.

Dans la suite, nous citons quelques méthodes de partitionnement qui sont adoptées dans la littérature et qui sont manuelles, semi-automatique ou automatique.
L'approche utilisée à l'Université de Californie / Berkeley sous l'environnement Ptolemy considère un partitionnement manuel. Ce partitionnement est guidé par les contraintes de surface, de vitesse et de flexibilité. Le concepteur essaie d'optimiser des fonctions coûts telles que : coût des communications, espaces mémoires locales et globales, etc. [5rd] [6rd] [7rd] [8rd] [10rd] [9rd] [11rd].
Dans [12rd] [13rd] [14rd] [16rd] l'approche de partitionnement commence avec une partition initiale où toutes les opérations, excepté celles à délai non limité, sont affectées au matériel. La partition est raffinée par la migration d'opérations du matériel vers le logiciel afin d'obtenir une partition à moindre coût. Une autre approche de partitionnement est adoptée à l'Université de Linkoping [28],

elle est formulée comme un problème de partitionnement de graphes. Elle réalise la transposition d'un graphe de flux de contrôle et de données sur un graphe unique de partitionnement. Différents types d'arcs sont utilisés entre les nœuds pour refléter les dépendances. Nous citons aussi la méthodologie de l'Université de Braunschweig dans le système COSYMA [20rd] [30rd] [31rd]. L'outil de partitionnement logiciel / matériel, dans ce cas, est automatique et basé sur un algorithme de recuit simulé. La stratégie adoptée pour le partitionnement logiciel / matériel consiste à démarrer d'une solution purement logicielle ensuite à extraire les parties critiques pour les réaliser en matériel en exploitant les informations obtenues par analyse de performances. Le système de partitionnement vérifie si les contraintes sur le temps sont satisfaites.

Nous terminons notre étude bibliographique sur les méthodes de partitionnement logiciel / matériel, par des approches récentes ; celles qui utilisent une théorie de clustering par exemple [32rd][33rd][25rd] pour privilégier le regroupement sur une même unité des fonctions ou tâches qui communiquent. Cette technique permet de minimiser le temps d'exécution total en annulant les temps de communication entre les tâches regroupées dans un même cluster.

Toutefois, toutes ces approches considèrent que les fonctions de l'application possèdent des temps de calcul constants, pour un nombre de ressources fixé. Or, de nombreuses applications, en particulier en traitement des images, font apparaître des charges de calcul variables en fonction des images à traiter.

La détection de mouvement sur un fond d'image fixe est utilisée dans des caméras intelligentes embarquées. Une telle application fait appel à des séquences d'opérations de traitement d'images. Parmi ces opérations, nous distinguons celles qui conservent un temps d'exécution fixe d'une image à une autre, de celles dont les temps de calcul varient suivant la nature des images.

C'est sur ce dernier type d'applications que nous focalisons notre étude et sur lesquelles il s'agit de tenir compte de la distribution variable de la charge de calculs lors du partitionnement logiciel / matériel des tâches.

Ce papier est organisé en quatre paragraphes, l'architecture cible est présentée dans le paragraphe deux, ensuite introduit la modélisation retenue des applications dans le paragraphe trois. Notre approche de partitionnement est détaillée dans le paragraphe quatre. Enfin nous terminons par une conclusion dans le paragraphe cinq.

2. Modèle d'application et architecture cible

Dans le projet EPICURE[1] l'architecture considérée est constituée d'un processeur connecté à un circuit reconfigurable dynamiquement à travers une interface générique comme le montre la figure1. Ce type d'architecture est bien adapté pour concevoir des systèmes embarqués de type caméra intelligente. La flexibilité du reconfigurable permet d'adapter les traitements à l'environnement dans lequel est placée la caméra et la reconfiguration dynamique autorise une meilleur exploitation des ressources matérielles et donc de diminuer la surface de silicium. Pour expérimenter nos travaux nous considérons le système EXCALIBUR d'Altera qui intègre dans un même circuit un processeur et une unité reconfigurable [8].

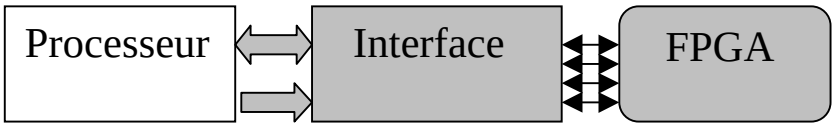


Figure 1 : système reconfigurable à interface externe

Les applications de traitement des images peuvent aisément se modéliser par des graphes de flots de données, au moins dans les niveaux bas et moyen de traitement. Chaque opération de traitement est

[1] Ce projet est financé par le Ministère français de la Recherche dans le cadre du Réseau National des Technologies Logicielles RNTL. Les partenaires de ce projet sont le CEA, Thales, Esterel Technologies, Université de Bretagne Sud (LESTER), Université de Nice Sophia Antipolis/CNRS (I3S).

modélisée par un nœud du graphe et les communications entre les opérations sont représentées par les arcs. Par exemple, nous pouvons modéliser notre application détection de mouvement sur un fond d'image fixe par le graphe de flots de données schématisé sur la figure 2.

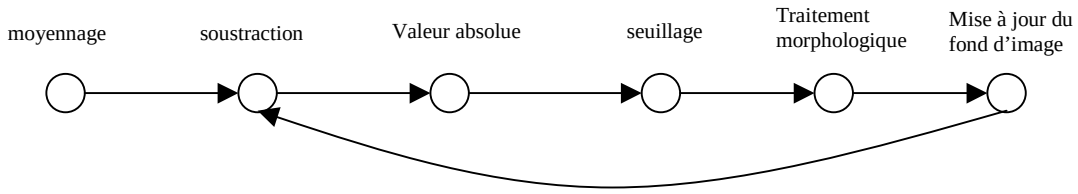


Figure 2 : Exemple du graphe de flots de données

3. Approche de partitionnement considérée

En traitement d'image de nombreuses opérations ont un temps d'exécution fortement corrélé au contenu de l'image traitée. Nous pouvons donc, considérer deux types de tâches : les tâches qui gardent un temps d'exécution constant pour les différents jeux de données et celles qui ont un temps d'exécution variable.

Pour développer une méthode de partitionnement d'applications contenant des tâches de ce dernier type, il faut tout d'abord définir la nature de la corrélation entre le temps d'exécution et les caractéristiques des données mises en jeu.

Le paramètre de corrélation est identifié par une étude préalable de la tâche concernée, suivie d'une confirmation à l'aide d'essais pratiques. Par exemple, dans l'application de détection de mouvement sur un fond d'image fixe, nous pouvons définir les paramètres de corrélation pour les tâches qui possèdent des temps d'exécution variables comme indiqué sur le tableau suivant :

Tâche	Moyennage	Reconstruction	Etiquetage	Enveloppe englobant	Centre de gravité	Test sur le déplacement
Paramètres de corrélation	Nombre des images	Taille totale des objets	Taille totale des objets	Nombre des objets	Nombre et taille des objets	Nombre des objets

En effectuant plusieurs mesures de temps d'exécution sur le processeur et sur un FPGA nous pouvons caractériser ces tâches en fonction des paramètres de corrélation. A titre d'exemple, la tâche de test sur les déplacements des objets [9] et [10], possède un temps d'exécution qui dépend du nombre d'objets en mouvement dans l'image. Si nous désignons par n le nombre d'objets dans l'image et si nous mesurons le temps d'exécution de la tâche sur cette image nous obtenons des points comme indiqué sur la figure 3.

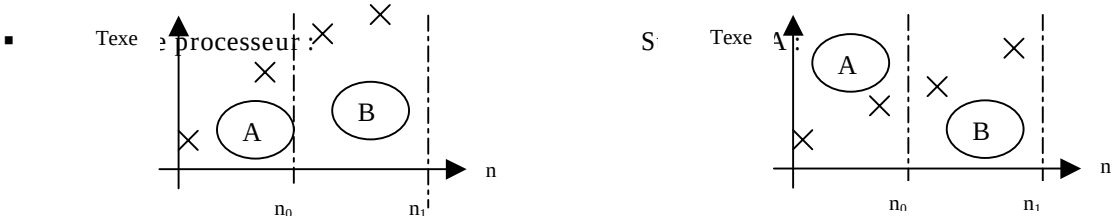


Figure 3 : courbes du temps d'exécution en fonction du nombre d'objets dans l'image

A partir de ces ensembles de points nous pouvons classer les images dans différentes catégories (A et B par exemple). Pour chaque catégorie nous allons attribuer un temps d'exécution qui sera le même

pour tous ses éléments. Ce temps d'exécution peut être le maximum des temps de l'ensemble des points qui appartiennent à la même catégorie dans le but d'obtenir des exécutions qui satisfassent les contraintes dans le pire des cas. La contrainte sur cette classification est que les seuils définis sur un paramètre de corrélation soient identiques pour le processeur et le FPGA. Nous pouvons alors construire un graphe de données conditionné dans lequel chaque tâche, à temps d'exécution variable, est répliquée autant de fois qu'il y a de catégories identifiées pour cette tâche (figure 4).

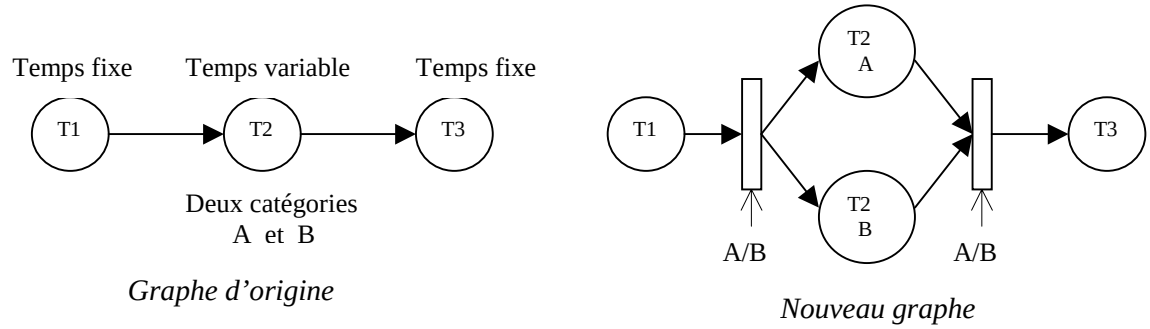


Figure 4 : construction d'un graphe conditionné

A partir de ce gr des configurations possibles (figure 5) qui sont autant de graphes de flots de données non conditionnées [11].
Pour que notre approche ne conduise pas à une explosion combinatoire des configurations possibles, nous avons considéré un niveau de granularité assez élevé tout en cherchant un compromis entre la précision des valeurs de temps d'exécutions choisies pour les seuils et le nombre de configurations à partitionner.

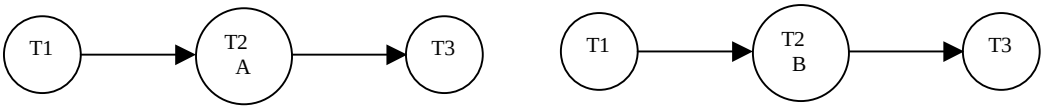


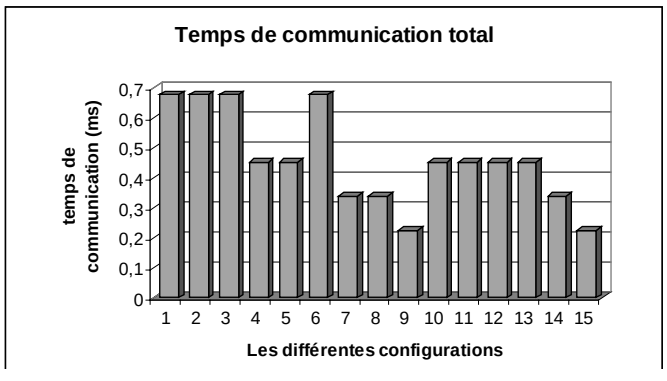
Figure 5 : Les deux configurations du graphe conditionné de la figure 4

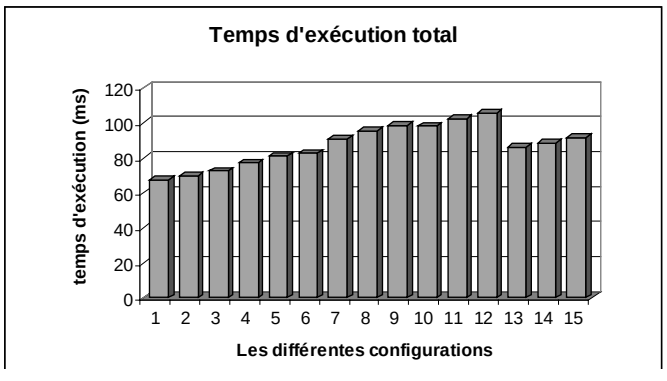
Sur chaque graphe ainsi obtenu nous appliquons un algorithme de partitionnement pour obtenir autant de contextes qui sont mémorisés dans l'architecture pour programmer le processeur et le FPGA en fonction des images traitées.
Dans notre approche nous utilisons un algorithme génétique pour effectuer le partitionnement [12].
Une fois les différents contextes obtenus à partir des configurations identifiées, il est possible de construire le flot de contrôle de l'application sur l'architecture. Ce flot de contrôle doit activer les traitements qui correspondent aux contextes définis par le partitionnement.

Au moment de l'exécution de l'application, nous utilisons donc comme valeur de contrôle les paramètres de corrélation cités précédemment pour pouvoir identifier le graphe de tâches à considérer et par la suite quel contexte il faut appliquer au FPGA et au processeur.

Ce critère de contrôle peut être le paramètre de corrélation lui même comme il peut être une combinaison de plusieurs paramètres. En régime permanent nous pouvons calculer le critère de contrôle pour l'image (i) à partir de l'image (i-1) déjà traitée ou bien par une pondération sur les paramètres des images qui les précèdent.

4. Résultats et analyses





5. Conclusion

L'approche présentée ci-dessus nécessite d'être validée. Il se pose par exemple le problème du choix des séquences de test pour caractériser les comportements de l'application qui induisent les configurations retenues pour le partitionnement. Ces configurations doivent être représentatives dans le but de garantir que les exécutions du système, sur les images réelles vérifient les contraintes. Si nous nous limitons à un cas particulier de traitement tel que la surveillance d'un parking de voitures, le choix d'une séquence de test représentative peut être aisé à déterminé. Par contre dans un environnement plus générale où il est plus difficile de caractériser les scènes analysées, il faut penser à un partitionnement dynamique en fonction de la nature des données. Une telle approche sera considérée dans nos futurs travaux.
Un degré de liberté non exploité dans l'approche exposée concerne la possibilité de considérer

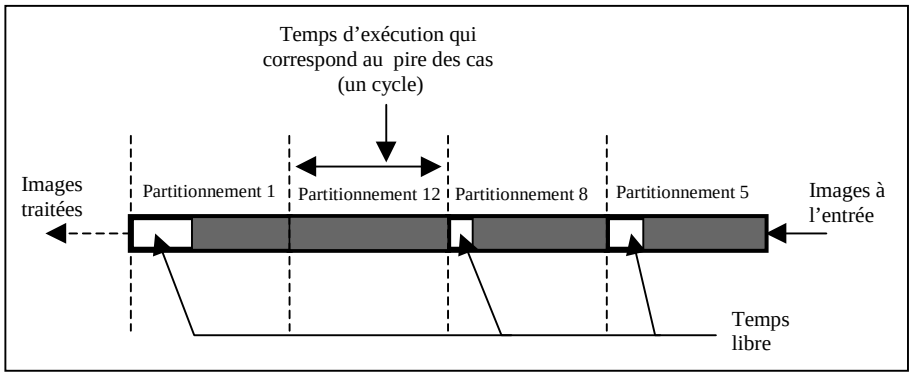


Figure 5.4 : Principe du temps libre pendant l'exécution de l'application

différents degrés de parallélisme dans les mises en œuvre des tâches sur le FPGA dans le but d'adapter dynamiquement la configuration du matériel aux besoins de traitement.
La méthode de partitionnement devrait prendre en compte cette possibilité.

Remerciement : Les auteurs tiennent à remercier M.Christian GAMRAT du CEA pour les fructueuses discussions et ses nombreux conseils en particulier sur l'application de traitement d'image.

6. Références

[1] M, Abid, Contribution à la conception des systèmes mixtes logiciel/matériel. thèse d'état présentée à l'ENIT (TUNISIE), mai 2000.

[2] A. Kalavade, E. Lee, Global criticality/local phase driven algorithm for the constrained hardware/software partitioning problem, International Workshop on Hardware/Software Codesign, pages (293- 311), septembre, 1994.

[3] R.Ernst , J.Henkel, et T.Benner. Hardware-software Cosynthesis for Microcontrollers. IEEE Design et Test, vol.12, pages 64-75, 1993.

[4] R.K.Gupta, C.Coelho, and G.De Micheli. Synthesis and Simulation of digital systems containing Interacting Hardware and Software Components. 29th ACM, IEEE Design Automation Conference, pages 225-230,1992.

[5] R.P.Dick and N.K.Jha. MOGAC : A Multiobjective Genetic Algorithm for the Co-Synthesis of Hardware-Software Embedded Systems, IEEE Trans. Computer-Aided Design Integrated Circuits & Syst, vol.17, no.10, pp.920—935, Oct. 1998.

[6] R.Niemann, P.Marwedel. Hardware/Software partitioning using integer programming. European Design and Test conference 1996.

[7] M.Kaul, R.Vemuri, S.Govindarajan, I.Ouaiss. An automated temporel partitioning and loop fission approach for FPGA based reconfigurable synthesis of DSP applications Design Automation Conference, New Orleans, LA, 1999.

[8] Nios Embedded Processor Getting Started Version 1.1 user Guide, ALTERA March 2001.

[9] E.Duchesene rapport de stage au CEA LIST : détection de mouvement sur un fond d'image fixe , 2001.

[10] M. Peythieux. Centaure : une architecture parallèle hétérogène pour la vision par ordinateur. Rapport CEA-R-5788, 1998.

[11] M.Auguin, L.Bianco, L.Capella, E.Gresset. Conception de systèmes embarqués par partitionnement de spécifications flots de données conditionnel, Conférence Architectures Nouvelles de Machines, Sympa'6, Besançon, 19-21 juin,2000.

[12] K.Ben Chehida, Partitionnement Logiciel/Matériel pour des architectures reconfigurables utilisant une approche génétique, Rapport de stage de DEA université de Nice Sophia-Antipolis, 2000/2001.

[13] K.Bazargan and M.Sarrafzadeh. Fast Online Placement for Reconfigurable Computing Systems. In Proceedings of IEEE Symposium on FPGAs for Custom Computing Machines, 1999.

[14] T.J. Callahan, P.Chong, A.DeHon, and J.Wawrzynek. Fast Module Mapping and Placement for Datapaths in FPGAs. In International ACM/SIGDA Symposium on Field Programmable Gate Arrays, February 1998.

[15] I.Ouaiss, S. Govindarajan, V. Srinivasan, M. Kaul, and R. Vemuri. An Integrated Partitioning and Synthesis System for Dynamically Reconfigurable Multi-FPGA Architectures. Fifth Reconfigurable Architectures Workshop, 1998.