Mapping multi Objectifs d'application intensive Sur architecture MPSOC

B.FARID, AH.BENYAMINA
Dept. Informatique Université d'Oran, Algérie
Dept. Informatique Université d'Oran, Algérie
farid.pgia@Gmail.com
benyanabou@yahoo.fr

***Résumé*— Ce travail nous permet de vulgariser une phase très importante du cycle de développement des systèmes embarqués : la conception software. Dans cette phase, le défi à relever est celui d'une application hiérarchique NOC hétérogène. Ceci Dans le but de satisfaire des objectifs contradictoires et de fournir des résultats dans des délais raisonnables. Nous utilisons pour cela les différentes approches exactes et heuristiques tout en adoptant un modèle permettant de représenter une structure à différents niveaux d'abstraction, dans le but d'être plus proche des problèmes d'association d'arbre hiérarchique : application et architecture.**

***Mots clés*— MPSoC, NOC, Dijkstra, Optimisation multi-objectifs, algorithme génétique, Branch and bound.**

I. INTRODUCTION

Un système embarqué [1] peut être défini comme un système électronique et informatique autonome ne possédant pas d'entrées/sorties standards comme un clavier ou un écran. On peut avoir un ou plusieurs systèmes embarqués sur puces (SOC, System On Chip), de nature hétérogènes et donc complexes. Ils sont souvent constitués de plusieurs processeurs de types différents (FPGA, DSP, GP par exemple), avec des fonctions dédiées ou reconfigurables [4].

Les applications embarquées qu'on a l'habitude de traiter sont généralement complexes [2] et parfois sont hiérarchiques telles les encodeurs MPEG, H263, H264…….etc. Ce type d'applications est caractérisé par des parties le constituant de types différents. L'un a trait au traitement irrégulier, l'autre a trait au traitement régulier [3]. Ce dernier représente le calcul intensif qu'on trouve en grande partie dans les applications temps réel embarquées [5]. A ces deux types d'applications correspondent deux types de traitements parallèles l'un est dit (irrégulier) correspond au parallélisme de taches et l'autre (régulier) corresponde au parallélisme de données.

C'est pour cette raison et afin d'avoir plus de performance qu'on ne traite pas ce type d'application dans sa globalité avec une seule stratégie. Pour faire le mapping de ce type d'application on dont réaliser le mapping des deux parties le constituant différemment. Ce qui nous amène à proposer une stratégie hybride sous le mapping hiérarchique constitué d'une part de stratégie de mapping pour le globale irrégulier et d'une stratégie pour le locale régulier [6]. D'où notre démarche proposant une solution globale hybride basée sur les algorithmes génétiques, Dijkstra multi objectif et la méthode Branch and bound.

II. DEFINFITION ET FORMULATION

Les communications entre les tâches de notre application et les composantes de notre architecture cible sont représentées par deux graphes orientés.

A. Définition 1

Le graphe d'application (TG : Task Graph) est un graphe orienté G(T,E) où chaque sommet $t_i \in T$ représente un module (tâche) de l'application et l'arc orienté (t_i, t_j) noté $e_{ij} \in E$ représente les communications entre les modules t_i et t_j. Le poids de l'arc e_{ij} noté par Q_{ij} représente le volume des communications de t_i à t_j (Fig1).

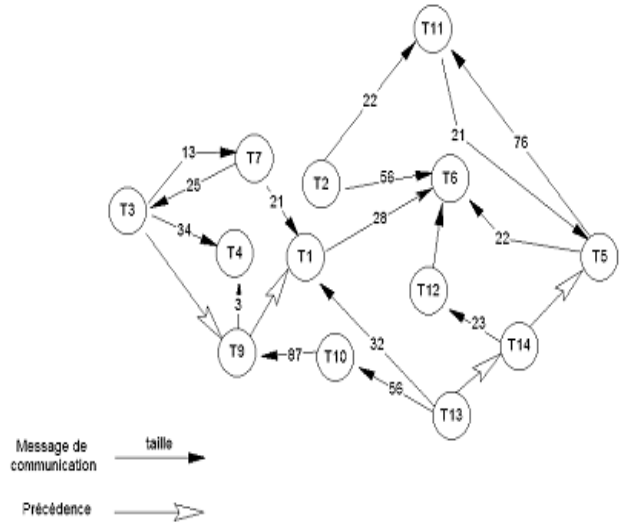

Fig. 1 Graphe d'application

B. Définition 2

le graphe d'architecture NT (NoC Topology graph) est un graphe orienté P(S,F) où chaque sommet $s_i \in S$ représente un nœud de la topologie et l'arc (s_i, s_j) noté par $f_i \in F$ représente un lien physique direct entre les éléments de s_i et s_j de l'architecture. Le poids de l'arc f_{ij} noté par bw_{ij} représente la bande passante,l'energié consomé et la latence disponible à travers le lien physique f_{ij} (Fig. 2).

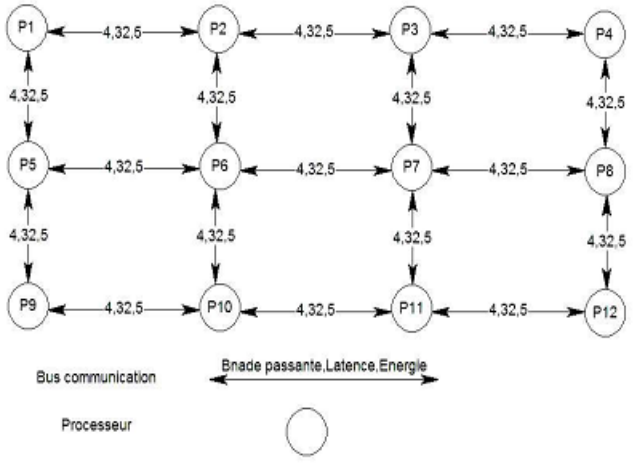

Fig. 2 Graphe d'architecture

Le mapping du graphe d'application G(T,E) sur le graphe d'architecture P(S,F) est définit par la fonction de mapping suivante :
map : T → S , s.t map (t_i)= s_j $\forall\, t_i \in T\ \exists\, s_j \in S$.
le mapping est définit si : $|T| \geq |S|$ (Fig3).

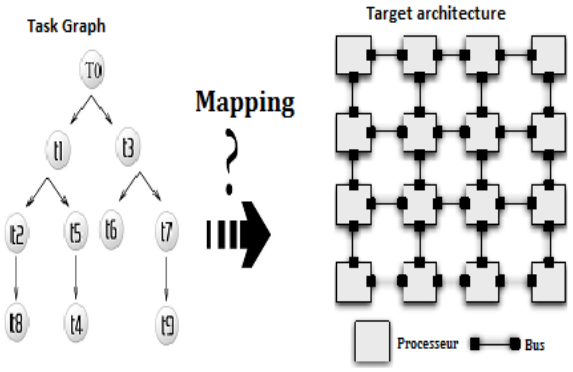

Fig. 3 Mapping de graphe de tache sur architecture NOC

C. Formulation mathématique

Notre application est définie comme un ensemble de tâches [7], T = {t1, t2, . . ., tn}, et l'architecture cible comme un ensemble de processeurs P = {P1,P2, . . .,Pn} .Comme hypothèse les processeurs Peuvent fonctionner en plusieurs modes m1, m2,m3. (Mettre les processeurs en plusieurs modes nous donne plus de diversité dans l'optimisation de placement)

- Le temps d'exécution (D):

Soit une tâche (T_i) place sur un processeur (P_j), calculer la communication $D_{com}(i)$ revient à chercher le plus court chemin entre (P_j) et tout autre processeur (P_L), là où les tâches successeurs sont placées.

$D^m_{Exec}(ij) = Taille_{(i)} / Freq^m_j$
Soit $|P_j, P_L|$: le plus court chemin entre P_j et P_L , alors :

$$D_{Com} = Max_k \left(\frac{Q_{ik}}{Min\,(Bd\,|P_j, P_L|)} \cdot \sum_{i \in |P_j, P_L|} Latence(P_j\ , P_{j+1}) \right)$$

Donc $D_i = D_{Exec(i)} + D_{Com\,(i)}$

Après l'ordonnancement:

$D = Max\,(D_i)$ /i=1……nombre de tâche

Tel que :
D: Le temps total d'exécution
$D^m_{Exec}(ij)$: La durée d'exécution d'une tâche i placée sur un processeur j s'exécutant au mode d'exécution m (sans prendre en compte les communications).
D_i : Le temps total d'exécution d'une tâche i en prenant en considération les communications.
$Taille_{(i)}$: Le nombre de cycles nécessaires d'une tache pour s'exécuter sur un processeur.
$Freq$: La fréquence d'horloge d'un processeur au mode m.
$D_{Com\,(i)}$: Durée de communication prise le transfert de données (résultat) de la tâche (i) ver ses successeurs.
Q_{ik} : Quantité de données échangées entre les tâches i et k
$Min\,(Bd\,|P_j, P_L|)$: La bande passante minimale dans le chemin de P_j à P_L .

- L'énergie consommée (E):

Energie d'exécution : $E^m_{Exec}(ij) = Taille_{(i)} * e^m_j$
Energie de communication : $E^i_{Com} = \sum Q_i * e(P_k, P_L)$
Energie totale : $E = \sum_{i=1}^{Nbre\ Tache} (E^i_{Exec} + E^i_{Com})$

Tel que :
$E^m_{Exec}(ij)$: L'énergie consommée par la tâche i affectée au processeur j au mode m
e_j : Energie d'exécution unitaire
E^i_{Com}: La consommation d'énergie due à la communication de la tâche i
E: La consommation d'énergie totale, en prenant en considération la consommation des processeurs et des liens.

III. METHODE DE RESOLUTION

Le problème à résoudre dans sa globalité est un problème d'assignation, d'affectation et d'ordonnancement «AAS». Ceci consiste principalement à assigner et ordonnancer les taches et les communications de l'application sur les ressources d'architecture cible de telle façon que les objectifs spécifiés soient atteints.

Dans notre approche il y'a des objectifs à respecter comme la minimisation de la consommation d'énergie, maximisation du temps de Performance. Ces objectifs sont très importants car le fonctionnement des systèmes embarqués mobiles dépendent de la durée de vie de la batterie [10] ce qui nécessite la réduction de la consommation d´énergie et maximiser de la vitesse d'exécution ce qui revient à dire minimiser le temps d'exécution. En réalité ces objectifs sont contradictoires car en réduisant l´énergie consommée on va augmenter le temps d'exécution des composants(CPU), fonctionnant en mode économie. Pour cela, on est obligé de faire du multi-objectifs afin de trouver un compromis entre les différents objectifs contradictoires.

Donc nous proposons une approche multi-objectifs basée sur la technique d'optimisation par les algorithmes génétiques et l'algorithme de B&B pour résoudre notre problème AAS.

1) Algorithme1: Algorithme génétique(AG): appartient à la famille des algorithmes évolutionnistes. Leur but est d'obtenir une solution approchée à un problème d'optimisation, lorsqu'il n'existe pas de méthode exacte (ou que la solution est inconnue) pour le résoudre en un temps raisonnable. Les algorithmes génétiques utilisent la notion de sélection naturelle et l'appliquent à une population de solutions potentielles au problème donné [8].

L'algorithme génétique se résume par :

Algorithme 1 Algorithme Génétique

- Initialisation de la population.
- Evaluation des fonctions objectives.
- pour i = 0 à MaxIter faire
- Sélection
- Croisement
- Mutation
- Evaluation des fonctions objectives
- fin pour

2) Algorithme2: Algorithme de branch and bound (B&B) :

Un algorithme par séparation et évaluation, également appelé selon le terme anglo-saxon *branch and bound*, est une méthode générique de résolution de problèmes d'optimisation, et plus particulièrement d'optimisation combinatoire ou discrète. C'est une méthode d'énumération implicite : toutes les solutions possibles du problème peuvent être énumérées mais, l'analyse des propriétés du problème permet d'éviter l'énumération de larges classes de mauvaises solutions. Dans un bon algorithme par séparation et évaluation, seules les solutions potentiellement bonnes sont donc énumérées [9].

L'algorithme de branch and bound se résume par :

Algorithm 2 Algorithme Branch and baund

Initialisation : Best_Sol =∞; $B_i(p_0) := f(p_0)$;
$p = \{(p_0, B_i(p_0))\}$
Tant que $p \neq \phi$ faire
  Sélection: sélectionner un nœud p de P :
  $p := p / \{ p \}$;
  Pour i=1 …k faire
    Evaluation $B_i(p_i) := f(p_i)$
    Si ($B_i(p_i) == f(x)$), x réalisable et ($f(x) <$ Best_Sol) alors
      Best_Sol:= $f(x)$
      Solution := x;
    Sinon
      Si $B_i(p_i) >$ Best_Sol alors
        Elaguer p_i;
      Sinon
        Séparation: Décomposer p en $p_1, p_2, ..., p_k$;
        $p \cup \{(p_i, B_i(p_i))\}$
      Fin Si
    Fin Si
  Fin Pour

3) Description de notre approche :

Dans le flot de conception, l'étape de placement et ordonnancement est directement liée à l'implémentation de l'application sur une architecture spécialisée. Les entrées de cette phase sont :

- Un modèle d'application
- Un modèle d'architecture cible
- Des contraintes de performance et d'énergie,
- Des fonctions objectives à optimiser.

La sortie de cette phase est une affectation des différentes tâches et communications aux ressources physiques, selon un ordonnancement d'exécution des différentes tâches sur ces ressources.

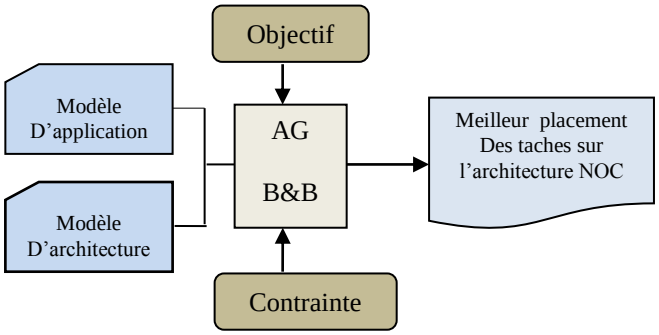

Fig. 4 Problématique de placement et d'ordonnancement

IV. Experimentation et Analyse des resultat

Pour le développement de notre approche, on a opté pour le langage de programmation JAVA, et toutes les expériences ont été réalisées sur un Pentium double core 2.04 GHZ sous Windows 7.

Après l'exécution de notre approche hybridée (GA et B & B) en utilisant les propriétés et les paramètres de base indiqués dans le tableau 1, les résultats obtenus sont comme suit:

Algorithme	Paramètre de base	
Algorithme génétique	-Nombre de tache -Nombre de processeur -Architecture -Latence -Taille de la Population -Nombre de Génération -Probabilité de croisement -Probabilité de Mutation	21 9 Topologie en étoile 1 unité 80 40 80% 1 %
Algorithme branch&bound	-Nombre de tache -Nombre de processeurs -Architecture -Latence	290 9 Mesh 2D (3*3) 1 unité

Tableau1 : représentation des paramètres de base

A. *Le modèle hybride proposé :*

Le modèle hybride proposé est illustré par la figure 5 :

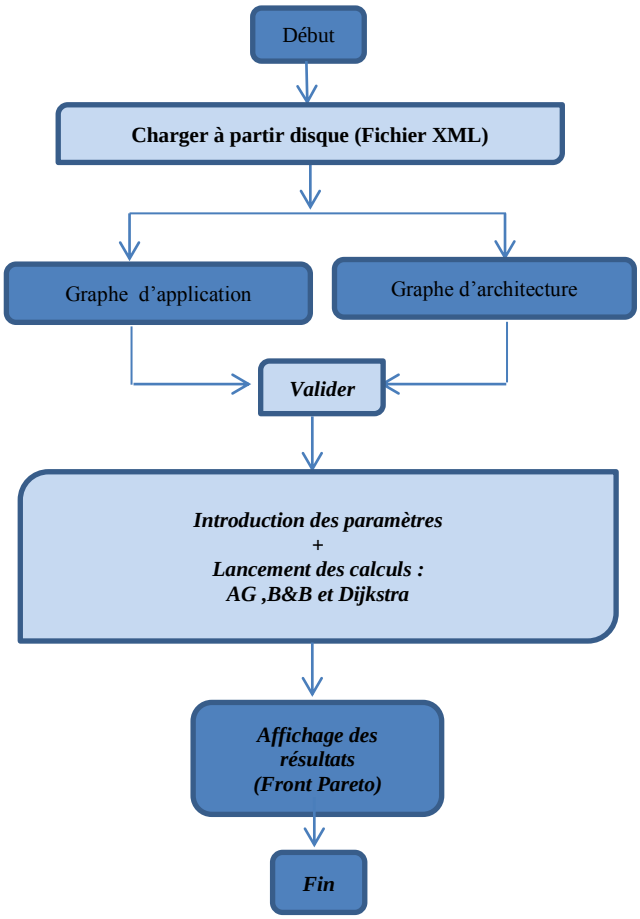

Fig. 5 Le modèle hybride proposé

B. *Affichage des résultats :*

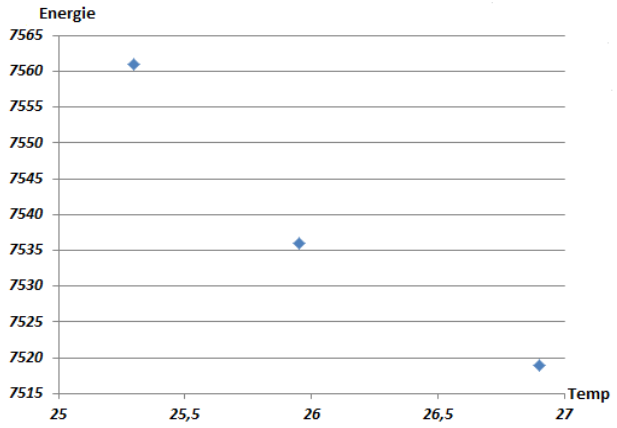

Fig. 6 Front Pareto pour B&B

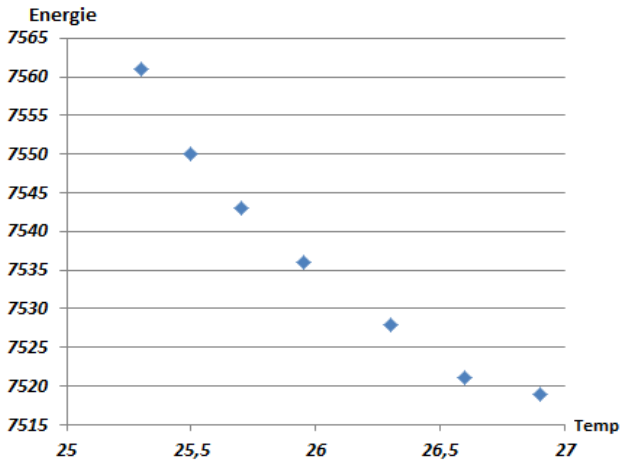

Fig. 7 Front Pareto pour AG

V. CONCLUSIONS

Dans cet article nous avons présenté notre approche, basée sur l'algorithme Génétique et algorithme de Branch and Bound. Ces derniers sont hybridés avec l'algorithme de plus court chemin dijkstra multi objectifs afin de résoudre le problème de placement et d'ordonnancement d'une application hiérarchique sur une architecture MPSoC.

Nous considérons pour le moment que notre solution est très efficace parce qu'on a utilisé une meilleur méta heuristique et exact pour traiter les deux types de taches régulières et irrégulières et assurer le parallélisme de tache et de donnés (GILR : Globally Irregular Locally Regular).

En attendant des expérimentations et simulations plus poussées, nous considérons que notre solution est très efficace parce qu'on a utilisé une meilleur méta heuristique et un algorithme exact pour traiter les deux types de tâches.

REFERENCES

[1] Y. A. Laribi, " Environnement de mapping pour la Conception des réseaux sur puce(NoCs)," pour obtenir le titre de Magistère. L'école nationale d'informatique ESI, Algérie, Juillet 2010.

[2] R.A. Mohamed, " Optimisation multicritère pour le placement d'applications intensives sur système sur puce (SoC)," pour obtenir le titre de Master. l'Université de Lille Mention : Informatique. Juin 2010.

[3] B. Abou-El Hassen, " Ordonnancement Hiérarchique Multi - Objectif D'Application Embarquées Intensives," thèse pour l'obtention du Diplôme De Doctorat D'Etat, université d'Oran , Algérie, décembre 2008.

[4] B. Henri et H. Matthew , " Approaches for integrating task and data parallelism," IEEE Concurrency, 6(3):74 –84, 1998.

[5] A..Benyamina, B.Beldjilali et S.Eltar, " Time Applications on NoC Architecture with Hybrid Multi-objective PSO Algorithm," COSI'2010 . Ouaragla, Algerie, Avril, 2010.

[6] A. Liefooghe, B. Matthieu, L. Jourdan, and T. El-Ghazali, " Paradiseo-moeo : A framework for evolutionary multi-objective optimization," In Evolutionary Multi-criterion Optimization (EMO 2007), vol 4403 of Lecture Notes in Computer Science, pp. 386–400, Matsushima, Japan: Springer-Verlag, 2007.

[7] A. Abdelkader, " Ordonnancement multi objectif d'application intensif sur une architecture régulières," pour obtenir le diplôme de magistère, université d'Oran, Algérie, février 2012.

[8] T. El-Ghazali, " Metaheuristics : From Design to Implementation," Wiley-Blackwell (an imprint of John Wiley sons Ltd), juillet 2009.

[9] M. ashish , " Allocation, Assignation, et Ordonnancement pour les systèmes sur puce multiprocesseurs," thèse pour l'obtention du Diplôme De Doctorat D'Etat, université de Lille , France, décembre 2006.

[10] A. Mehran,S. Saeidi,A. khademzadeh, and A. Afzali-khusha , "Spiral : A heuristic mapping algorithm for network on chip," IEICE Electronics Express, 4(15) :478–484, 2007.