Ordonnancement optimisé pour architecture hétérogène CPU/FPGA

CAMEL TANOUGAST[1], CAMILLE DIOU[1], LOIC SIELER[1]

[1] Laboratoire de Conception, Optimisation et Modélisation des Systèmes
ISEA, 7 rue marconi, 57070 Metz, France

[1]Camel.Tanougast@univ-lorraine.fr

Résumé - Nous proposons un modèle de programmation linéaire pour le problème d'ordonnancement de tâches dans une architecture hétérogène CPU/FPGA prenant en compte le parallélisme intrinsèque FPGA et les délais de communication hétérogènes. L'objectif est d'exécuter dans une structure MPSoC, une application décrite sous forme d'un graphe de flot de données en cherchant à minimiser la longueur de l'ordonnancement Cmax (makespan). Les résultats de simulation montrent que le modèle proposé fournit des solutions optimales d'ordonnancement de tâches tout en améliorant significativement le temps d'exécution par rapport à des travaux similaires.

Abstract - This article studies the scheduling problem of a CPU/FPGA heterogeneous architecture (Multi-Processors System on Chip) with the parallel processing FPGA and heterogeneous communication delays. We propose a linear programming model (exact or mathematic model) to run in the MPSoC an application described in the data flow graph form in order to minimize the scheduling length Cmax (makespan). Computational experiments are conducted to evaluate their performance. The obtained results reveal that proposed model provides optimal solutions for the scheduling task problems while improving significantly the running time compared to similar works.

1 Introduction

La complexité des applications de traitement du signal et des images, et le besoin de flexibilité des systèmes ont conduit à l'émergence d'architectures hétérogènes reconfigurables sur puce. L'objectif principal de ces architectures de calcul est de répondre aux différents besoins applicatifs en combinant les structures logiques et logicielles afin d'atteindre les meilleures performances (adéquation algorithme architecture) en optimisant le temps de calcul ou d'exécution. Les problèmes de placement et d'ordonnancement minimisant la longueur de l'ordonnancement ou le temps d'exécution maximal (c'est-à-dire le makespan) de tâches dans les architectures hétérogènes reconfigurables MPSoC (*Multi-Processor System on Chip*) composées de processeurs (*CPUs, Central Processing Unit*) et de blocs matériels configurables FPGA (*Field Programmable Gate Array*) reliés par une structure de communication, sont des problèmes importants et NP-difficiles [1].

Différents outils et modèles existent pour résoudre les problèmes d'ordonnancement homogènes à *m* machines parallèles (identiques, uniformes et non liées) pour minimiser le makespan [2]. Généralement, la résolution des problèmes d'ordonnancement et de placement repose principalement sur deux catégories de méthodes d'optimisation combinatoire. On trouve les méthodes exactes (ou complètes) qui garantissent l'exhaustivité de la résolution, et les méthodes approximatives (ou incomplètes) qui perdent leur intégralité pour gagner en temps de calcul de la solution. Les méthodes exactes comprennent la programmation dynamique (PD), la programmation linéaire (PL) et la méthode de *branch and bound* (méthode par évaluation et séparation). Les méthodes approximatives sont principalement des méthodes heuristiques basées sur une solution ou une population de solutions (recuit simulé, recherche tabou, algorithme génétique) [1, 3, 4]. Le tableau 1 compare ces méthodes.

Tab 1 : Comparaison entre les méthodes exactes et approximatives

Méthodes exactes	Méthodes approximatives
Solutions optimales : +	Pas de garantie d'optimalité : -
Pas de garantie sur le temps de calcul (exponentielle) : -	Temps de calcul polynomial : +

Pour résoudre les problèmes d'ordonnancement dans les architectures hétérogènes reconfigurables, un modèle mathématique a été proposé afin d'optimiser l'ordre des calculs en utilisant la programmation linéaire (LP) qui peut être résolue par un solveur de programmation linéaire CPLEX [5, 6]. Cependant, bien que ce modèle spécifique prenne en compte les délais de communication dans la structure d'implémentation hétérogène, il n'intègre pas le parallélisme potentiel des tâches dans les blocs configurables MPSoC. Ainsi, une linéarisation est

uniquement effectuée sur les contraintes de communication pour obtenir un modèle linéaire sans aucune autre variable.
Dans cet article, nous proposons un modèle amélioré de programmation linéaire pour l'ordonnancement dans une architecture hétérogène et exploitant la parallélisme intrinsèque FPGA pour optimiser le makespan (Cmax).
Cet article est organisé comme suit. La section 2 présente la formulation du problème d'ordonnancement et l'architecture MPSOC considérée dans ces travaux. La section 3 décrit le modèle existant et le modèle amélioré. La section 4 une comparaison entre les résultats des deux modèles. Une conclusion et des perspectives de poursuite de ces travaux clôture cet article.

2 Description et formulation du problème d'ordonnancement

Pour ces travaux, nous considérons comme travail de référence, la résolution du problème d'ordonnancement dans les architectures hétérogènes reconfigurables proposé dans [5], et basée sur une résolution exacte (méthode mathématique) par programmation linéaire (LP). La figure 1 présente le type d'architecture considéré dans ces travaux. Cette architecture est un système sur puce composé d'un ensemble de processeurs génériques CPUs avec une mémoire partagée et d'un circuit FPGA reliés par un bus communication.

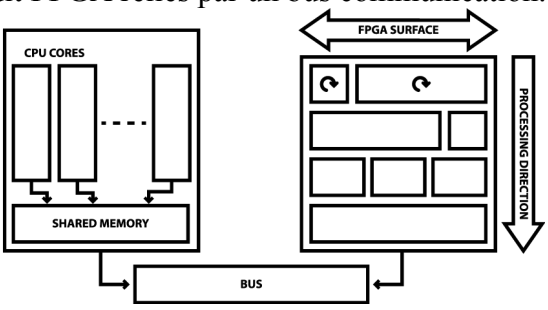

Fig 2 : Architecture hétérogène CPU/FPGA.

Une description globale du problème d'ordonnancement considéré est décrite par la figure 2 correspondant à un graphe de tâches interdépendantes, où sont visualisés l'ordre et la priorité d'exécution des tâches, le temps de traitement de chaque tâche sur chaque processeur différent, le coût de communication entre les différentes tâches et le débit de communication en fonction des différents processeurs disponibles, et enfin l'occupation surfacique de la tâche (lorsqu'elle est planifiée dans le FPGA).

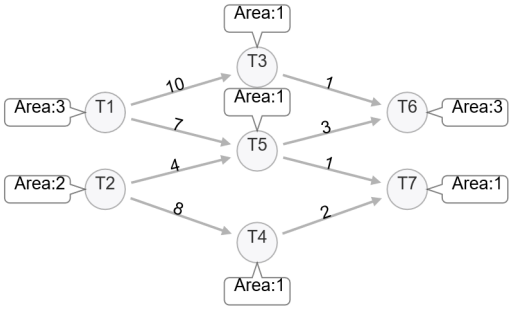

Fig 2 : Graphe de tâches avec les coûts de communication entre les tâches et occupations surfacique FPGA.

Les tables 2, 3 et 4 indiquent respectivement le temps d'exécution de chaque tâche sur chaque unité de calcul (CPU, FPGA), les temps d'interconnexion et le débit de communication entre les différents processeurs/FPGA de l'architecture.

Tab 2 : Temps d'exécution des tâches.

Tâches	CPU1	CPU2	CPU3	FPGA
T1	14	17	15	12
T2	13	16	10	9
T3	11	31	14	5
T4	12	21	15	16
T5	13	21	31	18
T6	7	6	16	3
T7	18	21	19	18

Tab 3 : Temps d'interconnexion entre les unités de calcul.

	CPU1	CPU2	CPU3	FPGA
CPU1	1	3	3	4
CPU2	3	1	3	4
CPU3	3	3	1	4
FPGA	4	4	4	1

Tab 4 : Débit de communication entre les unités de calcul.

	CPU1	CPU2	CPU3	FPGA
CPU1	1	2	2	1
CPU2	2	1	2	1
CPU3	2	2	1	1
FPGA	1	1	1	2

La résolution de ce problème d'ordonnancement, proposé dans [5], prend en compte les délais d'intercommunication entre les unités de calcul (CPU/ FPGA) en considérant l'exécution d'une seule tâche à la fois dans une unité de calcul. La figure 3 donne les résultats d'ordonnancement obtenus et correspondant à des valeurs de makespan de 54,5 et 59.

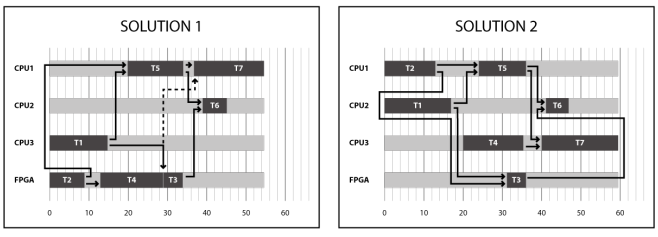

Fig 3 : Solutions d'ordonnancement proposée dans [5].

3 Modèle LP amélioré

Nous proposons une amélioration du modèle de programmation linéaire décrit dans [5], en intégrant le parallélisme potentiel FPGA. La résolution d'ordonnancement tient compte à la fois des délais d'intercommunication entre les unités de calcul CPU et FPGA et de l'exécution d'une seule ou de plusieurs tâches à la fois dans le cas du parallélisme potentiel FPGA, qui peut exécuter plusieurs tâches en parallèle tout en respectant sa contrainte surfacique. Le modèle proposé décrivant le problème d'ordonnancement d'architecture CPU/FPGA hétérogène avec la possibilité d'utiliser efficacement l'attribut de parallélisme FPGA est le suivant :

- **N:** ensemble de *n* tâches ;
- **M:** ensemble de *m* unités de traitement (CPUs / FPGAs) ;
- **G (N, A):** graphe acyclique dirigé, où N est l'ensemble des tâches, A l'ensemble des arcs representant la dépendance entre les tâches, i.e. *(i, j)* dans A signifie que la tâche *i* doit être exécutée avant la tâche *j* ;
- **Pred (i):** ensemble de tâches qui précèdent la tâche i ;
- N^+ **:** ensemble de tâches sans successeurs (sorite de graphe G) ;
- **P(*i*):** ensemble de tâches *j* tel que *i* et *j* appartiennent au même chemin ;
- **T*ik:*** *temps d'exécution de la* tâche *i* dans l'unité de calcul *k* ;
- **C*ik,jl:*** coût direct de communication entre la tâche *i* dans l'unité de calcul *k* and la tâche *j* dans l'unité de calcul *l* ;
- **F*k:*** ensemble de tâches qui ne doivent pas être assignées dans l'unité de calcul *k*.

Le coût d'interconnexion est défini par :

$$c_{ik,jl} = a_{kl} + \frac{d_{ij}}{r_{kl}}$$

Où a_{kl} est le coût fixe de communication entre les unités de calcul *k* et *l*, d_{ij} est la taille des données envoyées de la tâche *i* vers la tâche *j*. r_{kl} est le débit de communication entre les deux unités de calcul *k* et *l*.

Nous introduisons dans notre modèle le minimum interdit suivant :

$$\varphi_{ijk} = \begin{cases} 1 \; if \; tasks \; i \; and \; j \; are \; executed \; in \; parallel \; on \; processor \; k \\ 0 \; else \end{cases}$$

$$Minimize \; C_{max}$$

Toutes les tâches doivent être exécutées dans une unité de calcul

$$\sum_{k \; in \; M} x_{ik} = 1 \quad \forall i \; in \; N$$

Calcul de Cmax à partir de la dernière task (N+)

$$s_i + \sum_{k \; in \; M} t_{ik} x_{ik} \leq C_{max} \quad \forall i \; in \; N^+$$

...

L'ordonnancement respecte les contraintes de précédence et de temps de communication

$$s_i + t_{ik} x_{ik} + c_{ikjl}(x_{jl} + x_{ik} - 1) \leq s_j \quad \forall k, l \; in \; M, j \; in \; N, i \; in \; Pred(j)$$

Si les tâches i et j sont exécutées dans la même unité de calcul, elles sont soit en série ou en parallèle

$$x_{ik} = 1 \gg x_{jk} \leq \delta_{ij} + \delta_{ji} + \varphi_{ijk} \quad \forall i \; in \; N, j \; in \; N, k \; in \; M, j \; not \; in \; P(i)$$

La tâche i est exécutée en priorité de j

$$\delta_{ij} + \delta_{ji} \leq 1 \quad \forall i \; in \; N, j \; in \; N, k \; in \; M, j \; not \; in \; P(i)$$

$$\delta_{ij} = 1 \gg s_j \geq s_i + t_{ik} \quad \forall i \; in \; N, j \; in \; N, k \; in \; M, j \; not \; in \; P(i)$$

Les tâches peuvent être exécutées en parallèle seulement dans le FPGA

$$x_{ik} + x_{jk} + 1 \geq 3 * \varphi_{ijk} \quad \forall i \; in \; N, j \; in \; N, k \; in \; M, j \; not \; in \; P(i)$$

$$\varphi_{ijk} = 0 \quad \forall i, j \; in \; F(k), k \; in \; M$$

Définition de Domaine

$$\delta_{ij}, x_{ik}, \varphi_{ijk} \; binary$$

4 Résultats

Nous avons effectué nos tests sur plusieurs architectures d'instances : une instance dans laquelle le FPGA n'exécute qu'une seule tâche à la fois (pas de parallélisme de tâche) d'une part, et d'autre part deux instances dans lesquelles le FPGA exécute un parallélisme de tâches de 2 et 3. Les figures 4, 5 et 6 donnent les résultats d'ordonnancement obtenus par notre modèle LP de résolution. Dans le cas d'une résolution sans parallélisme FPGA, notre modèle génère le meilleur résultat obtenu par le modèle précédent décrit dans [5], avec une valeur Cmax égale à 54.5. Pour un ordonnancement dans lequel on permet au FPGA un parallélisme de 2 tâches, on obtient un meilleur résultat avec une valeur Cmax égale à 52.

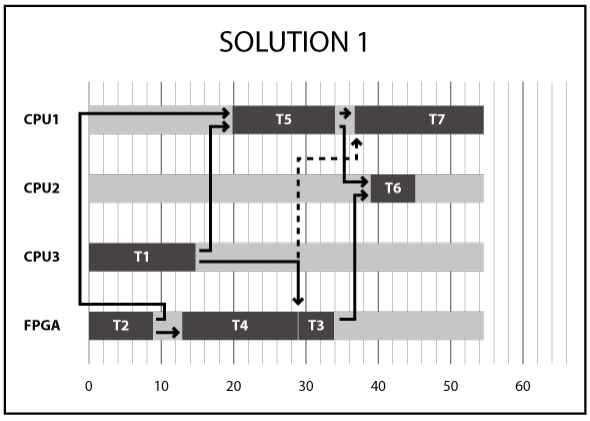

Fig 4 : Solution d'ordonnancement Parallélisme FPGA = 1.

Dans le cas, d'un parallélisme de 3, nous obtenons également une valeur Cmax égale à 52 qui correspond à la meilleure valeur pour tous les tests. En effet, avec une instance correspond à un parallélisme complet des tâches dans le FPGA (parallélisme de 7), nous obtenons la même valeur Cmax comme décrit dans la figure 7.

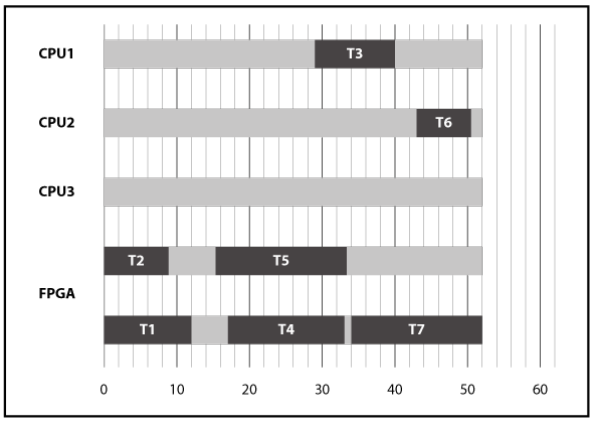

Fig 5 : Solution d'ordonnancement Parallélisme FPGA = 2.

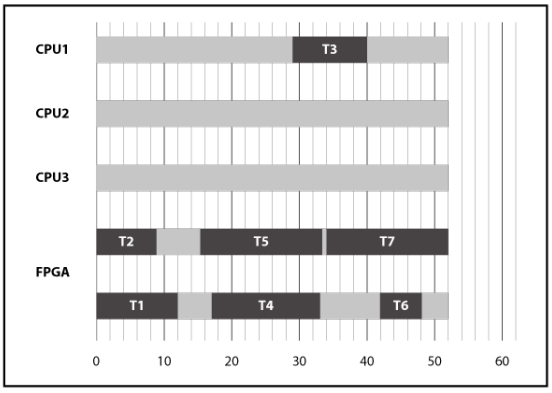

Fig 6 : Solution d'ordonnancement Parallélisme FPGA = 3.

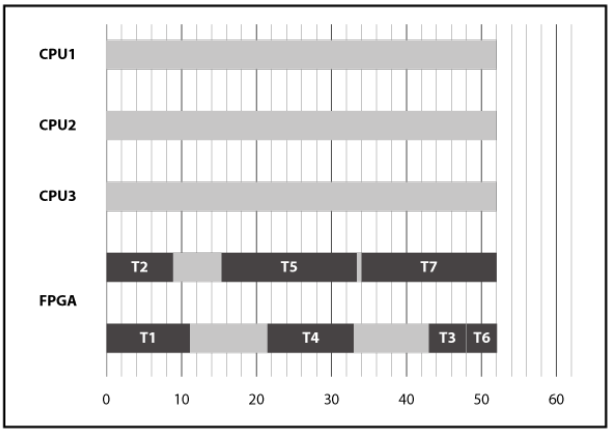

Fig 7 : Solution d'ordonnancement Parallélisme FPGA = 7.

5 Conclusion et perspectives

Dans cet article nous avons présenté une programmation linéaire améliorée pour la résolution de l'ordonnancement de tâches pour des architectures hétérogènes CPU/FPGA, considéré comme un problème NP-difficile. Notre méthode exacte permet de trouver une solution d'ordonnancement pouvant exécuter plusieurs tâches en parallèle dans l'unité FPGA en respectant la contrainte surface. Les résultats obtenus montrent l'efficacité de la méthode développée par rapport à des travaux similaires. La programmation LP proposée offre un très bon compromis entre la qualité des solutions obtenues (critère d'optimisation du makespan) et le temps de calcul nécessaire à leur obtention pour résoudre des problèmes à petite échelle. Dans le cas de recherche de solutions pour un nombre de tâches à grande échelle, des approches heuristiques seront abordées.

6 Références

[1] F. Abdallah, C. Tanougast, I. Kacem, C. Diou, D. Singer, "Genetic algorithms for scheduling in a CPU/FPGA architecture with heterogeneous communication delays," Computers & Industrial Engineering journal, Vol. 137, 2019, pp. 106006.

[2] Lee, W. C., Chuang, M. C. & Yeh, W. C.,"Uniform parallel-machine scheduling to minimize makespan with position-based learning curves", Computers and Industrial Engineering journal, Vol.63, 2012, pp813-818.

[3] F. Choobineh, E. Mohebbi, and H. Khoo. A multi-objective tabu search for a single-machine scheduling problem with sequence-dependent setup times. European Journal of Operational Research, Vol. 175, No. 1, pp. 318-337, 2006.

[4] F. Jina, S. Songa, and C. Wua. A simulated annealing algorithm for single machine scheduling problems with family setups. Computers & Operations Research, Vol. 36, No. 7, pp. 2133□2138, 2009.

[5] A. Ait El Cadi, O. Souissi, R. Ben Atitallah, N. Belanger, A. Artiba, "Mathematical programming models for scheduling in a CPU/FPGA architecture with heterogeneous communication delays", Journal of Intelligent Manufacturing, Vol. 29, 2018, pp. 629–640.

[6] ILOG AMPL CPLEX System version 9.0 User's Guide, September 2003.