République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Saad Dahlab, Blida

Faculté des Sciences

Département d'informatique

Mémoire présenter par :

- ***FERROUKHI** Nassiba*
- ***SNOUSSI** Nassima*

En vue de l'obtention du diplôme de Master

Domaine : Mathématique et Informatique

Filière : Informatique

Spécialité : GL

Thème :

Optimisation du temps dans la conception des réseaux sur puce (NoC)

Encadré par-devant :

- ***Mr.Bougherara***
- ***M^{elle}.Zahra***

le jury composé de :

- ***M.Boumehdi***
- ***M.Toubaline***
- **M.Arkam**

Année Universitaire

2012/2013

*On remercie tout d'abord, notre vénéré **Allah**, le tout puissant a qui nous devons le tout.*

*On tient à exprimer notre gratitude à nos promoteurs **Mr.Bougherara** et **Melle.Zahra** pour avoir dirigé les travaux de recherche, on les remercie également pour leurs précieux conseils et particulièrement pour leur disponibilité.*

On remercie aussi les enseignants de notre section de nous avoir conseillé pour le bon déroulement de cette soutenance, et pour leurs encouragements.

Comme on tient à remercier aussi les administrateurs de notre département pour avoir mis à notre disposition les ressources matérielles nécessaire pour la réalisation de ce travail.

On remercie enfin toute personne qui a participer de prêt ou de loin pour nous aider dans ce mémoire.

Merci a tous.

✓ *Snoussi Nassima :*

« Souhaitant que le fruit de nos efforts fournis Jour et nuit, nous mène vers le Bonheur fleuri. »

**Tout d'abord je dédie ce modeste travail à mes très chers parents, qui ont toujours été là pour m'apporter de l'aide, du soutien, des conseils et beaucoup de tendresse, en reconnaissance de tous les sacrifices consentis par tous, grâce a vous j'ai pu atteindre cette étape de ma vie Merci.*

**A mon frère Nadji, ma moitié, mon bonheur et ma joie. Dieu merci de me l'avoir donné, Je te souhaite un avenir plein de réussite et de sérénité.*

**A mes grands-parents, mes tantes, mes oncles, mes cousins pour l'amour qui m'ont apporté durant toute ma vie.*

**A ma très chère tante Karima et sa petit famille, merci d'être là dans les moments difficile de ma vie.je vous aimes*

**A ma binôme et sœur Nassiba, avec qui j'ai eu un grand plaisir de partager ce projet, c'était une source d'émotion agréable et des moments inoubliables.*

**A mon oncle, Monsieur le recteur Snoussi Sid ahmed, merci pour l'aide précieuse et les conseils fournis durant toute ma scolarité.*

A mes cousines : Lynda, Rym, Nesrine, inaam, nayla.

**A mes très chères amies : Amina, Nihad, Amira, Nawal, je vous aime toute tant.*

**A mes amis et frères : Kader, Redha, Amine, Boualem, Zaraki, Mohamed. Merci d'être la, de faire partie de ma vie.*

**A ma petite sœur Aziza, un rayonnement qui éclaire ma vie, je t'aime si fort.*

**Je n'oublierai jamais la gentillesse et la bonne humeur de tante Fouzia, mon oncle Youcef et le petit Adel.*

**Au gens du Sud de Timimoune et Tamanrasset, pour leur hospitalité, leur générosité et leur amour.*

Mes remerciements s'étendent également à toutes les personnes qui ont contribué à ma thèse de prés ou de loin.

Le réseau sur puce est un nouveau concept d'interconnexions dans les systèmes mono puce. Cette architecture facilite l'intégration de composants complexes et semble s'adapter à l'évolution des applications. Cependant, comme toute nouvelle technologie, elle requiert des efforts en recherche, en particulier pour l'accélération et la simplification des phases de conception.

Les deux phases de placement (mapping) et d'ordonnancement représentent des phases centrales lors de la mise en œuvre d'un réseau sur puce, la phase de placement permet de placer les éléments d'une application sur l'architecture, et la phase d'ordonnancement permet d'ordonnancer l'exécution de ses éléments, tout cela en optimisant certains objectifs. Donc il est plus rentable de mettre en place des outils et méthodes qui permettent d'automatiser ces phases.

C'est alors dans ce cadre que notre travaille s'élabore, en utilisant un algorithme évolutionnaire « DE » pour le placement des IP d'une application sur l'architecture d'un réseau sur puce, tout en minimisant le cout de communication. Et deux algorithmes d'ordonnancement «ASAP» et «ALAP» qui se basent sur les résultats du placement pour faire l'ordonnancement des IP en calculant le temps d'exécution.

Afin d'évaluer les performances des algorithmes utilisés, une application qui permet d'effectuer le mapping et l'ordonnancement est implémentée.

Mots clés :

Réseaux sur puce, système sur puce, mapping, ordonnancement, communication, l'algorithme DE « differentiel evolution », l'algorithme ASAP, l'algorithme ALAP.

The network on chip is a new concept of interconnections in single-chip systems. This architecture facilitates the integration of complex components and seems to adapt to change applications. However, like any new technology, it requires research efforts, especially for accelerating and simplifying the design phases.

The two phases of Mapping and scheduling phases are central in the implementation of a network on chip, the mapping phase will place the elements of an application architecture, and phase scheduling allows to schedule the execution of its elements, all while optimizing certain objectives. So it is more profitable to develop methods and tools that automate these steps.

It is then in this context that our work is developed, using an evolutionary algorithm "DE" for the mapping of an IP application on the architecture of a network on chip, while minimizing the cost of communication. And two scheduling algorithms "ASAP" and "ALAP", which based on the results of mapping to scheduling IP and calculating the execution time.

To evaluate the performance of the algorithms used, an application that allows the mapping and scheduling is implemented.

Keywords:

Networks-on-Chip, SoC, mapping, scheduling, communication, algorithm DE « differential evolution », the ASAP algorithm, ALAP algorithm.

Table des matières :

Liste des figures et tableaux :

Introduction générale

L'évolution rapide de la technologie informatique représente l'un des plus importants phénomènes techniques depuis plusieurs décennies d'où la miniaturisation de toute appareil électronique (téléphone mobile, GPS...)

Actuellement, le monde, avec la révolution numérique, est transformé par la création des téléphones mobiles numériques plus petits, plus légers et plus performants.

Ce rythme d'évolution effréné provient des avances technologiques des circuits intègres et de leurs architectures qui ont permis de développer des systèmes informatiques plus petits, plus compacts et plus rapides, grâce en grande partie a la capacité d'intégration de plus en plus élevée. Un type de ces systèmes qui a profité largement de cette évolution est le système embarqué.

Les techniques de conception de ces systèmes permettent maintenant de regrouper des systèmes hétérogènes sur la même puce électronique, donnant ainsi naissance à un nouveau paradigme dans les systèmes embarques : les systèmes sur puce (SoC : System on Chip).

Ces SoC permettent d'intégrer différents éléments comme des processeurs, des mémoires, des blocs d'entrée/sortie ou des médias de communications.

L'augmentation des performances et la miniaturisation des circuits ont amené les concepteurs à la réalisation de systèmes électroniques de plus en plus flexibles et complexes, intégrant plusieurs processeurs qui sont les systèmes sur puce multiprocesseurs (MPSoC : MultiProcessor System on Chip), ce pendant, les liens de communication n'évoluent pas à la même vitesse et devient un goulot d'étranglement.

Les solutions de communications actuelles telles que le bus partagé trouvent leurs limites en termes de bande passante et d'extension à mesure que le nombre d'éléments communicants augmente. De nos jours cette structure est la plus utilisée, mais elle ne semble pas s'adapter aux applications futures, c'est dans ce contexte que les réseaux sur puce (NoC : Networks On Chip) sont apparus.

Ce paradigme d'interconnexion, inspiré des réseaux informatiques classiques, offre une structure de communication évolutive, flexible et propose des solutions efficaces aux problèmes d'intégrations complexes des systèmes sur puce. Cependant, il n'existe pas d'outils d'automatisation de l'ensemble des phases de conception. C'est pourquoi ils constituent l'un des axes les plus actifs de la recherche.

Les deux phases de placement et d'ordonnancement sont des phases essentielles dans la conception des réseaux sur puces. La phase d'ordonnancement consiste à ordonnancer les tâches selon la plate forme d'exécution (homogène, hétérogène) et l'indice de performance qu'on veut optimiser (le temps total d'exécution, le temps moyen que passe une tâche dans le système, énergie consommé,. . .).

Tandis que, la phase placement consiste à placer chaque élément de l'application sur l'architecture du réseau sur puce de sorte à minimiser le cout de communication et respecter des contraintes telle que la bande passante ou le temps réel.

L'objectif de notre travail consiste à étudier certaines techniques et algorithme de placement et d'ordonnancement existant, et choisir des algorithmes ou techniques qu'on va implémenter qui permettent d'ordonnancer et de placer une application sur une architecture de NoC tout en optimisant une fonction qui est définie par la somme entre le cout de communication et le temps d'exécution.

Le mémoire présenter est structuré en six chapitres qui permettent un cadrage progressif du sujet.

Le premier chapitre parle sur les concepts de base des systèmes multiprocesseur, les différentes structures d'interconnexion ainsi que les limites de chacune.

Le second chapitre introduit les notions liées aux réseaux sur puce. Les composants de cette structure, ses caractéristiques ainsi que quelques exemples d'architectures, un tour d'horizons des problèmes liés à cette solution sont présentés, on va voir aussi les phases de conception de ces réseaux sur puce ainsi que des outils de mise en œuvre.

Dans le troisième chapitre, nous présentons les problèmes de placement et ceux de l'ordonnancement lié au réseau sur puce, quelques techniques pour la résolution de ces problèmes, et on va finir par présenter quelques approches de la littérature qui permettent de résoudre ces deux problèmes simultanément.

Le quatrième chapitre, va permettre de décrie les notions d'optimisation et quelques une de ses méthodes, dans un second lieu nous définissions quelques métas heuristiques existants.

Dans le cinquième et l'avant dernier chapitre, nous allons présenter et en détaille les algorithmes que nous avons choisit pour résoudre notre problèmes (DE, ASAP et ALAP).

Le dernier chapitre, vas contenir l'implémentation des algorithmes utilisés pour la réalisation de notre solution et les différents tests effectués ainsi qu'une comparaison entre les résultats obtenus et des résultats de méthode déjà existante pour mettre en évidence l'efficacité de la solution.

Ce mémoire s'achève par une conclusion générale et quelques perspectives de notre travail.

Chapitre I : système multiprocesseur

1. Introduction :

Les systèmes basés sur les puces électronique font de plus en plus partie de notre vie quotidienne sans pour autant s'en rendre compte, on peut les trouvé dans nos téléphones portables, les ordinateurs et aussi les différentes consoles de jeux qui existes.

Pour la mise en œuvre de ces architectures les concepteurs se basent sur les communications au sein de la puce car c'est ici que se trouve le cœur du système.

Dans ce chapitre on va définir ces systèmes et voir les moyens d'interconnexion dans ces structures.

2. Définition du système sur puce (SoC) :

Un système sur puce dont l'abréviation est (SoC du nom anglais System On Chip) est tout un système en entier embarqué sur une puce. Ce système doit comprendre de la mémoire, un ou plusieurs microprocesseurs, des périphériques d'interface, ou tout autre périphérique nécessaire à la réalisation de ses fonctions [1]. Ce système ne dois dépendre d'aucune (ou de très peu) de composants externe pour exécuter ses tâches.

3. Architecture monoprocesseur de base :

Un système monoprocesseur de base se compose de trois principaux composants qui sont [2] :

- La mémoire principale
- L'unité centrale de traitement (CPU)
- Le sous système entrée/sortie (E/S)

Ces composants son tout relié par un même bus comme le montre la figure suivante.

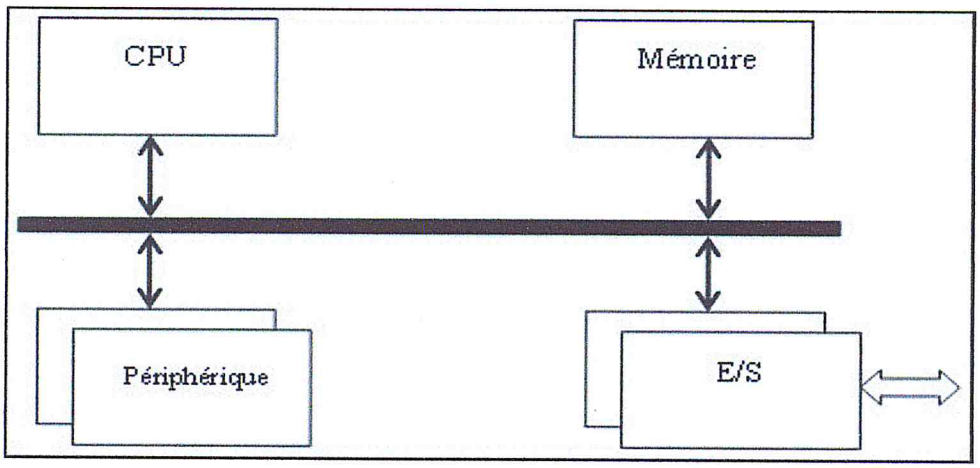

Figure1 : Architecture monoprocesseur de base

4. Système multiprocesseur mono puce (MP SoC) :

Un système multiprocesseur mono puce (MP SoC du nom anglais Multi-Processor System on Chip) [3] est un SoC qui intègre plusieurs composants logiciels complexes et hétérogènes, comme des éléments de calcules (CPU « central processing unit », DSP « Degital Signal Processor », FPGA « Fieled-Programmable Gate Array »), des réseaux de communication, des mémoires et des périphériques d'E/S, et tous cela sur une même puce [4].

5. Les architectures multiprocesseurs:

Dans les multiprocesseurs il existe deux générations d'architectures qui sont :

a. Multiprocesseurs 1ère génération :

Cette première génération représente une amélioration des architectures monoprocesseurs. Où on a amélioré les performances du CPU, et confié des traitements qui au paravent on été pris en charge par des accélérateurs matériels dédiés, a un ou plusieurs processeurs spécialisé comme le DSP (Digital Signal Processor).

Cette solution permet d'exploité à la fois la puissance et la flexibilité des processeurs spécialisés. [4]

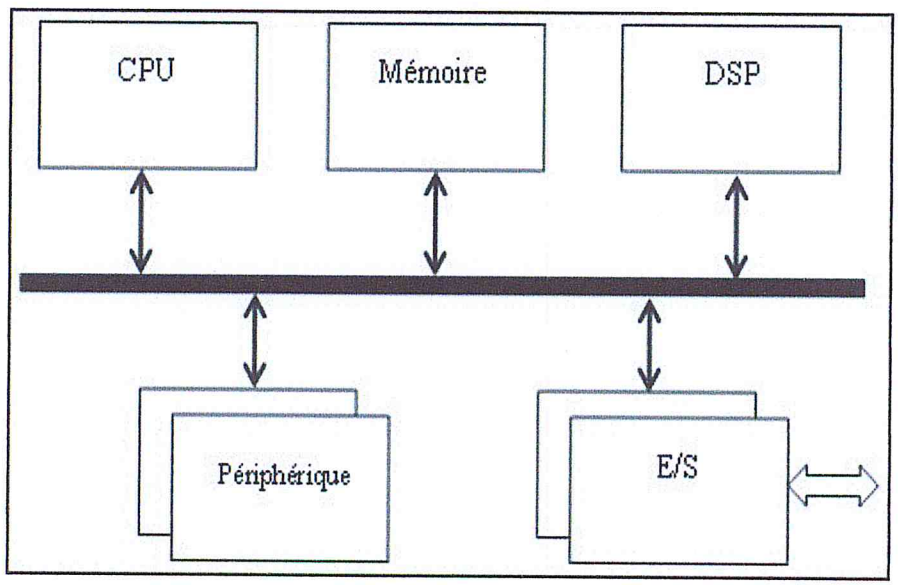

Figure2 : Architecture multiprocesseur de 1ère génération

b. Multiprocesseurs de 2ème génération :

Ces 2èmes générations d'architectures sont des architectures massivement parallèles, qui intègrent un grand nombre de processeurs et composants matériels sur une même puce. Et

cela est du a l'évolution du besoin en terme de puissance est de traitement d'un coté, et la percée importante de la capacité d'intégration d'un autres. [5]

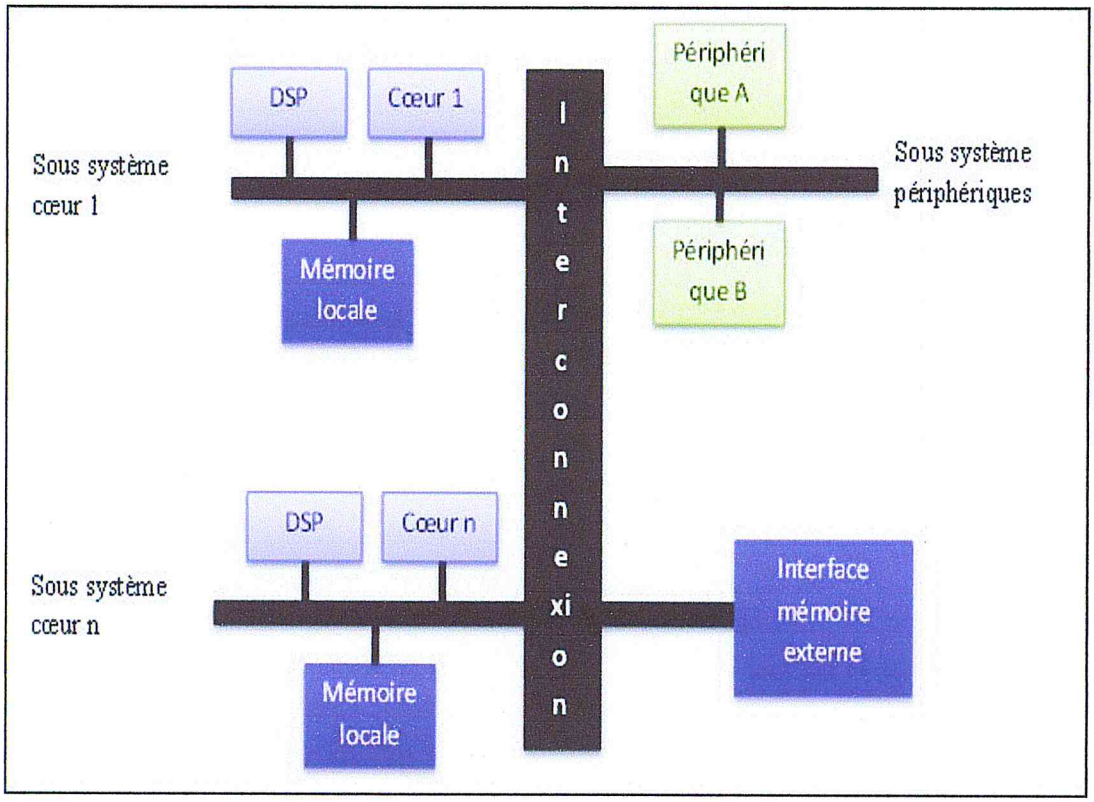

Figure3 : Architecture multiprocesseur de 2ère génération [6]

6. Exigence des futures structures d'interconnexions :

A fin que les architectures d'interconnexions s'adaptent à l'évolution des systèmes qui l'intègrent, elles doivent répondre aux exigences suivantes :

a. Flexibilité et extensibilité :

L'architecture d'interconnexion doit nous permettre d'interconnecté de nombreux bloc d'IP de nature hétérogène, qui peuvent être développés par différentes équipes, avec différents outils sous différents contexte. C'est pour cela que les structures d'interconnexion doivent êtres suffisamment flexible pour s'adapter à ces modules hétérogènes destiné à être intégrer dans le même circuit.

D'un autre coté, l'extensibilité qui désigne le fait de pouvoir augmenté le nombre de blocs interconnecté à tout moment, tout en conservent le même niveau de performance dans les communications. [7]

b. Simplicité :

Désigne la simplicité de mise en œuvre de cette structure, en minimisant le nombre de ressources nécessaires à sa réalisation.

7. Les types d'interconnexion actuelle dans les systèmes mono puce :

a. La connexion point à point :

C'est la connexion la plus simple pour connecter deux IP (Intellectual Property). Les blocs fonctionnels sont donc reliés entre eux directement sans avoir à définir un protocole de gestion de communication [8].

Chaque connexion est dédiée et spécifique ce qui permet d'avoir des connexions à haut débit et un parallélisme contrôlé [9].

Mais malheureusement ce type d'interconnexion est limité, car certaines IP peuvent avoir besoin de communiquer avec d'autre IP en fonction des besoins de l'application, et la chose la plus importante est que l'ensemble du SoC peut devenir inutilisable si un bloc fonctionnel de la chaine est défectueux, de plus elle n'est pas flexible donc difficilement réutilisable. C'est à cause de ces points négatifs de cette connexion que la topologie en bus a été mise en place.

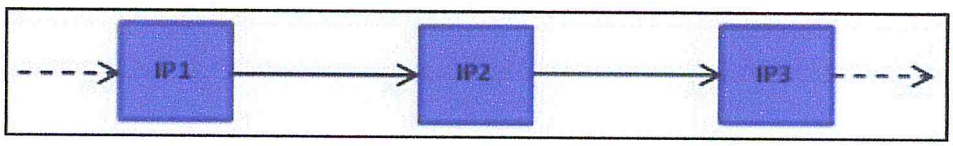

Figure4 : La connexion point à point

b. La connexion par bus :

Contrairement à la connexion point à point, la connexion par bus permet de connecter tous les blocs fonctionnels à un seul média de communication qui est le bus [8].

Ce type de connexion est plus flexible par rapport au type précédent et mieux réutilisable.

Par contre l'inconvénient majeur de ce type d'interconnexion est de ne pouvoir réaliser qu'une seule communication à la fois (les tâches peuvent se dérouler en parallèles mais pas les communications), comme toutes ces communications sont gérées toutes par le biais du bus alors on crée un goulot d'étranglement lorsque le nombre de communication augmente, mais aussi lorsque les contrainte des bandes passantes de plusieurs communications devient trop grandes.

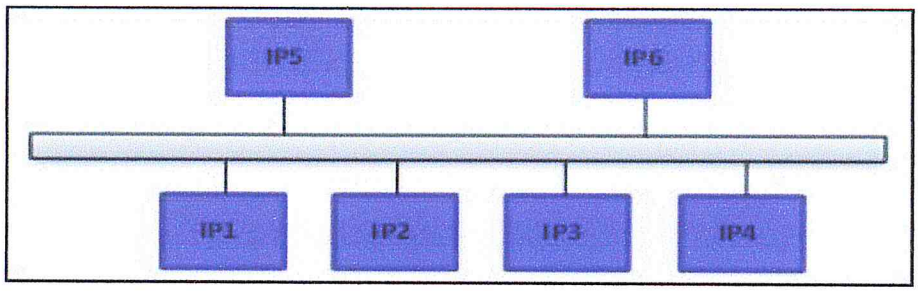

Figure5 : La connexion par bus

c. La connexion par bus hiérarchique :

Pour remédier aux problèmes rencontrés dans la structure d'interconnexion par bus on a proposé une nouvelle structure d'interconnexion qui est la structure de bus hiérarchique, elle consiste à connecter plusieurs segments de bus deux a deux par l'intermédiaire d'un pont (Bridge). Le point fort de cette structure est que les IP sont réparties dans plusieurs bus différents et non connecter à un seul bus comme dans la structure précédente, ce qui permet d'équilibrer la charge dans les bus et diminuer sa longueur ce qui augmente la performance du bus d'un côté, mais aussi d'un autre côté de rendre les communication qui ne se trouve pas dans les même lignes parallèles [8].

Mais comme les structures précédentes cette structure trouve des lacunes au niveau des ponts, c'est ici que se crée un goulot d'étranglement lorsque le nombre d'éléments communicants augmente.

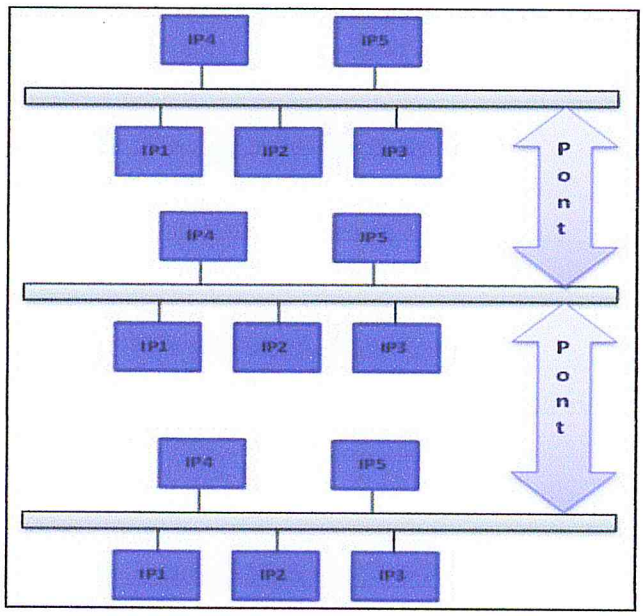

Figure6 : La connexion par bus hiérarchique

d. Le réseau sur puce :

Un réseau sur puce (NoC du nom anglais Network On Chip) est composé d'un ensemble de commutateurs (routeurs) reliés entre eux par des liens.

La tâche essentielle du réseau sur puce est d'échanger des informations d'un point vers un autre, tout en offrant une meilleure performance que le bus, au niveau de la bande passante. Et en plus cette architecture est flexible et facilement extensible, mais cette solution présente un cout de conception important.

Les travaux sur ces structures restent du domaine de la recherche académique. Il n'existe pas de solution industrielle car les approches ont été jugées manquantes de maturité [8].

8. Comparaison entre les types d'interconnexions :

Dans ce chapitre on a cité plusieurs types d'interconnexion dans les systèmes sur puce avec leurs avantages et leurs limites qui on menait aux solutions actuelles, notamment aux réseaux sur puces.

Le tableau ci-dessous représente une comparaison entres ces différentes structures d'interconnexions [9].

Connexion	parallélisme	consommation	scalabilité	réutilisation
Point à point	++	+	--	--
Bus partagé	--	--	--	++
Bus hiérarchique	+	-	-	++
NoC	++	++	++	++

++ : Très bon

+ : Bon

-- : Très mauvais

- : Mauvais

Tableau 1 : comparaison entre les différentes structures d'interconnexions dans les SoC.

9. Conclusion :

Les types de structure d'interconnexion vue dans ce chapitre et qui sont basé fondamentalement sur les bus partagés sont les plus utilisés de nos jours, mais elles sont limitées en terme de bande passante et d'extension à cause de l'augmentation du nombre d'éléments communicants, et c'est pour cela que ce type de structure d'interconnexion ne semble pas s'adapté aux applications futures. Et c'est dans ce contexte que les réseaux sur puce (NoC) ont été introduits, dans le but de résoudre ces problèmes.

Dans le chapitre suivant cette nouvelle structure d'interconnexion est détaillée.

Chapitre II : Les réseaux sur puce

1. Introduction

De nos jours un même système doit fournir un grand nombre de fonctionnalités, et pour un tel résultat un nombre énorme de blocs fonctionnels doivent communiquer entre eux.

Et pour ce fait les types d'interconnexion qu'on a vu dans le chapitre précédent deviennent limité en terme de bande passante, d'extensibilité et de parallélisme des tâches ce qu'ils ne pourront pas s'adapté aux besoins des systèmes qui augmentent de jour en jour, et c'est dans ce contexte que le concept des réseaux sur puce (Noc) a vu le jour, et est censé apporté des solutions aux problèmes des structures d'interconnexions actuelles et s'emble s'adapté aux systèmes futurs.

Dans ce chapitre on va définir les notions reliées à cette structure de communication.

2. Définition d'un réseau sur puce:

Un réseau sur puce (NoC) est une technique de conception du système de communication entre les cœurs sur les System on Chip (SoC). Le NoC applique les théories et méthodes de réseau aux communications à l'intérieur d'une puce [10]. Cela étant dit, la tâche essentiel d'un réseau sur puce (NoC) est d'échanger des informations d'un point vers un autre tout en améliorant les performances par rapport aux interconnexions de bus et cela au niveau de la bande passante et de la scalabilité.

3. Composants d'un Réseau sur Puce (NoC)

Un réseau sur puce est composé de routeurs, d'adaptateurs réseau (interface réseau), d'IPs (Intellectual Property), et de liens comme le montre la figure suivante :

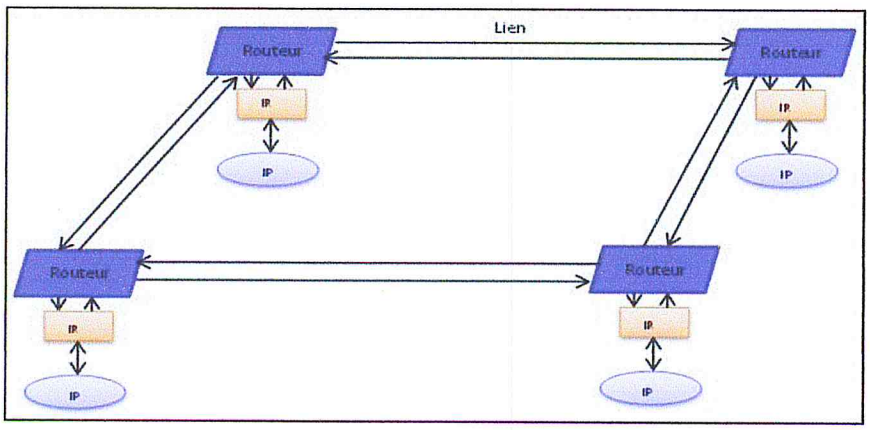

Figure7 : Composants d'un NoC [11]

a. Les adaptateurs réseau (IR) :

Ils réalisent l'interface entre le protocole du NoC et celui des blocs IPs, leurs rôle est de séparer le traitement (qui s'effectue au niveau des IPs) des communications (acheminés par le réseau).

b. Les nœuds de routages :

Leurs rôle principale est d'acheminer les données d'une source à une destination.

Un routeur est constitué de :

- ✓ Filles d'attente pour stocker les paquets qui circulent dans le réseau
- ✓ Des ports d'entrée et de sortie
- ✓ Une unité de commutation (switch) pour acheminer les paquets
- ✓ Une unité de contrôle pour calculer le chemin et allouer les ressources nécessaires

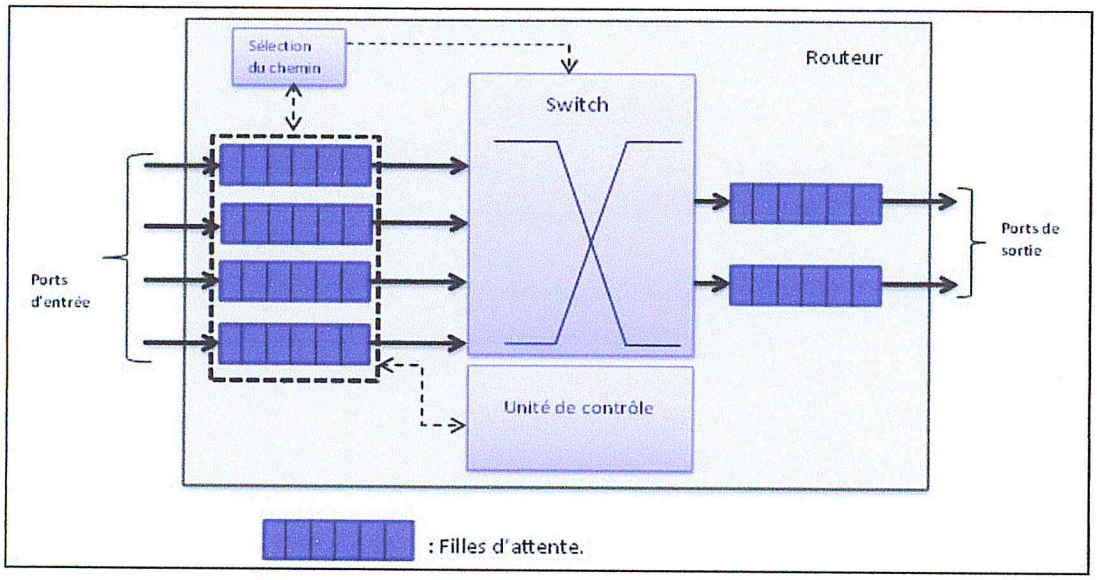

Figure8 : Structure d'un routeur sur puce [12]

c. Les liens :

Ils relient les routeurs entre eux ou les routeurs avec les interfaces réseau. Ce sont eux qui offrent la bande passante pour la communication entre la source et la destination. Un lien peut posséder plusieurs canaux virtuels et peut être mono ou bi directionnel.

4. Caractéristiques des réseaux sur puce :

Un réseau sur puce (NoC) est caractérisé par : la topologie, le mode de commutation, et le contrôle de flux.

✓ Topologies :

La topologie d'un réseau spécifie l'architecture physique du réseau, et définie la disposition de ces éléments. Une topologie peut aussi se caractérisé par sa dimension et sa forme.

Dans la littérature on peut trouver plusieurs topologies de réseaux sur puce, mais les plus utilisés sont :

a. La topologie 2D maillée :

Cette topologie est la plus souvent utilisé et cela reviens a sa facilité de mise en œuvre [8]. Le principe de cette topologie est que tout nœud et connecté directement a ses voisins et cela permet une certain fiabilité, car en cas de panne dans un lien ou un nœud d'autres chemins seront possible.

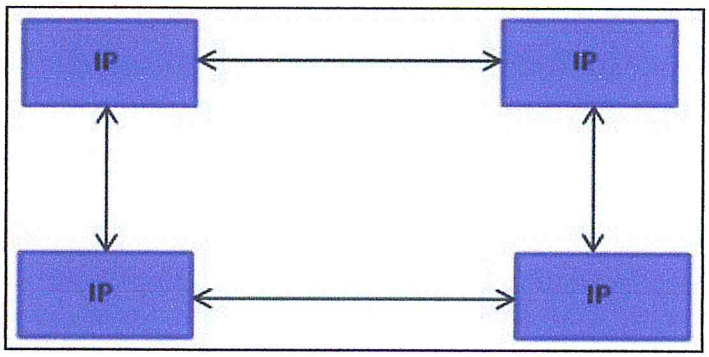

Figure9 : Topologie 2D maillée [13]

b. La topologie en tore :

C'est une topologie dérivé de celle des 2D maillée, elle à juste une caractéristique en plus qui est que, les bords extérieurs sont repliée sur eux-mêmes. Cette topologie offre une bande passante légèrement plus élevé comparé à celle de la 2D maillée, mais elle est plus difficile.

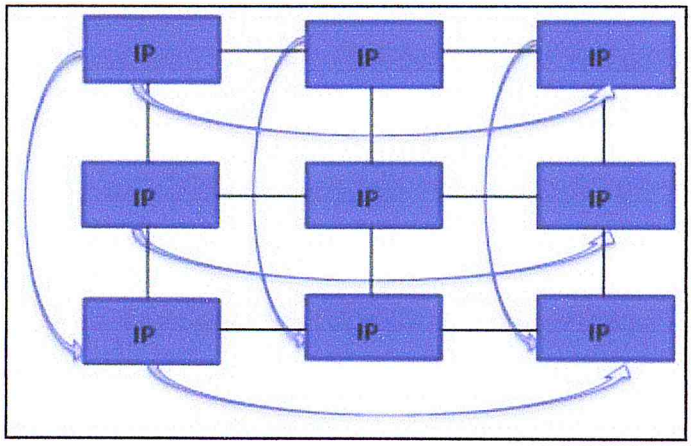

Figure10 : Topologie en tore [13]

c. <u>La topologie en anneau :</u>

Cette topologie comme son nom l'indique est structurée sous forme d'anneau, elle a la caractéristique d'être facilement intégrable sur silicium mais pas extensible car ses performances diminuent lorsque le nombre d'IP connecté augmente [8].

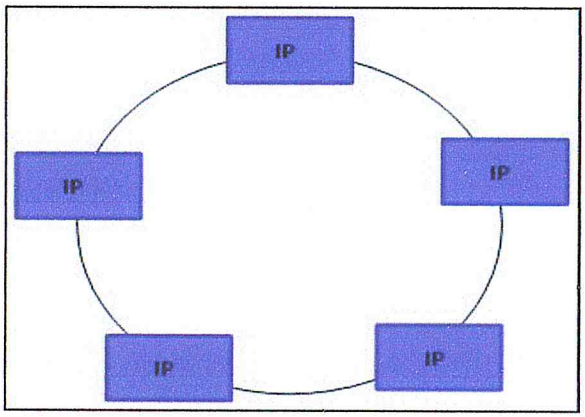

Figure11 : Topologie en anneau [14]

d. <u>La topologie en arbre</u> :

Cette topologie se base sur une hiérarchie de relation pères-fils, et a pour intérêt d'être scalable et d'offrir une latence qui peut être plus faible que la topologie 2D.

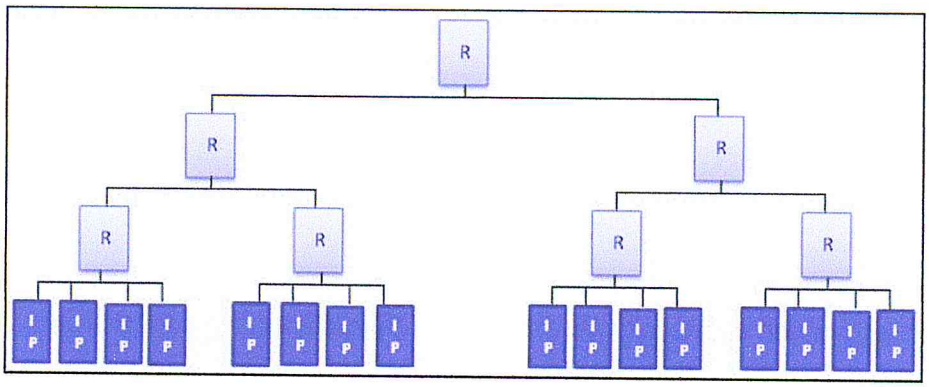

Figure12 : Topologie en arbre [8]

✓ Le mode de commutation :

Après avoir choisis la topologie du réseau il faut commencer a pensé comment faire pour réaliser la principale tâche d'un réseau qui est l'échange d'information de bout en bout en allouant les différentes ressources pour réaliser cette acheminement, et c'est ce qu'on appelle la commutation. Et pour gérer la commutation dans un réseau sur puce (NoC) il existe deux modes de commutation fondamentale et qui sont :

a. La commutation de circuit :

Dans ce type de communication on établit une connexion entre chaque source d'information et sa destination (émetteur/récepteur) [7] ce qui assure une grande bande passante entres ces deux points, et rends le système plus performant. Mais par contre dans ce mode de communication lorsque des ressources du réseau sont allouées pour le transfert, ces derniers sont réquisitionnés jusqu'à la fin de la connexion ce qui pénalise le système.

b. La communication de paquet :

Dans ce type de communication l'information à transmettre est découpée en paquets, et cela nous conduit à dire que la structure échangée au sein du réseau et le paquet. Ce dernier est envoyé sans qu'il y ait une connexion établie au préalable [7]. Le paquet est constitué de deux parties une partie qui s'appelle en-tête cette dernière contient les informations indispensable pour acheminer et reconstituer le message, et une autre partie données qui contient l'information à transmettre. Dans ce cas lorsque le paquet est envoyé sur le réseau le routeur analyse l'en-tête de ce dernier pour pouvoir l'acheminer et le transmettre à sa destination [8]. Ce principe de communication diminue la latence et augmente les performances du système [7].

Dans le mode de communication par paquet on a trois sous mode principaux [8], qui définissent comment les paquets transitent dans le réseau, et qui sont :

- Store and Forward (Stocker et propager) : dans ce type de communication les paquets qui transitent dans le réseau d'un nœud à un autre doivent être entièrement stockés dans ces nœuds avant d'être transmis. cela implique que chaque nœud doit être en mesure de stocker la totalité du paquet [8].

- Virtual cut-through : dans ce mode de communication un nœud peut commencer à envoyer le paquet au nœud suivant, même avant que ce premier (le nœud actuel) n'ai reçus le paquet en entier. Dans le cas ou le nœud suivant n'est pas disponible, le nœud actuel doit pouvoir stocké chez lui la totalité du paquet, et c'est l'inconvénient majeur de ce type de commutation [15].

- Wormhole : dans ce mode de communication on n'exige pas que tout le paquet soit stocké avant de l'envoyé, mais par contre on découpe le paquet en éléments de taille fixe appelé « flits » (flow control unit), et ces derniers passent d'un nœud a un autre des qu'il y a de la place pour un flit. Le flit d'en tête contient des informations de contrôle, parmi eux des informations sur la destination, lorsque ce derniers empreinte un chemin pour arriver a destination tout les autres flits qui constituent ce paquet suivent le même chemin [7].

Mode	Store and forward	Virtual cut-through	Wormhole
Taille mémoire du nœud	Paquet complet	Paquet complet	Fraction d'un paquet
Latence	Grande	Faible	Faible
Inter-blocage	Non	Non	Possible

Tableau 2 : Comparaison entre les différents modes de commutation [7]

✓ Les algorithmes de routage :

Ces algorithmes déterminent les nœuds qu'un paquet doit emprunter pour arriver à destination. Ils doivent éviter les situations d'inter blocage tout en utilisant les canaux de communications de manière optimale.

On a plusieurs types de routage parmi eux [7] :

- ✓ Le routage est dit source, si c'est l'émetteur qui définit le chemin de routage.
- ✓ Le routage est dit distribué si chaque nœud du réseau choisit la destination du paquet en fonction de son adresse cible.
- ✓ Le routage est dit déterministe s'il dépend seulement du destinataire de l'expéditeur sans tenir compte de la charge du réseau lors du routage.
- ✓ Le routage est dit adaptatif si les décisions de routage sont prises en fonction de l'état du réseau.

L'algorithme de routage le plus utilisé dans les réseaux sur puce, et celui qu'on a utilisé nous dans notre travail est l'algorithme XY. Cet algorithme de routage est de type déterministe et il garanti une solution pour toute situation de blocage [16].

Cet algorithme route les flits en premiers lieu dans la direction X et en suite dans la direction Y [17]. Si un saut est utilisé dans le NoC par un autre paquet, le flit reste bloqué dans le routeur jusqu'à ce que le chemin soit libéré. La figure suivante montre le principe de fonctionnement de cet algorithme

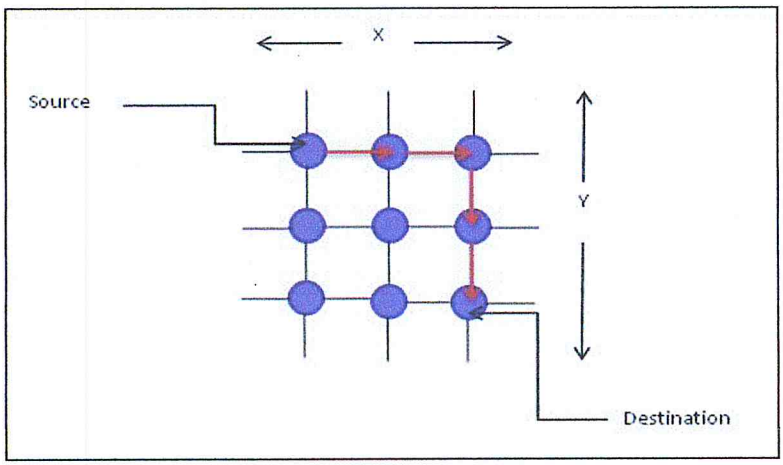

Figure13 : Routage ordonné X-Y [16]

- ✓ <u>Le contrôle de flux</u> :

Le contrôle de flux se basent sur des mécanismes qui évitent la surcharge du réseau et régulent le trafic. Pour réaliser ces fonctions les routeurs se basent sur des signaux de requêtes et d'acquittements.

On va citer deux stratégies de contrôle de flux qui sont :

- Handshake : quand un routeur envois des données il est en attente d'un acquittement qui lui permet de reprendre ses transactions. Ce mécanisme est simple de mise en place mais il augmente la latence du système car il a besoin d'au moins deux cycles d'horloge pour effectuer un seul transfert.

- Credit-based : avec ce mécanisme les données sont envoyées jusqu'à ce que les files d'attente du routeur récepteur se remplissent. Le routeur récepteur envois un message « credit » pour indiquer que ses files d'attente viennent de se libérées. Ce mécanisme est simple de mise en place, et améliore les performances du système puisque le transfert des données ne nécessite qu'un seul cycle d'horloge.

5. Quelques NoC académiques :

Dans la littérature il existe un grand nombre de réseaux sur puce, ici nous allons voir quelque un de c'est NoC

✓ SPIN :

C'est un réseau sur puce développé dans les années 2000 par le laboratoire LIP6 (Laboratoire d'informatique de Paris 6), dont son nom est l'abréviation de « Scalable Programmable Integrated Network » [8]. Ce dernier considérer comme la première proposition de réseau sur puce à commutation de paquets.

Ce réseau est fondé sur une topologie arbre quaternaire élargie ce qui conduit à réduire la latence du système. De plus cette structure aide à économiser de l'énergie vue qu'elle utilise un nombre de routeur réduit. Dans ce SPIN le contrôle de flux ce fait par le mécanisme based-credit sur des liens bidirectionnels.

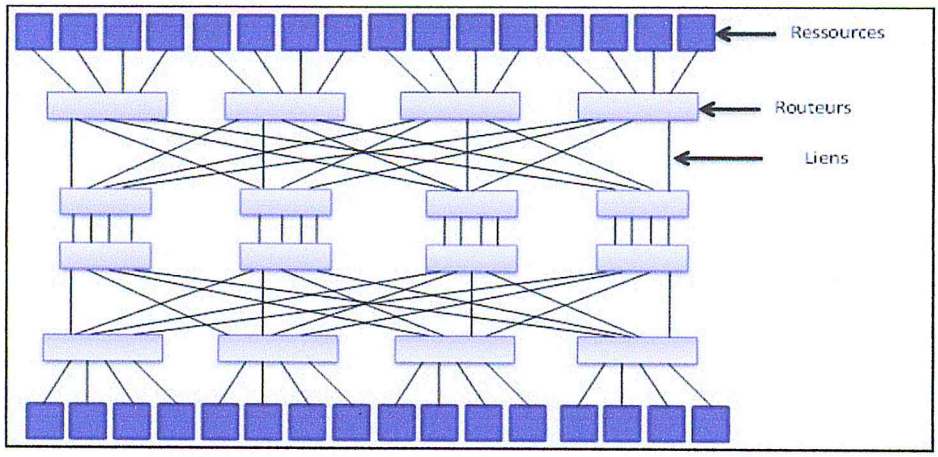

Figure14 : Architecture du NoC SPIN [18]

✓ ANoC :

Le Asynchronous NoC d'où vient l'abréviation ANoC, est une architecture proposé par « Laboratoire d'Électronique et de Technologies de l'Information de Grenoble » LETI. Ce réseau a été utilisé pour construire un réseau sur puce qui gère les besoin des applications de télécommunications. La structure de ce réseau est flexible et performante en termes de latence et de débit, il est disposé en grille de 2D et utilise une commutation de paquet wormhole [15].

✓ QNoC :

Le réseau QNoC (Quality-of-service NoC) qui est propose par l'institut technologique du Technion (Israël) [site1], utilise une topologie en grille 2D et assure une commutation de paquet wormhole [19]. Ce dernier fonctionne avec des échanges synchrones, comme il peut fonctionner avec des échanges asynchrones [11].

✓ Hermes :

Le réseau Hermes est développé par l'équipe F.Moraes Brezil [site2], est un réseau sur puce avec un type de commutation de paquet wormhole, un algorithme de routage X-Y, sur une grille 2D dont la taille peut être fixée en fonction des préférences utilisateur [13].

6. Les limites des réseaux sur puce :

Malgré tout ce qu'on a vu de bien sur les réseaux sur puce, et ce qu'ils ont apportés aux anciens structures d'interconnexion mais ils ont eux aussi leurs limites et points faible parmis eux on peut citer :

✓ Cohérence des caches :

Quand une donnée est partagée, chaque ressource (IP) stock une copie de cette donnée dans sa mémoire cache et effectue des opérations sur sa copie, ce qui conduit à un problème de cohérence des caches.

Pour remédier à ce problème on trouve deux techniques de cohérence de cache [20] :

- Cohérence par espionnage : Le protocole de cohérence par espionnage utilise un bus commun, ce qui permet à chaque ressource de toutes les requêtes émises sur le bus.
- Cohérence par répertoire : Le protocole de cohérence par répertoire utilise une structure supplémentaire qui est un répertoire centralisé, permet de mémoriser l'état actuel des caches.

✓ La fiabilité :

Les NoCs doivent garantir une bonne transmission et une intégrité des données transférées. Des technique de détection et de correction des erreurs sont mises en place pour résoudre ce problème [8].

✓ L'ordre de communication :

Dans les réseaux sur puce qui utilisent le type de commutation de paquets, l'ordre de l'envoi des messages peut ne pas être conservé, ce qui conduit dans ce cas a l'obligation de mettre en place des mécanismes de réorganisation des messages à leurs réceptions [21].

✓ La conception :

La conception d'une architecture de réseaux sur puce qui satisfait tous les besoins d'une application donnée est un processus très complexe. Afin de résoudre ce problème et réduire le temps de mise sur le marché, il est important d'automatiser la plupart des phases de conception.

7. Conception des réseaux sur puce :

Un réseau sur puce permet d'interconnecter plusieurs modules sur une puce.

Il représente une solution aux types de communications actuelles, mais ça conception reste difficile et délicate. Pour y remédier a ça il faut automatiser la plus part des tâches de conception par des méthodes et des outils.

a. Problèmes de conception des réseaux sur puce :

La complexité des fonctionnalités intégrées dans les systèmes sur puce impose différentes contraintes sur la mise en œuvre des réseaux d'interconnexion. Ces contraintes peuvent se présenter à plusieurs niveaux :

- Au niveau application, le défi principal consiste à exploiter le parallélisme des tâches et de capturer les communications concurrentes dans les modèles de calcul [7].
- Au niveau architecture, plusieurs topologies sont explorées afin de choisir celle qui répond le mieux aux exigences et aux besoins de l'application ciblée. Ceci peut être complexe et très long à effectuer.
- Au niveau communications, l'application doit être placée sur l'architecture de telle sorte que les coûts soient minimisés. Cette procédure est délicate car un mauvais placement peut engendrer une violation des contraintes de bande passante ou de temps réel.

b. Phase de conception d'un réseau sur puce :

Le processus de conception d'un réseau sur puce efficace qui satisfait les contraintes les contrainte de l'application est très complexe. Il passe par plusieurs niveaux, du niveau de modélisation jusqu'à l'implémentation physique.

La conception d'un NoC passe par les phases suivantes [22] :

- ✓ Modélisation du trafic.
- ✓ Détermination de la topologie NoC.
- ✓ Assignation des tâches aux nœuds de l'application.
- ✓ Ordonnancement des tâches.

- ✓ Placement des tâches sur topologie.
- ✓ Allocation des chemins de routage et réservation des ressources.
- ✓ Vérification des performances.
- ✓ Développement des modèles de synthèse et de simulation.

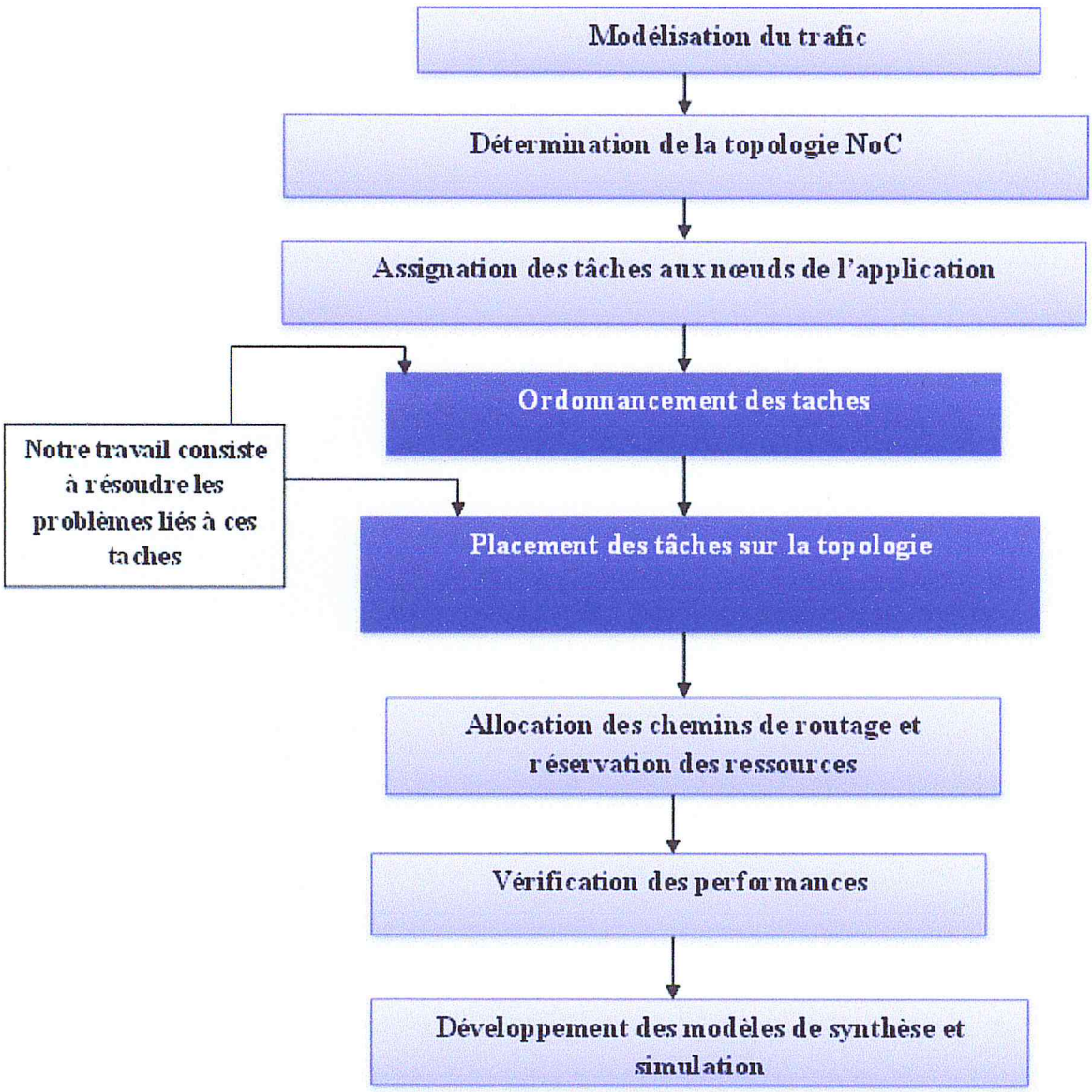

Figure15 : Phases de conception d'un NoC

c. Outils d'aide à la conception des réseaux sur puce :

Voici quelques outils qui automatisent différentes phase de la conception des NoC.

✓ SUNMAP [23]:

Cet outil est utilisé pour la conception des réseaux sur puce, son principe se base sur la génération d'une topologie pour une application donnée fin de minimiser l'utilisation de la bande passante. Pour cela il exécute le mapping de l'application sur différentes topologie et choisi celle qui fournit le meilleur résultat en cout de communication.

✓ NoCExplorer :

NoCExplorer [site3] est un environnement conçu par l'entreprise Artéris. Il permet d'explorer l'application et de récupérer ses besoins de communication entres les blocs IPs, pour analyser les différentes topologies qui peuvent satisfaire les contraintes [site4].

✓ XpipesCompiler :

Il réalise la génération du code SystemC pour la simulation ainsi que le code pour la synthèse en se basant sur une bibliothèque de composants et de paramètres définis.

8. Conclusion :

Dans ce chapitre nous avons essayé de présenter les réseaux sur puce (NoC) qui sont venu pour remédier et résoudre les problèmes rencontrés dans les structures d'interconnexions des SoC, on a défini les Noc, on a vu un par un les composants d'un NoC, ainsi que ses caractéristiques, on a cité quelques réseaux sur puce académiques, ce pendant aucune solution n'a été commercialisée, ceci est du à la complexité de conception comme on l'a dis précédemment. On a aussi vu la conception des réseaux sur puce.

Dans le chapitre suivant on va voir les problèmes liés aux deux phases de conception sur les quelles notre problème se base.

Chapitre III : Problème de mapping et d'ordonnancement

1. Introduction :

Dans un système multi processeur des questions se posent, quelle tâche sera exécuter, sur quel tuile, à quel moment? Cela correspond à l'ordonnancement et le Mapping.

C'est une étape importante du processus de conception du NoC qui traite la mise en œuvre de l'application sur une architecture spécifié. La phase ordonnancement est l'une des étapes les plus importantes, il s'agit de fournir la séquence dans laquelle les tâches d'une application s'exécute tandis que la phase Mapping permettra de placer les tâches dans l'architecture de l'application.

Le problème d'ordonnancement et de Mapping constituent un des principaux domaines d'étude de la recherche opérationnelle. De bon algorithmes et techniques sont proposés pour avoir un maximum de performance pour une application donnée.

Dans ce chapitre on va définir dans un premier lieu le Mapping ces techniques et méthodes, on passera après a l'ordonnancement et ces algorithmes, a la fin on va étudier et comparer divers approches de mapping et ordonnancement sur des architecture NoC.

2. Définition du Mapping :

Le Mapping est une étape cruciale dans l'exploration de l'architecture dans les NoCs, ça consiste à placer des tâches en outre a affecter chaque IP de l'application à une ressource de l'architecture (tuile) tout en respectant les contraintes imposées (minimisation des couts de communications, bande passante, etc..) [24]

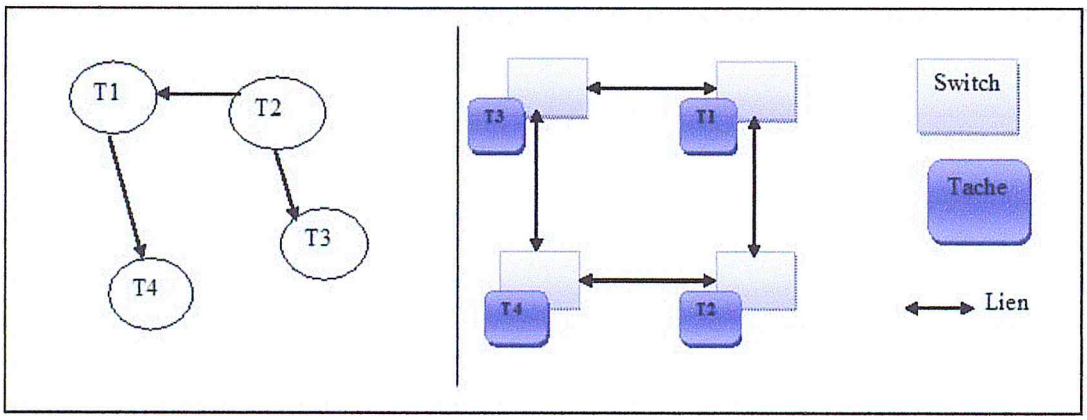

Figure16 : Schéma explicatif du principe du placement

3. Type de Mapping :

a. Mapping Statique :

L'un des principaux avantages de ce type de Mapping est la vue globale du système. Il est adapté à des plateformes spécifiques ou les IPs sont affecté aux tuiles avant l'exécution de l'application. [25]

b. Mapping Dynamique :

Il fournit généralement une vue local du système, les IPs sont affecté durant l'exécution de l'application, une tâche peut être ajouté, supprimé ou remplacer.

Le Mapping dynamique peut offrir de meilleur performance et apporte des avantage au système mais il reste difficile a tester. [25]

4. Définition d'un problème de Mapping :

Le problème de Mapping est l'un des problèmes NP-Difficile, puisque si on place « m » IPs sur « n » tuiles, si le nombre de tuiles « n » et le nombre d'IP « m » devient grand, le nombre de solution possible devient exponentiel. [24]

5. Méthodes de Mapping:

Pour résoudre ce problème, deux classes de méthodes sont utilisées, il s'agit de la classe des méthodes exactes et la classe des méthodes approchées.

a. Méthode exacte :

Cette méthode garanti de trouver un ensemble de solutions optimal mais l'inconvénient qui se pose est que le temps d'exécution sera augmenter d'une façon exponentiel surtout si la taille du problème est trop grande. [26]

b. Méthode approchée :

Cette méthode garantie de trouver une solution envisageable en un temps réduit même lorsque la taille du problème a traité est grande. [27]

Ce type de méthode peut être classé en différentes catégorie :

- Méthode Constructive.

- Recherche locale (recherche tabou, recuit simulé).
- Méthode évolutives telle que les algorithmes génétiques et les fourmis artificielle.
- Les heuristiques.

6. Techniques de Mapping existantes:

Il existe beaucoup de techniques pour résoudre un problème de Mapping, voici quelques unes qui sont spécifique pour les réseaux sur puce :

a. Technique de BB (Branch and Bound) :

C'est un algorithme d'optimisation déterministe appliqué sur le placement d'IPs sur une topologie de 2 D ayant pour objectifs de minimiser le cout de communication, sont but est de trouver dans l'arbre du graphe la feuille ayant un cout d'énergie le plus faible possible. [8]

b. Technique de NMAP (Network Mapper) :

C'est un algorithme rapide, il permet de placer les IPs sur une topologie de 2D maillé, cette technique permet de minimiser le cout de communication, elle procède en trois étape :

- placer les IPs(1) qui ont une communication maximale avec leur voisin.
- placer les IPs(2) qui communique le plus avec les IPs(1) placé.
- placer le reste des IPs avec les IPs(2) déjà placé. [28]

c. Technique CGMAP :

Son but est de minimiser le cout de communication, elle combine entre deux algorithmes (algorithmes génétique et chaotique), elle utilise les algorithmes génétique pour pouvoir trouver l'espace des solutions réalisable facilement tandis que l'algorithme chaotique affinent les recherches trouvé. [29]

d. Technique PMAP(PhysicalMapping) :

Elle est faite en deux phases, la première consiste à placer les IPs qui communique le plus entre eux sur des tuiles adjacente, la deuxième phase consiste à placer le reste des IPs en fonction des IPs déjà placer. [30]

e. Technique SPIRAL :

Son but est de minimiser le cout de communication et optimise le temps d'exécution, elle est basé sur le fait que les tâches de l'application sont affecté au ressources d'une manière Spirale (en démarrant du centre pour atteindre les tuiles frontières). [31]

f. Technique de PLBMP :

Son but est d'optimiser le l'énergie, elle est basé sur l'algorithme essaim particulier (PSO) ou la meilleure solution est une particule. [32]

Les techniques vues auparavant permettent d'optimiser plusieurs objectifs, notre travail consiste à optimiser le cout de communication et le temps d'exécution.

La phase Mapping nous permettra de calculer et d'optimiser le cout de communication tandis que la phase ordonnancement permet de calculer le temps d'exécution.

g. Technique de GBMAP :

Elle est appliquée sur une topologie maillé, l'objectif principale de cette technique est de réduire la bande passante nécessaire, elle est basé sur l'algorithme génétique, dont sa phase d'évaluation est basé sur plusieurs paramètres de largeur de la bande passante. [33]

h. Technique Onyx :

C'est une méthode avec moins de complexité par rapport a d'autre. Elle définit quatre mouvements pour assigner des priorités aux tuiles sur un chemin en forme de losange, elle minimise le nombre de saut entre les cœurs IP dans le but d'optimiser la consommation d'énergie et le cout de communication. [34]

i. Technique Citrine :

Son but est de minimiser le cout de communication dans les Noc. Elle est composé de deux étapes.la première étape consiste a utiliser la technique Onyx pour récupérer l'ordre des noyaux qui doit être placer tandis que la deuxième étape consiste a utiliser la méthode Branch and Bound qui essaie de chercher d'autre permutation parmi ceux définis dans la règle en forme de losange de Onyx.[35]

7. Définition de l'ordonnancement :

Un ordonnancement est une séquence d'exécution de tâches qui vise à satisfaire un ou plusieurs objectifs. Dans un autre terme c'est définir l'ordre d'exécution des tâches quand un ou plusieurs tâches doivent être exécuté sur un ou plusieurs processeurs matériels et /ou logiciels, un algorithme d'ordonnancement doit être appliqué pour décider de l'ordre d'exécution.

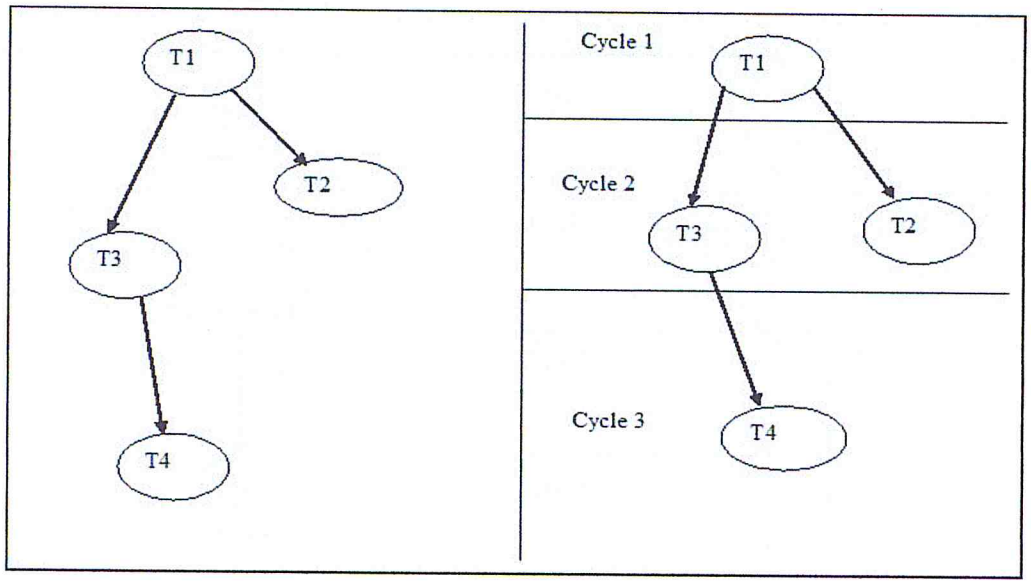

Figure17 : schéma explicatif du principe d'ordonnancement des tâches

8. Définition d'un problème d'ordonnancement :

Le problème d'ordonnancement constitue un des principaux domaines d'étude de la recherche opérationnelle, il est classé dans la catégorie des problèmes NP-Complets. On définit le plus souvent un problème d'ordonnancement comme étant formé d'un ensemble de tâches dont on cherche à déterminer ses dates d'exécution, ceci doit se faire en général en optimisant certain critères. [36]

9. Les Principaux Algorithmes d'Ordonnancement :

Au cours des années, les chercheurs ont essayé d'appliquer différents type de solutions aux problèmes d'ordonnancement dans les réseaux sur puce. Plusieurs algorithmes sont disponibles, de ce qui suit nous citons quelque uns :

Les entrées de cette phase sont : [1]

- Un modèle d'application,
- Un modèle d'architecture,
- Des contraintes et fonctions objectifs à optimiser.

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

a. Modèle d'application :

L'application est généralement donnée par un graphe de tâches (TG : Task Graph) qui est un graphe orienté G(T, E) ou chaque nœud représente une tâche de l'application et chaque arc représente la communication. Un nœud peut être caractérisé par :

- Complexité : Définit la structure de la tâche qui peut être:- un composant de calcul.
  -un ensemble de composant.
  -un autre graphe de tâches.
- Taille : Définit la taille de la tâche en cycle.
- Energie : Définit la consommation d'énergie par cycle.
- Donnée : la quantité de donnée d'entré et de sortie et à quel moment elles sont produite.
- Communication : elle représente l'interaction entre les tâches du même graphe.

b. Modèle d'architecture :

Ce modèle est généralement un graphe orienté P(S, F) ou il existe deux types de nœuds qui représentent les processeurs et les Switch qui sont interconnecté par des arcs représentant les liens de communication dans la plateforme. Chaque composants est caractérisé par :

- Architecture : Chaque architecture a ses propres caractéristiques qui ont un impact important sur sa performance comme la fréquence, mémoire...etc.
- Hétérogénéité/Homogénéité : Généralement les MPSOC sont globalement Hétérogène avec quelque nœud homogène.
- Energie : représente la quantité d'énergie consommée pour le traitement, le stockage…etc.
- Topologie : représente la structure de la plateforme.

c. Contrainte et fonctions objectifs :

Un Mapping et ordonnancement réalisable est celui qui satisfait les contraintes spécifiée telle que les contraintes de temps, contrainte de taille mémoire…etc. Un Mapping et ordonnancement de qualité est celui qui en plus de satisfaire les contraintes, doit optimiser les fonctions objectifs spécifique telle que la minimisation du cout de communication, et le temps d'exécution.

Dans la suite nous exposons quelques approches de Mapping et ordonnancement qui existent.

11. Les approches existantes :

a. Approche de G.Varatkar et R.Masculescu :

Ils ont développé une méthode pour minimiser la consommation d'énergie totale dans le Noc. Le Mapping et l'ordonnancement des tâches est traité simultanément en utilisant LS en cherchant à réduire la consommation d'énergie par minimisation du volume de communication inter processeurs. [39]

b. Approche de T.Lei et S.Kumar :

Ils utilisent une méthode à deux étape, dans la première étape, ils utilisent l'algorithme génétique pour la phase Mapping ensuite comme deuxième étape, appliquent les techniques ASAP et ALAP pour l'ordonnancement des tâches. L'objectif du Mapping est de maximiser le temps de performance tandis que celui de l'ordonnancement est de respecter les contraintes de temps. [40]

c. Approche de J.Hu et R.Marculescu :

Ils ont développé une méthode d'optimisation qui cherche le plus court chemin des communications dans le Noc et calcule la consommation d'énergie. Elle a deux phases, la première utilise LS pour le Mapping et l'ordonnancement des tâches parallèles, la deuxième phase utilise une méthode déterministe pour le Mapping et l'ordonnancement des communications parallèles. [41]

d. Approche de D.Shin et J.Kim :

Ils visent à réduire la consommation d'énergie. Cette méthode est basée sur l'algorithme génétique pour le Mapping des tâches, et utilise « LS » pour l'ordonnancement. Le placement des tâches tend à minimiser le volume de communication inter processeurs tandis que l'ordonnancement réduit la consommation d'énergie des liens. [42]

12. Conclusion :

Le problème de Mapping et ordonnancement est l'un des sujets qui se traite de plus en plus dans le domaine informatique. Nous avons présenté dans ce chapitre différentes notions concernant le Mapping et l'ordonnancement, leur technique et algorithmes, ainsi un état d'art ou nous avons constaté que dans les travaux vus et surtout ceux fait récemment, on ne traite presque jamais le Mapping sans l'ordonnancement et vise a versa .Tous les travaux abordent les ensembles, seulement la manière change, soient ils sont traiter séquentiellement sois ils sont traité simultanément, d'une autre part ,pour chacune des approches vus il y'a des avantages comme des inconvénients mais a la fin toutes les approches tende a optimiser un ou plusieurs objectifs, c'est ce qu'on appelle l'optimisation qui sera présenter dans le chapitre suivant ainsi quelques méta heuristiques connues dans ce domaine.

Chapitre IV : Optimisation

1. Introduction:

L'optimisation joue un rôle très important dans le domaine informatique, elle consiste généralement à optimiser une fonction objective. On va donner dans ce chapitre une idée générale sur l'optimisation combinatoire, la signification d'un problème combinatoire.

Ensuite on va entamer les heuristiques ainsi que les métas heuristiques. Enfin, on va donner quelques descriptions sur l'ensemble des méthodes d'optimisation combinatoires et les méthodes des méta-heuristiques couramment utilisées en optimisation.

2. Définition d'optimisation combinatoire :

L'optimisation mathématique est une science qui consiste à sélectionner un meilleur élément d'un certains ensembles d'alternatives disponible par rapport a certains critères. D'une autre manière c'est de trouver la meilleure solution réalisable pour un problème donnée. Dans le cas le plus simple un problème d'optimisation combinatoire consiste à minimiser ou maximiser une fonction réelle en choisissant systématiquement des valeurs d'entrée à partir d'un jeu autorisé et le calcul de la valeur de la fonction. Ainsi, les problèmes d'optimisation combinatoire sont en général très coûteux à résoudre de façon optimale. La solution optimale à un problème d'optimisation ne peut que très rarement être déterminée en un temps polynomial. [43]

Il existe plusieurs méthodes d'optimisations : [44]

- les méthodes complètes : on obtient toujours une solution.
- les méthodes optimales : on trouve toujours la meilleure solution (optimum global).
- les méthodes exactes : explore l'espace de recherche dans sa totalité (optimal).
- les méthodes approchées (approximative) : explore une sous partie de l'espace de recherche.
- Les méthodes déterministes : exécute toujours la même suite d'opérations.
- Les méthodes probabiliste (ou stochastique) : fait des choix probabilistes guidée par des tirages aléatoire.

Voici un schéma qui résume tout cela :

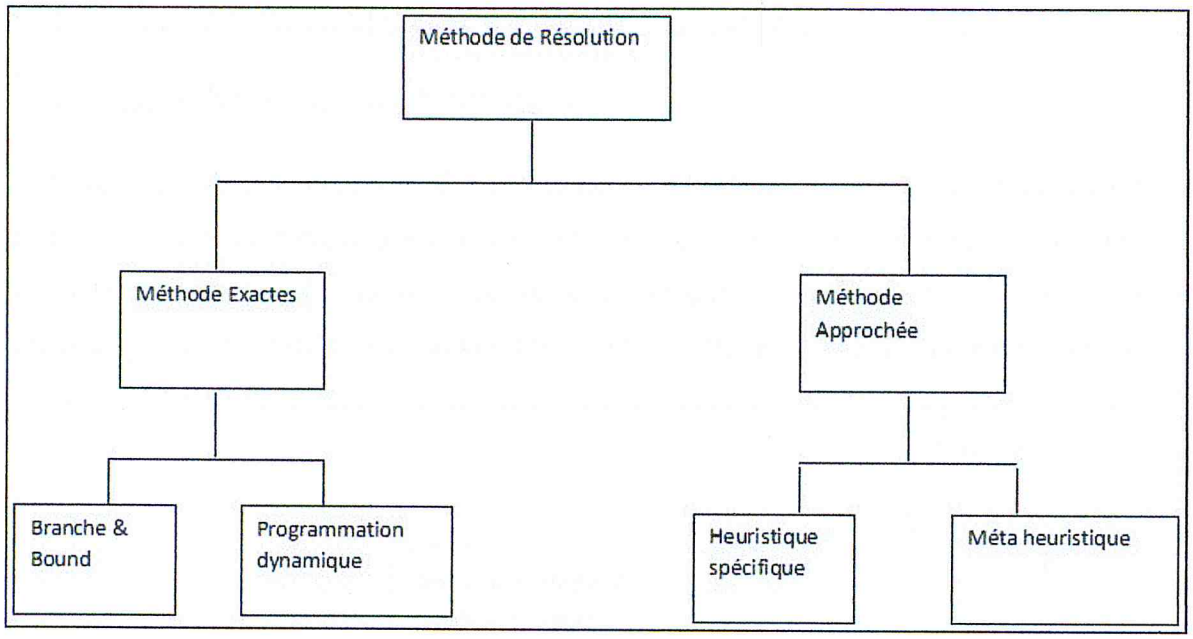

Figure20 : Méthodes d'optimisation [45]

Les méthodes exactes ne sont efficaces que pour les instances de problèmes de petite taille, on va s'intéresser qu'aux méthodes approchées.

3. Définition d'une heuristique :

Une heuristique est une méthode approchée qui est basées sur l'expérience et les résultats déjà obtenue. Elle n'aboutie pas nécessairement a une solution optimal. [46]

4. Définition d'une méta-heuristique :

Les méta-heuristiques est une heuristique généraliste, pouvant s'appliquer à plusieurs problèmes d'optimisation, c'est une méthode générique qui peut atteindre des solutions en temps raisonnable à des problèmes combinatoire difficiles (exponentiel). Elle se comporte comme des algorithmes de recherche, tentant d'apprendre les caractéristiques d'un problème afin d'en trouver une approximation de la meilleur solution d'une manière proche des algorithmes d'approximation. Il existe de nombreux métas heuristiques ceux qui sont simple et ceux qui utilisent des algorithmes plus complexes, on parle alors des différentes méthodes de la méta-heuristique. [46]

candidate a le choix de la solution vers laquelle se déplacer en utilisant les informations des solutions voisines de la solution courante.

Le processus s'arrêt lorsque la solution courante trouvée par l'algorithme n'a pas été améliorée depuis un nombre donné d'itérations ou parce que le nombre maximal d'itérations qui était fixé au départ est atteint.

Algorithme générale de la recherche local : [49]

Générer une solution initiale s et poser s*=s

Tant qu'aucun critère d'arrêt n'est satisfait faire

1-Générer une solution s' dans le voisinage N(s) de s.

2-Poser s=s' et mettre à jour s* si f(s)<f(s*).

3- chaque itération de f : f (s')= min f (s").

4-critère d'arrêt : si f ne s'améliore pas entre deux itérations.

- Recherche Tabou :

La recherche Tabou est une méta-heuristique développer par Fred Glover 1986 [50] qui combine une procédure de recherche locale avec un ensemble de règles lui permettent de surmonter l'obstacle des extremum locaux ainsi que les problèmes de cycles, elle est destinée principalement a guider d'autre méthodes afin d'en trouver les meilleures solutions a partir d'une solution optimale obtenue par l'une des heuristique.

Elle est utilisée dans divers domaines d'applications tel les Problèmes de transport, la planification et ordonnancement, optimisation de graphes, les télécommunications, la logique et intelligence artificielle.

Le principe de cette méthode se base sur la notion de voisinage a partir d'une position X donnée, tel que cet position doit minimiser la fonction objective, dans un problème de minimisation on choisi le minimum local lorsque les point de voisinages ont une valeur plus élevé, après des itérations on peut retomber sur le même minimum local et pour éviter cela on doit avoir une mémoire ou on doit mettre les dernières positions explorer dans une file FIFO [51] .Une nouvelle solution sera acceptée seulement si elle n'appartient pas à cette liste.

Algorithme générale de la recherche tabou [52]

1-Initialisation.

2-Créer une liste des mouvements candidats.

3-Choisir le meilleur candidat. Ce choix est basé sur les restrictions Tabou et le critère d'aspiration.

-On obtient ainsi une autre solution, mais qui ne sera enregistrer que si elle est meilleur que la solution précédente.

4-Appliquer le critère d'arrêt.

-Continue: changer les candidats d'admissibilité (restriction Tabou et critère d'aspiration). Aller à 2.

-Stop: passer aux stratégies d'intensification et diversification.

- Recuit simulé :

Le recuit simulé est une méta-heuristique qui était mise en œuvre par les trois chercheurs S.Kirkpatrick, C .D .Gelatt et M.P.Vecchi de la société IBM en 1983, puis en 1985 par le chercheur V.Lery [53]. Le principe de cet algorithme c'est de parcourir de manière itératif l'espace de solutions, en débutant par une solution initial, et après itération on obtiendra une seconde solution, elle sera sois accepter et c'est avec cette dernière qu'on calculera la prochaine itération ou refuser et cela en suivant certains critéres.si la solution trouver améliore la fonction objective elle aura permis de diminuer l'énergie du système ; si celle-ci la dégrade, l'énergie du système augmentera (recuit). Le même procédé est répété itérativement jusqu'à ce qu'une condition d'arrêt soit vérifiée.

Les meilleurs voisins ont une probabilité plus élevée, et les moins bons ont une probabilité plus faible. On utilise un paramètre, appelé la température (notée T), si la température est élevée alors tous les voisins ont à peu près la même probabilité d'être acceptés, sinon si elle est faible, un mouvement qui dégrade la fonction de coût a une faible probabilité d'être choisi. La température varie au cours de la recherche, elle est élevée au début, puis diminue et finit par tendre vers 0. Si la température a atteint un seuil assez bas fixé au préalable ou que le système devient figé, l'algorithme s'arrête.

Algorithme général de recuit simulé [54]

Engendrer une configuration initiale S0 de S ; S := S0

• Initialiser T en fonction du schéma de refroidissement

• Répéter

– Engendrer un voisin aléatoire S' de S

– Calculer D = f(S') – f(S)

– Si CritMetropolis(D, T), alors S := S'

– Mettre T à jour en fonction du schéma de refroidissement

• Jusqu'à <condition fin>

• Retourner la meilleure configuration trouvée

b. Méta heuristique à solution multiple :

➢ Algorithme évolutionnaire :

Les Algorithmes Évolutionnaires (AEs) sont des méthodes de recherche inspirées par la théorie darwinienne de l'évolution, travaillant sur une population de solutions potentielles, d'individus, de gènes (variables). Parmi les algorithmes évolutionnaire les plus connu, on trouve l'algorithme génétique.

➢ L'Algorithme Génétique :

Les algorithmes génétiques sont des méthodes évolutives qui s'inspirent fortement des mécanismes biologiques liés aux principes de sélection et d'évolution naturelle. Développer par Holland en 1997 [50]. Ils ont été adaptés à des contextes très varié.

Cet algorithme est constitué d'une population P de solution appelé individus, son principe général consiste à simuler l'évolution d'une population d'individus jusqu'à atteindre un critère d'arrêt. [47]

Ce type d'algorithme nécessite :

- Un codage de donnée : pour caractériser chaque individu de la population, on associe à a chaque point de l'espace de recherche une structure de donnée spécifique.
- La définition la fonction d'évaluation appelée aussi fitness définie selon des critères d'optimisation du problème.

- Une population initiale : qui regroupe un ensemble d'individus, dont dériveront les futures générations.
- Des opérations d'évolution de la population
- Un processus de sélection des individus les mieux adapté, qui sera appliqué sur chaque nouvelle génération d'individus.

- Les principes étape d'un algorithme génétique : [50]

1. Génération de la population initiale

2. Constitution d'une nouvelle population :

   a. mesure de l'adaptation de chacun des individus

   b. reproduction des individus en fonction de leur adaptation. Les plus performants se reproduisent en priorité.

   c. croisement de paires de séquences choisies aléatoirement

   d. mutation de séquences tirées de manière aléatoire

3. Critère d'arrêt. Il peut être défini en fonction du nombre de génération, de l'adaptation du meilleur individu, etc.

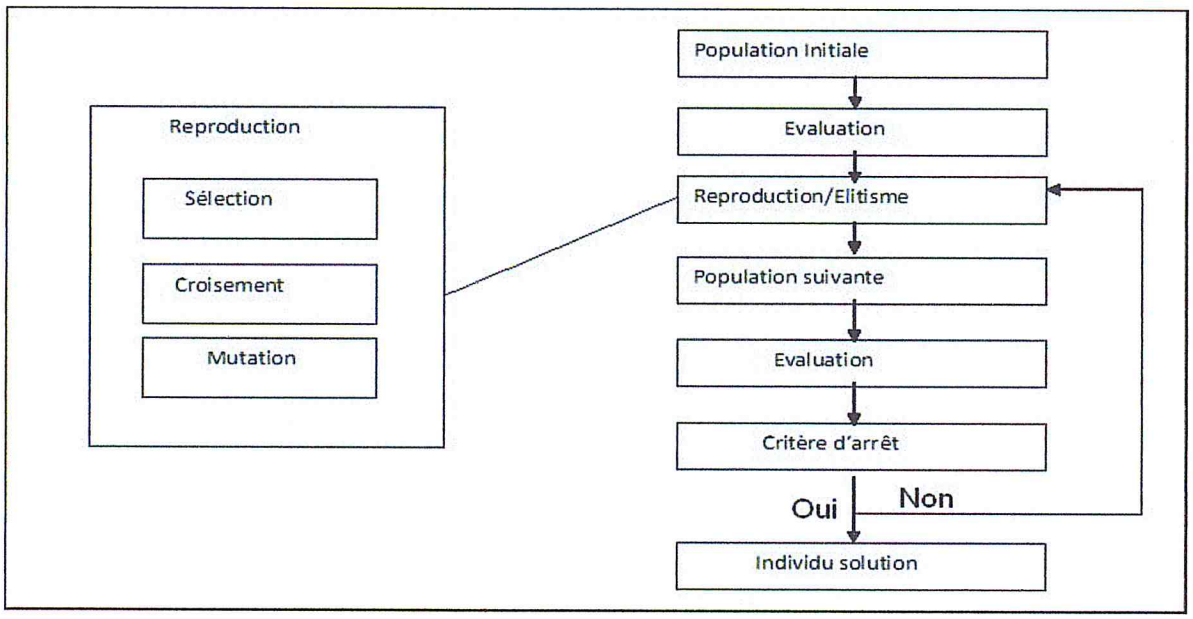

Figure22 : Les Etapes d'un algorithme génétique

✓ La population de base :

La population initiale celle qui servira de population racine a cet algorithme, la création de nouvelle génération de population se fait en trois étapes :

- Sélection
- Renouvellement (croisement)
- mutation

✓ La sélection :

La reproduction des meilleurs individus sera favorisée. Elle choisie parmi la population présente les futur parents nécessaires à l'étape de remplacement, elle permet d'identifier les meilleurs individus de la population courante qui seront autorisée à se reproduire, il existe plusieurs opérateurs de sélection :

- **Sélection par tournois** : c'est la méthode avec laquelle on obtient les résultats les plus satisfaisants, les individus sont choisi aléatoirement, ils seront confronter entre eux par le biais de la fonction d'adaptation, on sélectionnera ensuite le meilleur parmi eux, l'opération est répéter jusqu'à obtenir les N individu qui servirons de parents
- **Sélection par roulette** : le cas idéal d'application de cette méthode est celui ou la population est de taille infinie, cette méthode associe à chaque individu un segment, ce segment est ensuite concaténer sur un axe que l'on normalise entre 0 et 1 ,on tire un nombre aléatoire entre 0 et 1 puis on regarde quel est le segment sélectionnée on reproduit l'individu correspondant .Avec cette méthode les bon individu seront plus souvent sélectionnée que les mauvais , et un même individu pourra être sélectionner plusieurs fois.
- **Sélection par rang :** la sélection par rang est similaire à celle par roulette mais les proportions sont en relation avec le rang plutôt qu'avec la valeur de l'évaluation vus qu'elle est basé sur le trie de la population selon la fitness de chaque individus en attribuant un entier de 1 a N pour une population.

✓ Le croisement :

Le croisement permet la création de nouveau individus à partir du patrimoine génétique de parents. Cette reproduction à pour but d'engendrer les individus enfants mieux adapté que leur parents, les plus souvent deux enfants sont née par le croisement des deux parents sélectionner.

On trouve deux types de croisement :

L'algorithme général est relativement simple, et repose sur un ensemble de fourmis, chacune parcourant un trajet parmi ceux possibles. À chaque étape, la fourmi choisit de passer d'une ville à une autre en fonction de quelques règles [55] :

1. elle ne peut visiter qu'une fois chaque ville ;
2. plus une ville est loin, moins elle a de chance d'être choisie (c'est la « visibilité ») ;
3. plus l'intensité de la piste de phéromone disposée sur l'arête entre deux villes est grande, plus le trajet aura de chance d'être choisi ;
4. une fois son trajet terminé, la fourmi dépose, sur l'ensemble des arêtes parcourues, plus de phéromones si le trajet est court ;
5. les pistes de phéromones s'évaporent à chaque itération.

Cet algorithme a été adapté aux problèmes dynamiques, en variables réelles, aux problèmes stochastiques, multi-objectifs ou aux implémentations parallèles.

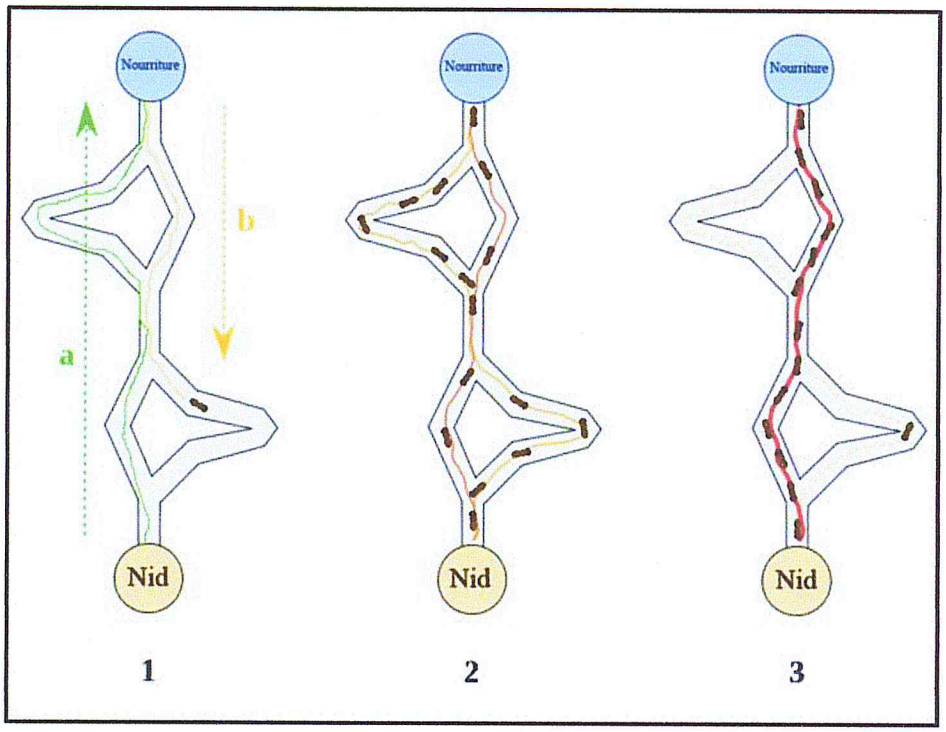

Figure23 : Principe de l'algorithme de Colonie de fourmis

➢ Essaim particulier :

- Principe :

Inspirer par le déplacement d'un groupe d'oiseaux, développer par Russel .Eberhart et James .Kennedy 1995 [site7], elle est basé sur la collaboration des individus entre eux, il partage de nombreux caractéristiques avec les algorithmes évolutionnaire comme les algorithmes génétique, Elle a d'ailleurs des similarités avec les algorithmes de colonies de fourmis, qui s'appuient eux aussi sur le concept d'auto-organisation. Cette idée veut qu'un groupe d'individus individuellement peu intelligents puisse posséder une organisation globale complexe.

Ainsi, grâce à des règles de déplacement très simples (dans l'espace des solutions), les

Particules peuvent converger progressivement vers un minimum local. Cette méta-heuristique semble cependant mieux fonctionner pour des espaces en variables continues.

Au départ de l'algorithme, chaque particule est donc positionnée, aléatoirement ou non, dans l'espace de recherche du problème.

Chaque itération fait bouger chaque particule, en fonction de trois composants : [46]

- sa vitesse actuelle
- sa meilleure position
- la meilleure position obtenue dans son voisinage

À l'aide de trois paramètres parfois appelés *coefficients de confiance*, qui pondèrent trois tendances :

- tendance à suivre sa propre voie

- tendance conservatrice (revenir sur ses pas)

- tendance « panurgienne » (suivre le meilleur voisin)

- Principe d'Algorithme d'essaim particulier : [22]

Pour une génération initiale un ensemble de solutions est aléatoirement choisi dans le domaine de la fonction à minimiser, et chaque particule aura une position et une vitesse.

A chaque génération, la fitness de chaque position sera calculer, et si G désigne la position du meilleur, pour chaque particule i, on regarde dans son voisinage pour chercher le meilleur n leader du groupe.

On mettra a jours la vitesse v_i de chaque particule i, et on déplacera les particules a leur positions Xi(t+1).on rebouclera sur le calcule de la fitness jusqu'à ce que le critère de fin sois vérifié.

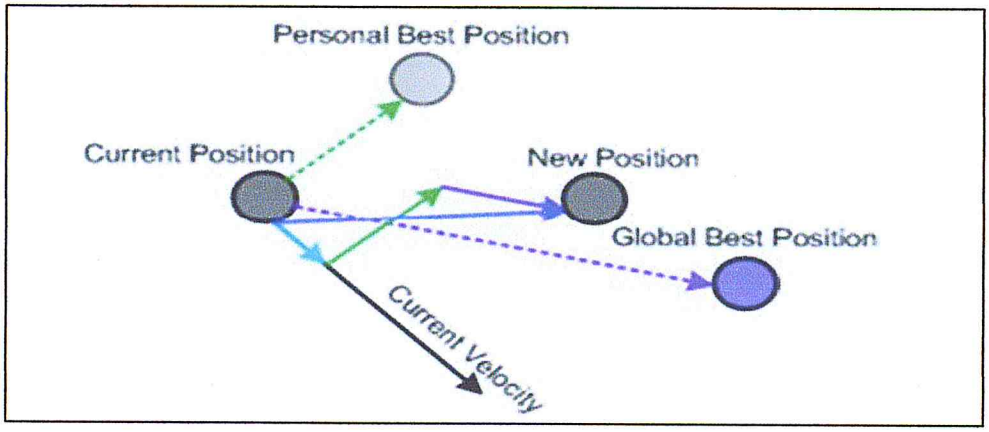

Figure24 : Le déplacement des individus « Essaim particulier » [56]

6. Conclusion :

Les métas heuristiques constituent une classe de méthodes approchées adaptables à un grand nombre de problèmes d'optimisation combinatoire.

Mais, si l'on a pu constater leur grande efficacité sur de nombreuses classes de problèmes, il existe en revanche très peu de résultats permettant de comprendre la raison de cette efficacité, et aucune méthode particulière ne peut garantir qu'une méta-heuristique sera plus efficace qu'une autre sur n'importe quel problème.

Concrètement, certaines méta-heuristiques présentent l'avantage d'être simples à mettre en œuvre, comme nous l'avons vu avec le recuit simulé ; d'autres sont plutôt bien adaptées à la résolution de certaines classes de problème, très contraints, comme le système de colonies de fourmis.

La qualité des solutions trouvées par les méta-heuristiques dépend de leur paramétrage et de l'équilibre à trouver entre un balayage de tout l'espace des solutions et une exploration locale poussée. Le choix d'une bonne représentation, d'un bon voisinage, sont également, nous l'avons dit, des facteurs influençant grandement l'efficacité de la méthode choisie, quelle qu'elle soit.

Dans le chapitre suivante nous allons présentés un algorithme évolutionnaire, qui nous permet de réaliser le mapping, et deux autre algorithmes d'ordonnancement, qui ont été utilisés pour la résolution de notre problème.

Chapitre V : Solution proposée

1. Introduction :

Actuellement les systèmes sur puce nécessitent des performances de plus en plus élevés que ce soit en termes de bande passante ou de ressources de calculs. Ainsi les techniques de communication par le biais du bus trouve leurs limites par rapport à la flexibilité et au besoin en bande passante qui augmente de manière considérable entre les IP.

C'est alors dans ce contexte que les réseaux sur puce sont apparus comme alternative aux techniques de communication par bus, et semble bien s'adaptés au besoin des futurs systèmes sur puce, ils offrent de meilleures performances en termes d'extensibilité et de bande passante, mais ils s'souffre d'un manque dans les outils qui aident qui automatisent certaine phase de la conception de ces réseaux sur puce, comme le mapping ou l'ordonnancement.

Nous utilisons ici une technique de mapping qui permet de placer les différents IP qui exécutent les tâches d'une application, sur une architecture NoC de taille donnée. Cette solution se base sur un algorithme évolutionnaire qui n'a pas été beaucoup utilisé auparavant dans le problème qu'on traite, et qui est l'algorithme à évolution différentielle « DE ».

Et deux techniques d'ordonnancement, qui permettent d'ordonnancer l'exécution des tâches d'une application donnée, qui se basent sur deux algorithmes d'ordonnancement « ASAP » (As Soon As Possible) et « ALAP » (As Late As Possible).

2. Présentation détaillée des techniques:

a. L'algorithme d'ordonnancement As Soon As Possible (ASAP): [38]

C'est un algorithme d'ordonnancement simple, sont principe générale est qu'une tâche ne peut être exécutée que si tous ses prédécesseurs issus du graphe d'application sont déjà ordonnancés.

L'algorithme ASAP (As Soon As Possible) commence par les nœuds les plus hauts du graphe d'application c'est-à-dire les nœuds qui n'ont pas de prédécesseurs, il les ordonnances dans un premier temps et en suite il vérifie tant qu'il reste des nœuds non ordonnancé, il cherche les nœuds dont leurs prédécesseurs sont ordonnancés et il les ordonnance, jusqu'à ce que tous les nœuds du graphe soient tous ordonnancés.

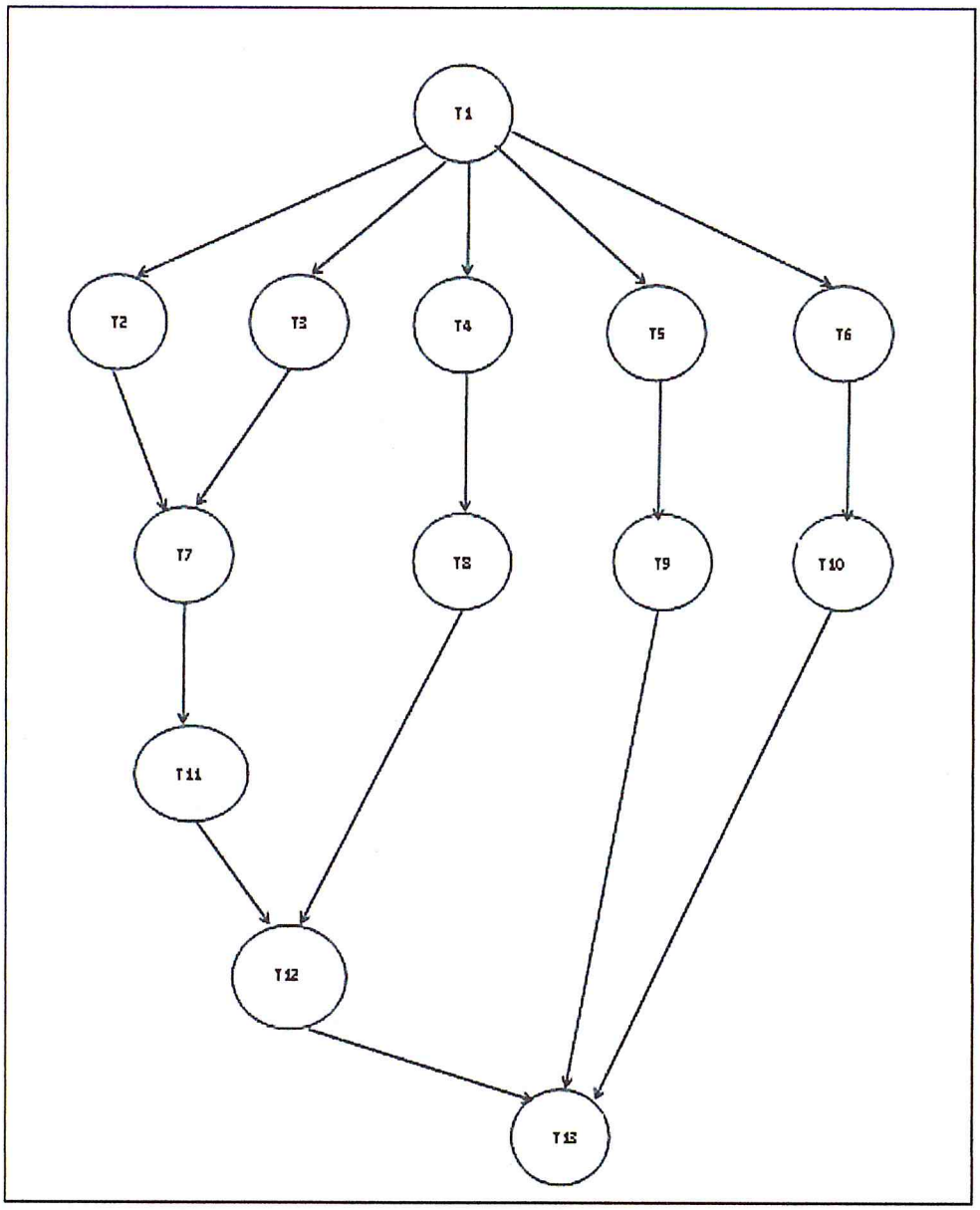

Figure25 : Graphe de précédence

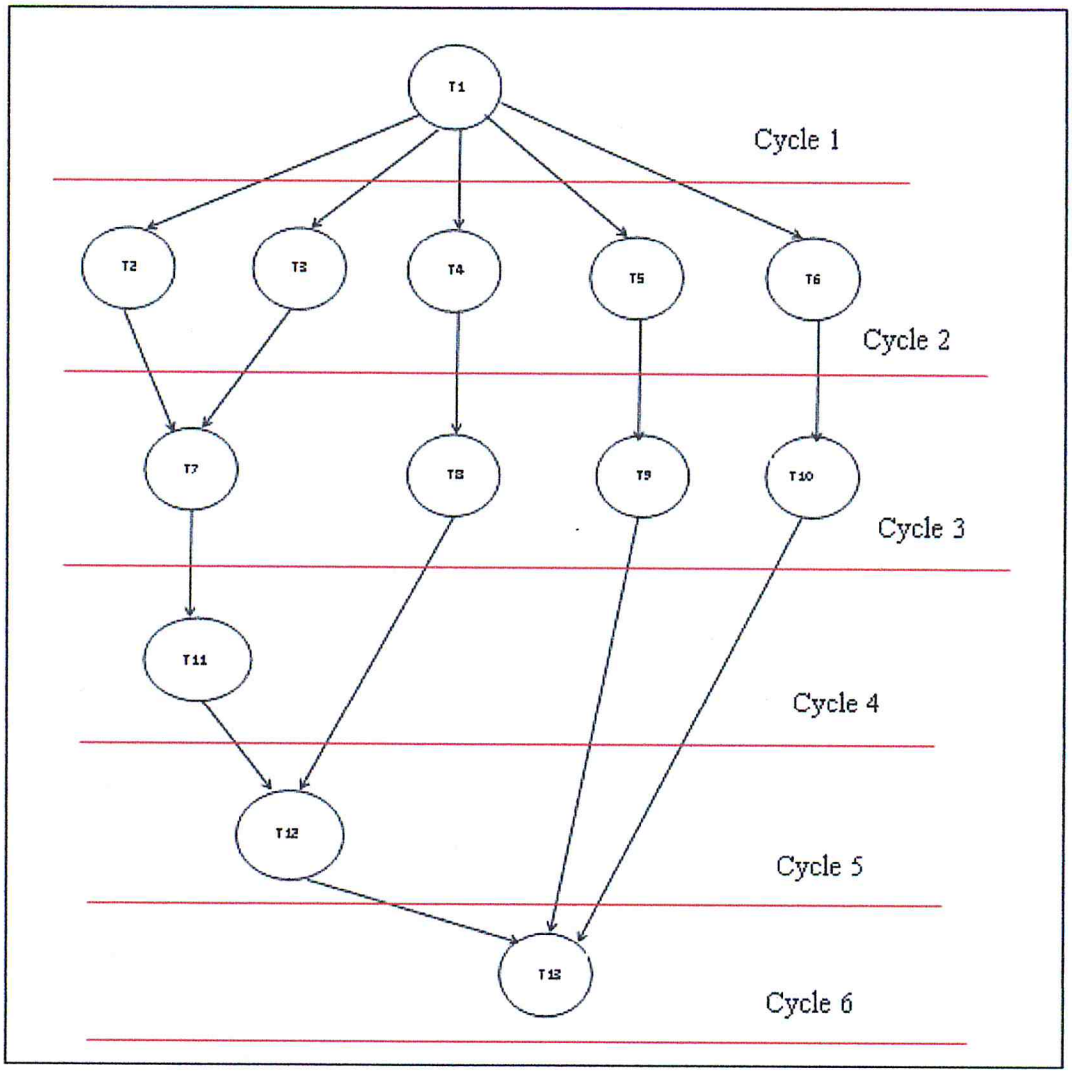

Figure26 : Application de l'algorithme ASAP sur le graphe de précédence

b. L'algorithme d'ordonnancement As Late As Possible (ALAP) : [38]

C'est aussi un algorithme d'ordonnancement simple, son principe général est contraire à celui de « ASAP », puisque il se base sur le fait que la tâche s'exécutera que si tous ses successeurs sont déjà ordonnancés.

C'est pour cela que l'algorithme ALAP commence par les dernières tâches du graphe d'application, puisque ce sont eux qui n'ont pas de successeurs il les ordonnance, en suite tant qu'il reste des tâches à ordonnancer, il ordonnance ceux qui ont les successeurs déjà ordonnancés, et ainsi de suite jusqu'à ce qu'il n'en reste plus tâche (nœuds) a ordonnancer.

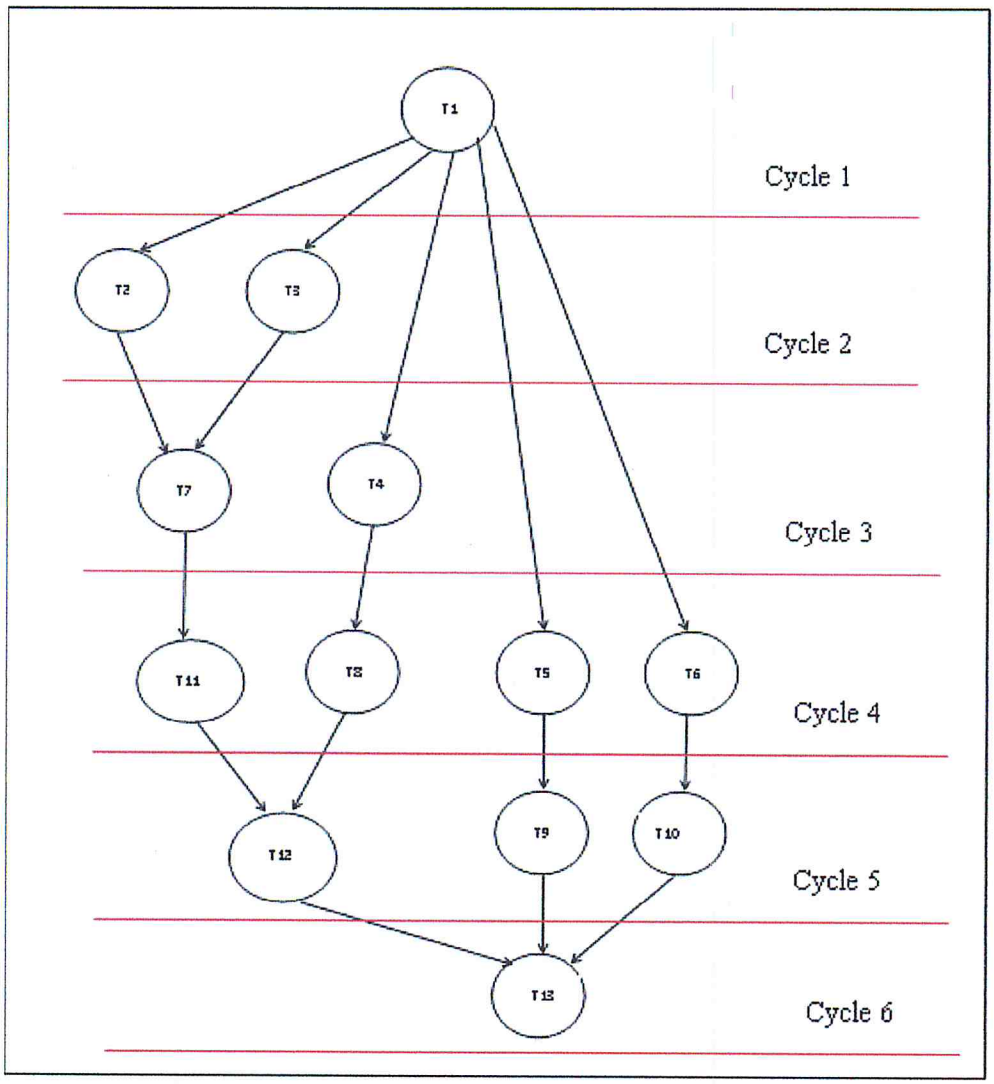

Figure27 : Application de l'algorithme ALAP sur le graphe de précédence

NB : Une petite remarque concernant les deux algorithmes d'ordonnancement ASAP et ALAP ils s'appliquent sur des graphes de précédence qui ne contiennent pas de circuit, parce que dans le cas ou le graphe contient un circuit on peut avoir un blocage au niveau de l'algorithme, où chaque nœud attends son successeur ou prédécesseurs à être ordonnancé pour qu'il s'ordonnance.

Voici un petit graphe de précédence explicatif pour le problème de circuit.

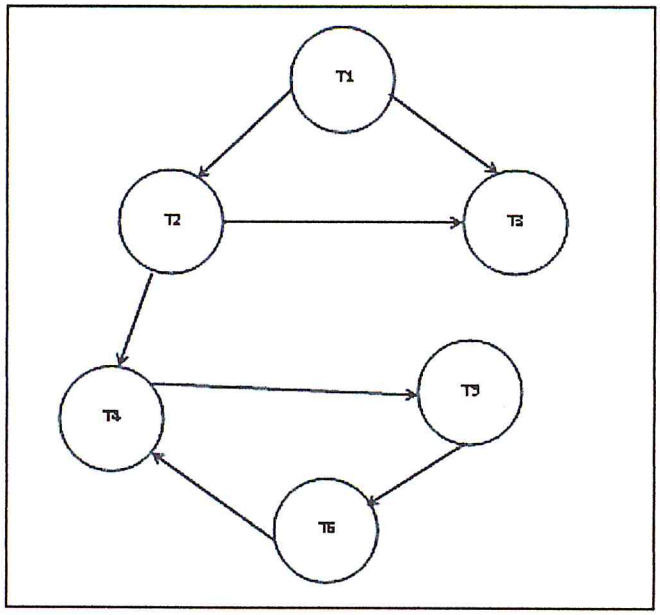

Figure28 : graphe de précédence contenant un circuit

- Si on essaye d'appliquer l'algorithme ASAP sur ce graphe de précédence on va obtenir ce résultat :

Cycle1 : T1

Cycle2 : T2

Cycle 3 : T3

Et ensuite l'algorithme bloque malgré qu'il reste des nœuds a ordonnancé et cela est du au circuit qui se trouve entre les nœuds (T4, T5, T6), ici le nœud T4 attends que son prédécesseur T6 soit ordonnancé, et le T6 attends le T5 et le T5 attends le T4. Et dans ce cas l'algorithme se bloc.

- Même chose pour ALAP si on le déroule on obtient ceci :

Cycle1 : T3

Et ici bloque l'algorithme, il ne peut pas ordonnancer le T1 parce qu'il faut que son successeur T2 soit ordonnancé, et T2 attend que son successeur T4 soit ordonnancé, T4 attend le T5, le T5 attend le T6 et le T6 attend le T4, et ça reste bloqué.

Donc pour pouvoir appliquer ces deux algorithmes d'ordonnancement librement, il faut trouver une solution réalisable qui nous permet d'éliminer les circuits à l'intérieur du graphe.

Il y'a eu une idée pour contourner ce problème proposée par « Benyamina Abou el hassen » dans sa thèse [57] qui se base sur le principe suivant : quand on rentre à l'intérieur d'un circuit on commence par ordonnancé en premier lieu la tâche qui à l'identité (N° de tâche) le plus petit. Mais pour cela il nous faut une technique ou algorithme qui nous permet de détecter à chaque tâche si elle est a l'intérieure d'un circuit.

Faute de temps on n'a pas pu traiter cette partie des circuits, et c'est pour cela qu'on va tester nos algorithmes d'ordonnancement sur des benchmarks qui ne contiennent pas de circuit dans leurs graphes d'application.

c. L'algorithme a évolution différentielle (DE) :

L'évolution différentielle (Differentiel evolution « DE ») est une méta-heuristique d'optimisation, inspirée des algorithmes génétiques et des stratégies évolutionnaires. Elle a été proposée par Rainer Storn en 1995. [58]

Cet algorithme se base essentiellement sur trois phases, les mêmes que ceux des algorithmes génétiques (sélection, croisement, mutation) mais dans le « DE » l'ordre des ces phases est comme suit : [59]

- Mutation.
- Croisement.
- Sélection.

Voici l'algorithme général du « DE » :

L'algorithme :

- Génération de la population initiale.
- Faire Pour chaque individu de la population initiale
  - Mutation
  - Croisement
  - Sélection
- Fin Pour

Le principe général de l'algorithme « DE » est le suivant, on génère la population initiale par un tirage aléatoire sur l'ensemble des valeurs possible pour chaque variable, les bornes inférieurs et supérieurs des variable sont spécifié par l'utilisateur selon la nature du problème.

Après la génération de la population initiale l'algorithme entame une phase ou il effectue une série de transformation sur les individus, dans un processus appelé « évolution ».

La population initiale contient N individus ou « N » représente la taille de la population, où chaque individu $x_{i,\,G}$ est représenté par un vecteur de dimension K, ou « G » désigne la génération.

L'individu de la population $x_{i,\,G} = (x_{1i,\,G}\ x_{2i,\,G}\ x_{3i,\,G}\ \ldots\ldots\ x_{Ki,\,G})$ où i=1…N

Comme on la dit un peu plus haut le « DE » se base sur trois technique (Mutation, Croisement, Sélection), où a chaque génération l'algorithme applique successivement ces trois techniques sur chaque individu de la population initiale, pour obtenir un vecteur qu'on appelle vecteur d'essaie $(u_{i,G+1})$:

$$u_{i,G+1} = (u_{1i,G+1},\ u_{2i,G+1},\ u_{3i,G+1}\ldots\ldots,\ u_{Ki,G+1}) \text{ où } i=1\ldots N$$

L'opération de sélection permet de choisir les individus à conserver pour la nouvelle génération (G+1).

- Détaille des techniques (mutation, croisement, sélection) :

✓ La mutation :

Dans la phase mutation de l'algorithme « DE » il existe plusieurs techniques de mutation qu'on va détailler un peu plus bas.

La phase mutation permet de créer à l'aide d'une des techniques de mutation, pour chaque vecteur courent $x_{i,\,G}$ un vecteur mutant $(v_{i,G+1})$.

Les techniques de mutation existante pour le « DE » sont les suivantes : [59]

❖ Rand/1 :

Dans cette première technique on choisit trois vecteurs aléatoires $(x_{r1,G},\ x_{r2,G},\ x_{r3,G})$ de notre population et on applique la formule suivante :

$$v_{i,G+1} = x_{r1,G} + F.(x_{r2,G} - x_{r3,G})$$

❖ Best/1 :

Dans cette technique on choisit le vecteur qui représente la meilleure solution pour notre problème (x_{best}), et deux autres vecteurs aléatoires ($x_{r1,G}$, $x_{r2,G}$) de la population et on applique la formule qui suit :

$$v_{i,G+1} = x_{best,G} + F.(x_{r1,G} - x_{r2,G})$$

❖ Current to best/1 :

Cette technique consiste a choisir deux vecteurs aléatoires de la population ($x_{r1,G}$, $x_{r2,G}$), le vecteur qui représente la meilleur solution (x_{best}), et le vecteur courent ($x_{i,G}$) et appliquer cette formule :

$$v_{i,G+1} = x_{i,G} + F.(x_{r1,G} - x_{r2,G}) + F.(x_{best} - x_{i,G})$$

❖ Best/2 :

Dans cette technique on choisit quatre vecteurs aléatoires de la population ($x_{r1,G}$, $x_{r2,G}$, $x_{r3,G}$, $x_{r4,G}$), et, le vecteur qui représente la meilleur solution (x_{best}) pour appliquer cette formule :

$$v_{i,G+1} = x_{best} + F.(x_{r1,G} - x_{r2,G}) + F.(x_{r3,G} - x_{r4,G})$$

❖ Rand/2 :

Dans cette dernière technique on choisit Cinque vecteurs aléatoires de la population initiale ($x_{r1,G}$, $x_{r2,G}$, $x_{r3,G}$, $x_{r4,G}$, $x_{r5,G}$) et on applique la formule suivante :

$$v_{i,G+1} = x_{r1,G} + F.(x_{r2,G} - x_{r3,G}) + F.(x_{r4,G} - x_{r5,G})$$

NB : « F » représente une valeur constante, appelé « differentiel weight » qui appartient à l'intervalle [0,2].

✓ Le croisement : [59]

Après la mutation et l'obtention du vecteur mutant ($v_{i,G+1}$) la technique de croisement est appliquer pour obtenir le vecteur d'essai final ($u_{i,G+1}$).

Ce vecteur d'essai et formé selon le vecteur $x_{i,G}$ et le vecteur mutant correspondant $v_{i,G+1}$ selon la formule suivante :

$$u_{ji,G+1} = \begin{cases} v_{i,G+1} \text{ si (randb (j) <= CR)} \\ x_{i,G} \text{ si (randb (j) > CR)} \end{cases} \quad \text{Pour tout j= 1…K}$$

NB : « randb(j) » est une valeur aléatoire appartenant à l'intervalle [0,1], et « CR » et un coefficient de croisement qui appartient à l'intervalle [0,1] aussi.

✓ La sélection : [59]

Est la dernière phase de l'algorithme, elle nous permet de décider quel vecteur parmi les deux vecteur ($u_{ji,G+1}$) ou ($x_{i,G}$) vas êtres choisit dans la nouvelles génération G+1.

Pour faire notre choix on va se basé sur la fonction du cout de ces deux vecteur, et celui qui a la fonction de cout minimale c'est celui qui vas être choisit dans la nouvelle génération.

L'expression de choix du nouveau vecteur ($x_{i,G+1}$) est la suivante :

$$x_{i,G+1} = \begin{cases} u_{ji,G+1} \text{ si } (f(u_{ji,G+1}) < f(x_{i,G})) \\ x_{i,G} \text{ si non} \end{cases}$$

C'est trois phases sont répétés pour chaque individus de la population initiale. Donc après avoir fini tous les individus, on choisi le vecteur ($x_{i,GM}$) qui a la fonction de cout minimale, et c'est ce vecteur qui vas représenter la solution au problème de mapping.

Voila en détaille les trois algorithmes utilisées pour la réalisation de notre travail.

3. Conclusion :

Pour la résolution de notre problème qui consiste à optimiser le temps dans les deux phases placement et ordonnancement de conception des NoC, on a implémenté une méthode de placement qui se base sur les algorithmes évolutionnaires qui est le « DE », et deux algorithmes d'ordonnancement le « ASAP » et « ALAP ».

En ce qui concerne les deux algorithmes d'ordonnancement, a moins de trouver une solution réalisable qui nous permet d'éliminer les circuits à l'intérieur d'un graphe de précédence l'application des deux algorithmes d'ordonnancement qu'on a utilisé As Soon As Possible et As Late As Possible, ne s'applique que sur des graphe de précédence sans circuit, et c'est pour cela qu'on a choisi pour tester notre solution d'utiliser les benchmarks qui sont le plus utilisés dans la littérature, ainsi que d'eux autres benchmarks qui ne sont pas souvent utilisé mais qui satisfait pour notre cas la contrainte de graphe sans circuit pour pouvoir tester la phase d'ordonnancement.

Dans le chapitre suivant on va définir les notions de graphe d'application et graphe d'architecture, qui sont des notions relié à notre travail, et présenté la manière dont la quelle nous avons adapté les algorithmes qu'on a utilisés pour résoudre notre problème, en plus des tests et résultats obtenus.

Chapitre VI : Mise en œuvre

Tests et résultats

1. Introduction

L'objectif de notre travaille, comme on la déjà préciser dans le chapitre précédent consiste à résoudre un problème de mapping et d'ordonnancement à la fois, tout en optimisant le cout de communication et le temps d'exécution. Donc pour résoudre ce problème on a utilisé en premier lieu deux technique d'ordonnancement, qui nous permettent de faire l'ordonnancement des tâches d'une application donnée, et en second lieu une technique de mapping qui permet d'affecter à chaque tuile de l'architecture du réseau sur puce une et une seule tâche du graphe d'application, tel que le coût de communication entres les tâches affecter a l'architecture soit minimiser, et par la suite selon le résultat de mapping obtenue on calcule le temps d'exécution de ces tâches.

Pour l'ordonnancement deux algorithmes sont implémentés qui sont ASAP (As Soon As Possible) et ALAP (As Late As Possible).

Tandis que pour la technique de mapping on a opté pour le DE (Differentiel evolution), c'est une méthode qui se base sur les algorithmes évolutionnaires, on a choisi cette méthode car elle n'a pas été beaucoup utilisée dans notre problème auparavant.

2. Définition de notre fonction objective :

On a proposé ces techniques d'ordonnancement et de placement dans le but de minimiser une fonction objective, cette dernière réside dans le fait de minimiser dans un premier temps dans la phase de mapping le cout de communication entre les différents IP de l'application, disposés sur l'architecture du réseau sur puce donnée, pour par la suite en second lieu utilisé ce résultat de mapping qui est considéré comme 'optimale' pour calculer le temps d'exécution des tâches, et l'ajouté au cout de communication, pour obtenir le cout de notre fonction objective.

$$f_{objective}(GA) = \alpha \text{ cout de communication} + \beta \text{ temps d'exécution}$$

GA : désigne le graphe d'application

Pour favoriser un critère de la fonction objective par rapport a l'autre, on peut les multiplier par des facteurs différents l'un de l'autre chacun selon le poids qu'on veut donner au critère. Mais dans notre cas on a considéré que les deux critères on le même poids, alors on les a multiplié tous les deux par un.

3. Définition du graphe d'application :

Le graphe d'application G (V, E) et un graphe orienté, où chaque sommet ($v_i \in V$) numéroté correspond à une IP de l'application, ces sommet sont reliés entre eux par des liens orienté ($e_{ij} \in E$) qui désigne la communication entre le sommet v_i et le sommet v_j. Chaque lien possède un poids w (e_{ij}) qui représente le nombre de paquets envoyé de v_i à v_j.

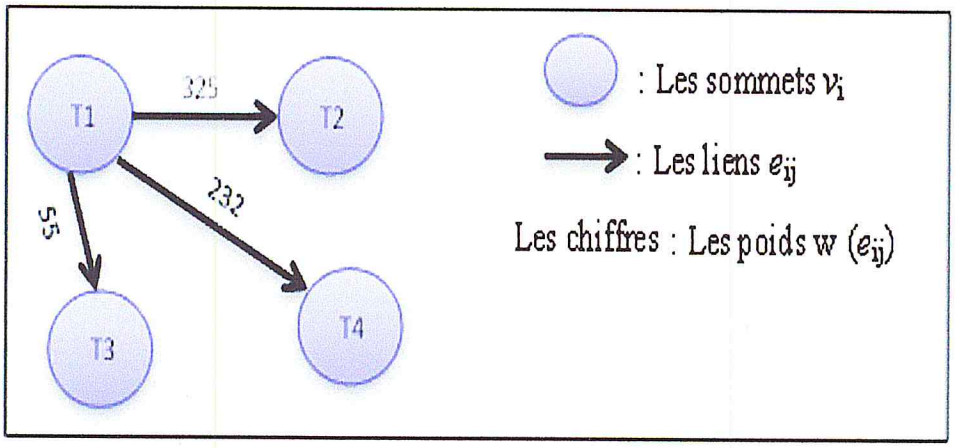

Figure29 : graphe d'application

4. Définition du graphe d'architecture

Le graphe d'architecture du NoC A (T, L) est un graphe orienté où chaque sommet t_i numéroté désigne une tuile de l'architecture du NoC, et chaque lien l_{ij} définit un lien de communication physique entre t_i et t_j.

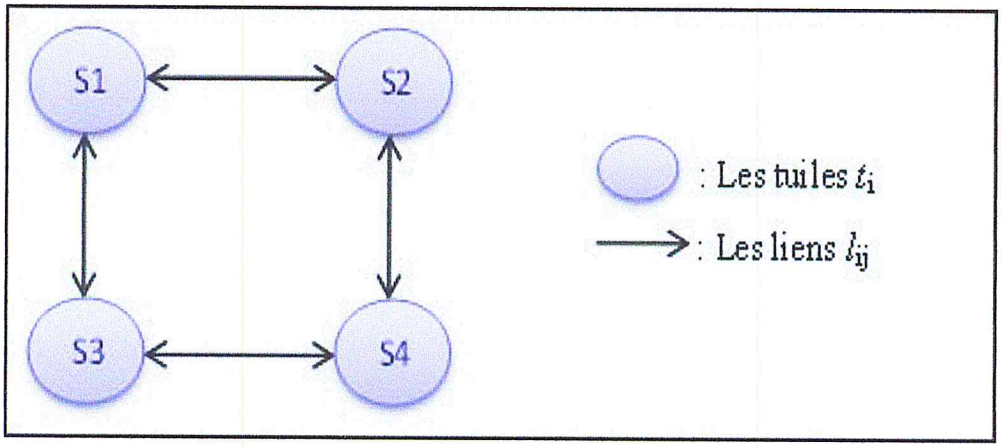

Figure30 : graphe d'architecture

5. Formulation du problème :

a. La phase d'ordonnancement :

La technique d'ordonnancement, va permettre de donner le séquencement d'exécution des IP de l'application (les v_i du graphe d'application). Selon l'algorithme d'ordonnancement choisit, on peut avoir dans un même cycle (C) plusieurs IP qui s'exécutent en même temps.

$$Ord\ (GA) = C_1 \dots\ C_n \dots\dots\dots\dots\dots\dots (1)$$

$$\text{Où } C_i = v_i \ ||\ v_i \dots\ v_p$$

$$\text{Où } i \text{ et } p \in \{1 \dots |V|\}, \text{ et } v_i\ ,\ v_p \in V$$

Et permet de calculer le temps d'exécution totale des IP.

$$temps\ d'exéc = \sum_{k=1}^{C} \big(Max(vi(k))\big) \dots\dots\dots\dots\dots\dots\dots\dots\dots (2)$$

Tel que $(vi(k))$ représente IP ordonnancé dans le même cycle (k).

b. La phase de mapping :

La technique de mapping (placement), va permettre d'affecter **chaque IP** de l'application (chaque v_i du graphe d'application) **à une et une seule tuile** t_i du graphe d'architecture.

$$\forall\ vi\ \in V, \exists\ tj\ \in T : map(vi) = tj \dots\dots\dots\dots\dots\dots\dots\dots (3)$$

Map() : représente la fonction de mapping.

Pour que le problème de mapping soit réalisable, il faut que le nombre de tuiles t_i soit **supérieur ou égal** au nombre d'IP de l'application.

$$|T| \geq |V| \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots (4)$$

Tout cela en minimisant le cout de communication totale des IP une fois posés sur le graphe d'architecture.

$$cout\ de\ comm = \sum_{k=1}^{E} \big((dk) * nombre\ de\ saut\big(source(dk), desti(dk)\big)\big) \dots\dots\dots (5)$$

Tel que (dk) représente w (e_{ij}) ; $source(dk) = map\ (v_i)$; $desti(dk) = map\ (v_j)$; $saut(a, b)$désigne le nombre de saut minimale entre la tuile a et la tuile b.

Le nombre de saut entre deux tuiles du réseau sur puce dans notre cas est calculé par la distance de Manhattan entre la tuile source et la tuile destination.

$$saut(a, b) = |x_a - x_b| + |y_a - y_b| \dots\dots\dots\dots\dots\dots\dots\dots (6)$$

NB : une solution de mapping peut être représenté par un tableau (t_{map}) de taille m, tel que m=|V| où l'indice du tableau représente une tuile (t_i) et le contenu de la case une IP (v_i).

6. Résolution du problème :

Pour pouvoir résoudre notre problème, une adaptation des structures nécessaires pour l'utilisation des algorithmes est obligatoire, c'est pour cela qu'on va présenter et définir en détaille toutes les structures utilisées.

6.1. Adaptation des algorithmes « ASAP » et « ALAP » pour le problème d'ordonnancement :

a. L'algorithme « ASAP » :

Pour utiliser l'algorithme « ASAP » on a besoin d'utiliser notre graphe d'application, donc nous l'avons représenté par une matrice carrée de taille (n*m) où n = m = nombre de sommet du graphe d'application les (v_i), et le contenu de la matrice représente le poids w (e_{ij}).

Les indices de lignes (i) et de colonnes (j) représentent les IP, et la case de la matrice qui correspond aux deux indices le poids w (e_{ij}).

$$\text{MatriceGraphe} = \begin{bmatrix} w(e_{ij}),\dots\dots\dots\dots\dots\dots w(e_{im}) \\ . \\ . \\ . \\ w(e_{nj}),\dots\dots\dots\dots\dots\dots w(e_{nm}) \end{bmatrix}$$

Voici à présente le corps de l'algorithme :

L'algorithme ASAP :

- Parcourir la matrice MatriceGraphe et chercher toutes les tâches qui n'ont pas de prédécesseurs.
- les ordonnancés dans un premier cycle selon (1)
- parcourir la matrice MatriceGraphe et tant qu'il reste des tâches à ordonnancer
  - ✓ chercher les tâches dont les prédécesseurs sont déjà ordonnancés
  - ✓ les ordonnancé dans un cycle suivant selon (1)
- jusqu'à ce que toutes les tâches soient ordonnancées

Dans notre solution on a choisi que le résultat de l'algorithme sera mis dans une structure de type matrice où (le nombre de ligne = au nombre de cycles) et (le nombre de colonne = au nombre de sommets de notre graphe d'application), qu'on va appeler (MatriceOrdo)

b. L'algorithme « ALAP »

Pour utiliser l'algorithme « ALAP » on a besoin aussi comme dans le « ASAP » de la matrice qui représente notre graphe d'application, c'est la même matrice « MatriceGraphe » que l'algorithme précédent et elle est définie de la même manière.

Pour le corps de l'algorithme « ALAP » le voici :

L'algorithme ALAP :

- Parcourir la matrice MatriceGraphe et chercher toutes les tâches qui n'ont pas de successeurs.
- les ordonnancés dans un premier cycle selon (1)
- parcourir la matrice MatriceGraphe et tant qu'il reste des tâches à ordonnancer
  - ✓ chercher les tâches dont les successeurs sont déjà ordonnancés
  - ✓ les ordonnancé dans un cycle suivant selon (1)
- jusqu'à ce que toutes les tâches soient ordonnancées

Pour la structure qui vas contenir le résultat de l'ordonnancement par l'algorithme « ALAP » c'est exactement la même que pour le « ASAP ».

6.2. Adaptation de l'algorithme DE pour le problème de mapping :

L'algorithme « DE » comme les deux algorithmes qui le précède, aura besoin lui aussi d'utiliser la matrice qui représente le graphe d'application. Cette matrice c'est la matrice

« MatriceGraphe » qui est définie exactement de la même façon que dans les deux algorithmes précédents.

Selon l'algorithme, le « DE » se base sur une population initiale, qui est générer aléatoirement parmi toutes les solutions possibles. Dans notre application la population initiale et considéré comme plusieurs tableaux de mapping (t_{map}) qui représentent des solutions de mapping réalisable.

Donc la population initiale peut être considérer comme une matrice où, le nombre de ligne est égale à la taille de la population (n), et le nombre de colonne est égale à la taille (m) des tableaux qui représente la solution du mapping (t_{map}).

$$PopInitiale = \begin{bmatrix} V_{ij}, \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots V_{im} \\ . \\ . \\ . \\ V_{nj}, \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots V_{nm} \end{bmatrix}$$

Selon l'algorithme « DE » pour chaque individu de la population initiale (chaque ligne de la matrice) on fait : mutation, croisement, sélection.

Pour les structures utilisées au niveau des phases : mutation, sélection, et croisement on a utilisé des tableaux de la même taille que les tableaux (t_{map}).

L'algorithme « DE » appliqué au problème de mapping et le suivant :

L'algorithme :

- Génération de la population initiale.
- Faire Pour chaque individu de la population initiale
  - Mutation
  - Croisement
  - Sélection
- Fin Pour

- Génération de la population initiale :

La population initiale du « DE » comme on la dit est générer aléatoirement.

Donc on fait :

Génération population initiale :

- Tant que indice < taille de la population
  - ✓ Générer aléatoirement un vecteur t_{map} selon (3)
  - ✓ Calculer son cout de communication selon (5)
  - ✓ L'insérer dans la matrice PopInitiale à la ligne indice
- fin de tant que

- Calcul du cout de communication :

Pour calculer le cout de communication d'un mapping on fait :

Calcule du cout de communication :

- Pour chaque IP du vecteur t_{map}
  - ✓ Repérer les autre IP avec qui il communique
    - Pour chaque IP repéré qui communique avec l'IP courent
      - On calcule le nombre de saut entre les deux dans l'architecture selon (6)
      - on actualise la valeur du cout de communication selon (5)
    - fin de pour
- fin pour

- La mutation :

Permet de créer un vecteur mutant (v_{Mut})

Pour la mutation nous avons implémenté Cinque technique de mutation qu'on va voir une par une :

✓ Rand/1 :

La technique Rand/1 :

- On choit à partir de la matrice PopInitiale trois lignes aléatoirement ce qui correspond a trois vecteur de type t_{map}
- On crée un vecteur qui va contenir le résultat de la soustraction entre le 2^{em} vecteur généré aléatoirement et le 3^{em}
- On crée un autre vecteur qui va contenir le résultat de la multiplication du vecteur de soustraction par une valeur constante F
- On remplie notre vecteur v_{Mut} par le résultat de l'addition entre le vecteur de multiplication et le 1^{er} vecteur choisi aléatoirement

✓ Best/1 :

La technique Best/1 :

- On choit à partir de la matrice PopInitiale deux lignes aléatoirement ce qui correspond a deux vecteur de type t_{map}
- On choisit aussi à partir de la matrice PopInitiale la ligne qui représente le cout de communication minimale ce qui correspond à un vecteur de type t_{map} qui représente la meilleure solution parmi la population initiale
- On crée un vecteur qui va contenir le résultat de la soustraction entre le 1^{er} vecteur généré aléatoirement et le 2^{em}
- On crée un autre vecteur qui va contenir le résultat de la multiplication du vecteur de soustraction par une valeur constante F
- On remplie notre vecteur v_{Mut} par le résultat de l'addition entre le vecteur de multiplication et le vecteur qui représente la meilleure solution

✓ Current to Best/1 :

La technique Current to Best/1 :

- On choit à partir de la matrice PopInitiale deux lignes aléatoirement ce qui correspond a deux vecteur de type t_{map}
- On choisit aussi à partir de la matrice PopInitiale la ligne qui représente le cout de communication minimale ce qui correspond à un vecteur de type t_{map} qui représente la meilleure solution parmi la population initiale
- On crée un vecteur qui va contenir le résultat de la soustraction entre le vecteur qui représente la meilleure solution et le vecteur courant dans la population initiale
- On crée un autre vecteur qui va contenir le résultat de la 1ere multiplication de ce 1er vecteur de soustraction par une valeur constante F
- On crée un vecteur qui va contenir le résultat d'une 2em soustraction entre le 1er vecteur choisi aléatoirement et le 2em
- On crée un autre vecteur qui va contenir le résultat d'une 2em multiplication de ce 2em vecteur de soustraction par la même valeur constante F
- On remplie notre vecteur mutant v_{Mut} par le résultat de l'addition entre le vecteur courant dans la population initiale, le 2em vecteur de multiplication et le 1er vecteur de multiplication

✓ Best/2 :

La technique Best/2 :

- On choit à partir de la matrice PopInitiale quatre lignes aléatoirement ce qui correspond a quatre vecteur de type t_{map}
- On choisit aussi à partir de la matrice PopInitiale la ligne qui représente le cout de communication minimale ce qui correspond à un vecteur de type t_{map} qui représente la meilleure solution parmi la population initiale
- On crée un vecteur qui va contenir le résultat de la soustraction entre le 4em vecteur choisit aléatoirement et le 3em
- On crée un autre vecteur qui va contenir le résultat de la 1ere multiplication de ce 1er vecteur de soustraction par une valeur constante F
- On crée un vecteur qui va contenir le résultat d'une 2em soustraction entre le 1er vecteur choisi aléatoirement et le 2em
- On crée un autre vecteur qui va contenir le résultat d'une 2em multiplication de ce 2em vecteur de soustraction par la même valeur constante F
- On remplie notre vecteur mutant v_{Mut} par le résultat de l'addition entre le vecteur qui représente la meilleure solution, le 2em vecteur de multiplication et le 1er vecteur de multiplication

✓ Rand/2 :

La technique Rand/2 :

- On choit à partir de la matrice PopInitiale Cinque lignes aléatoirement ce qui correspond a Cinque vecteur de type t_{map}
- On crée un vecteur qui va contenir le résultat de la soustraction entre le 5^{em} vecteur choisit aléatoirement et le 4^{em}
- On crée un autre vecteur qui va contenir le résultat de la 1^{ere} multiplication de ce 1^{er} vecteur de soustraction par une valeur constante F
- On crée un vecteur qui va contenir le résultat d'une 2^{em} soustraction entre le 3^{em} vecteur choisi aléatoirement et le 2^{em}
- On crée un autre vecteur qui va contenir le résultat d'une 2^{em} multiplication de ce 2^{em} vecteur de soustraction par la même valeur constante F
- On remplie notre vecteur mutant v_{Mut} par le résultat de l'addition entre le 1^{er} vecteur choisit aléatoirement, le 2^{em} vecteur de multiplication et le 1^{er} vecteur de multiplication

Voila les Cinque technique de mutation qu'on a utilisée.

- Le croisement :

Cette phase nous permet d'obtenir un vecteur qu'on appelle vecteur d'essaie finale v_{Croi}.

Le croisement :

- Tant j < taille de v_{Croi}
  - ✓ v_{Croi} (j) reçois v_{Mut} (j) si le j^{em} nombre généré aléatoirement depuis un intervalle constant <= a une constante CR
  - ✓ v_{Croi} (j) reçois vecteur courant dans la population initiale (j) si le j^{em} nombre généré aléatoirement depuis un intervalle constant > a une constante CR
- fin de tant que

- La sélection :

Cette étape va nous permettre de décider si on va garder le vecteur courant de la population initiale de la génération courante dans la génération suivante ou bien de le remplacer par le vecteur v_{Croi}. Et cela dépends du cout de communication de chaque vecteur.

La sélection :

- Le vecteur à choisir dans la génération suivante est :
  - ✓ v_{Croi} si la fonction objective (v_{Croi}) < la fonction objective (vecteur courant de la population initiale de la génération courante)
  - ✓ vecteur courant de la population initiale de la génération courante si non

Après avoir obtenue le résultat de l'ordonnancement et celui du mapping, on va maintenant pouvoir calculer le temps d'exécution de l'application représenté par notre graphe d'application selon le résultat du mapping. Et pour cela on a besoin d'une structure qui va contenir les temps d'exécution de chaque IP sur chaque tuile du graphe d'architecture donnée, puisque l'architecture de notre réseau sur puce est considérée comme hétérogène par rapport au temps d'exécution des tâches.

Pour représenter ceci on a opté dans notre solution a une structure matricielle de taille (n*m) où n = nombre de lignes = les sommets de notre graphe d'application (les IP ou Tâche) et m = nombre de colonnes = les tuiles de notre graphe d'architecture, le contenu de la matrice c'est des valeurs qui représente le temps d'exécution de chaque tâche sur la tuile correspondante.

Les indices de lignes (i) représentent les IP et de colonnes (j) représentent les tuiles, et la case de la matrice qui correspond aux deux indices représente le temps d'exécution de la tâche i sur la tuile j.

$$MatriceTempsExec = \begin{bmatrix} T_{ij}, \ldots\ldots\ldots\ldots T_{im} \\ . \\ . \\ . \\ T_{nj}, \ldots\ldots\ldots\ldots T_{nm} \end{bmatrix}$$

Donc pour calculer le temps d'exécution des tâches on a besoin de cette matrice (MatriceTempsExec), de la matrice qui contient le résultat de notre ordonnancement

(MatriceOrdo) et d'un vecteur de type (t_{map}) qui représente un résultat de mapping, ce vecteur va nous préciser pour chaque tâche (IP) sur quelle tuile elle est mise (mappé).

Le calcule du temps d'exécution :

- Parcourir la matrice MatriceOrdo et tant que indice < nombre de ligne de la matrice
  - ✓ Chercher le temps d'exécution des tâches ordonnancées dans le cycle qui correspond a indice à partir du vectur t_{map} et de la matrice MatriceTempsExec
  - ✓ Mettre à jour la valeur du temps d'exécution selon (2)
- Fin de tant que

Et à la fin pour obtenir le cout total de notre fonction objective qui, rappelons est définie comme suite :

$$f_{objective}(GA) = \text{cout de communication} + \text{temps d'exécution}$$

GA : désigne le graphe d'application

On fait l'addition entre le cout de communication de notre solution de mapping obtenue par l'algorithme « DE » et le temps d'exécution obtenue.

7. Implémentation de notre application :

Pour l'implémentation de notre application on implémenté tous les algorithmes et fonction qu'on a cité plus haut dans ce chapitres.

Dans cette partie du chapitre on va voir chaque fonction avec ses paramètre d'entrées et sorties.

a. La technique d'ordonnancement :

- ✓ Son paramètre de sortie est une matrice
- ✓ Ses paramètres d'entrées sont :
  - La matrice qui représente notre graphe d'application
  - Le nom de l'algorithme d'ordonnancement a appliqué

Cette technique fait appel à d'autre méthode selon l'algorithme d'ordonnancement qu'on veut utiliser, qui sont :

- La méthode qui applique l'algorithme « ASAP »
  - ✓ Son paramètre de sortie est une matrice
  - ✓ Son paramètre d'entrée est la matrice qui représente le graphe d'application
- La méthode qui applique l'algorithme « ALAP »
  - ✓ Son paramètre de sortie est une matrice
  - ✓ Son paramètre d'entrée est la matrice qui représente le graphe d'application

b. <u>La technique de mapping (L'algorithme « DE ») :</u>

- ✓ Son paramètre de sortie est une matrice.
- ✓ Ses paramètres d'entrées sont :
  - La matrice qui représente notre graphe d'application
  - Le nombre de ligne du réseau sur puce
  - Le nombre de colonne du réseau sur puce
  - La taille de la population
  - La constante F
  - La constante CR
  - Le nom de la technique de mutation qu'on va utiliser

Cet algorithme fait appel des méthodes qui sont :

- Une méthode qui fait l'initialisation du « DE ». Elle crée la population initiale
  - ✓ Son paramètre de sortie est une matrice.
  - ✓ Ses paramètres d'entrées sont :
    - La matrice qui représente notre graphe d'application
    - La taille de la population
    - La taille totale de notre réseau sur puce (nombre de lignes*nombre de colonnes)
- Une méthode des méthodes qui correspond a la technique de mutation qu'on va appliquer. Les Cinque méthodes implémentés ont les mêmes paramètres d'entrées sorties, sauf la « current to best/1 » elle a un paramètre d'entée en plus
  - ✓ Leurs paramètre de sortie est un vecteur
  - ✓ Leurs paramètres d'entrées sont :
    - La matrice qui représente la population initiale
    - Une constante F
    - Pour la technique « current to best/1 » on à ce paramètre en plus qui est l'indice de la ligne qui représente le vecteur courant.

- Une méthode qui fait le croisement.
  - ✓ Son paramètre de sortie est un vecteur
  - ✓ Ses paramètres d'entrées sont :
    - Le vecteur mutant
    - Le vecteur courant
    - Une constante CR
- Une méthode qui fait la régulation de vecteur.
  - ✓ Sont paramètre de sortie est un vecteur
  - ✓ Ses paramètres d'entrées sont :
    - Un vecteur qui représente un vecteur régularisé
    - Un vecteur qui représente un vecteur non régularisé
- Une méthode qui calcule le cout de communication.
  - ✓ Son paramètre de sortie est un entier
  - ✓ Ses paramètres d'entrées sont :
    - Le vecteur qui représente une solution de mapping réalisable dont on veut calculer le cout
    - La matrice qui représente notre graphe d'application
    - Le nombre de colonne de notre architecture de réseau sur puce
- Une méthode qui fait la sélection
  - ✓ Son paramètre de sortie est un vecteur
  - ✓ Ses paramètres d'entrées sont :
    - Un vecteur qui représente le vecteur d'essai final
    - Le vecteur courant
- Une méthode qui nous donne le résultat du mapping
  - ✓ Son paramètre de sortie est un vecteur
  - ✓ Son paramètre d'entrée est la dernière génération de la matrice de la population initiale

Ensuite après avoir obtenu le résultat du mapping et celui de l'ordonnancement on fait appel à :

- Une méthode qui nous calcule le temps d'exécution
  - ✓ Son paramètre de sortie est une valeur
  - ✓ Ses paramètres d'entrées sont :

- Une matrice qui représente le résultat de l'ordonnancement
- Une matrice qui représente les temps d'exécution de chaque IP du graphe d'application sur une tuile de l'architecture du réseau sur puce
- Un vecteur qui représente le résultat de mapping

c. Une fonction qui calcule le cout totale de notre fonction objective :

- ✓ Son paramètre de sortie est une valeur
- ✓ Ses paramètres d'entrées sont :
  - Un vecteur qui représente le résultat du mapping
  - Une valeur qui représente la valeur du temps d'exécution

Après avoir implémenté les algorithmes, on va faire une série de tests selon plusieurs critères pour vérifier la performance des techniques de placement et d'ordonnancement qu'on a choisi.

8. Tests et résultats :

Ces tests sont faits sur des benchmarks existant et qui ont été déjà utiliser dans la littérature, pour pouvoir comparer nos résultats avec les résultats déjà obtenue.

a. Présentation des benchmark utilisés :

Dans cette partie on va présenter les différents benchmark utilisés pour la réalisation de ces tests.

- Le benchmark (MPEG4) :

Le MPEG4 est une norme de codage d'objets audio visuels, elle a été spécifiée par le « Moving Picture Experts Group » [site8].

Le benchmark MPEG4, est composé de 12 nœuds (IP) reliés entre eux par 26 liens bidirectionnels comme le montre la figure :

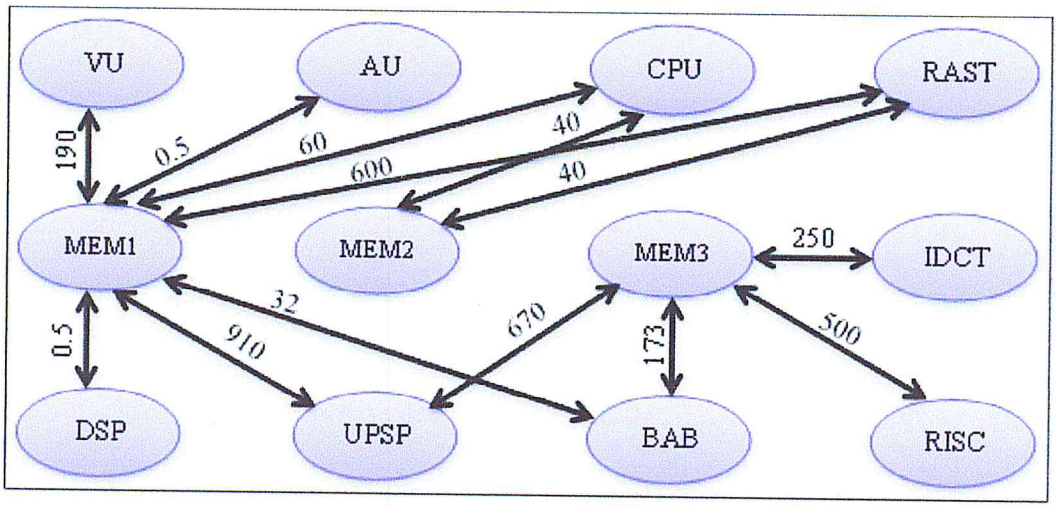

Figure31 : le graphe d'application du benchmark MPEG4 [22]

- Le benchmark (MWD) :

L'application MWD abréviation du nom complet « Multi-Window Display » [60]. C'est une application qui permet comme son nom l'indique d'afficher plusieurs fenêtres.

Elle est composée de 12 IP qui communiquent entre eux à travers 13 liens, comme le montre le graphe suivante :

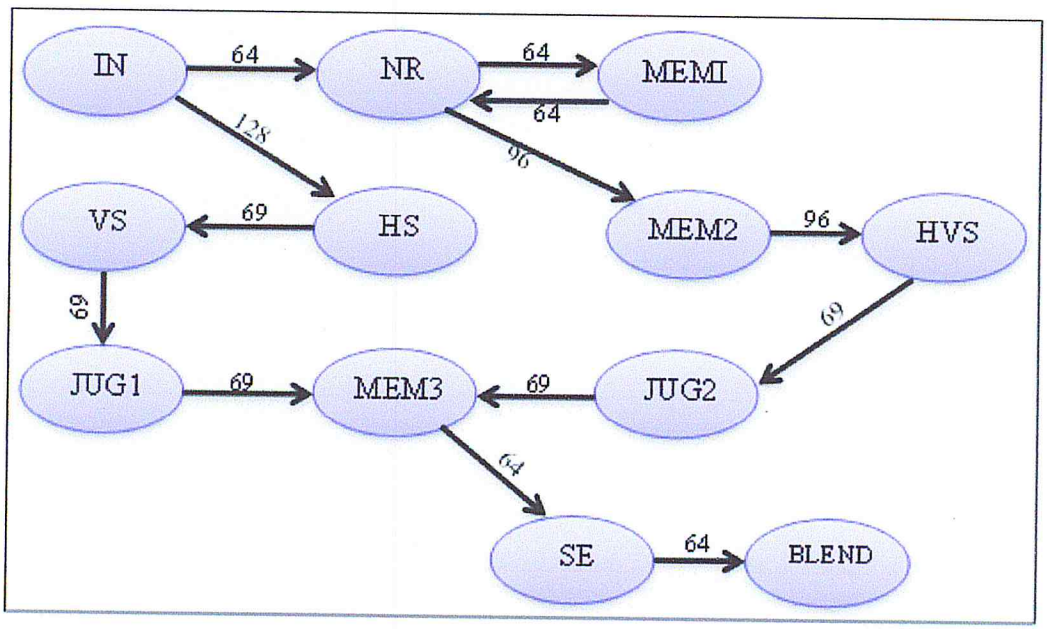

Figure32 : le graphe d'application du benchmark MWD [61]

- Le benchmark (VOPD) :

L'application VOPD du nom complet « Video Object Plane Decoder » [60]. C'est une application de décodage vidéo, qui est constituée de 16 IP communicant à travers 21 liens, comme dans la figure suivante :

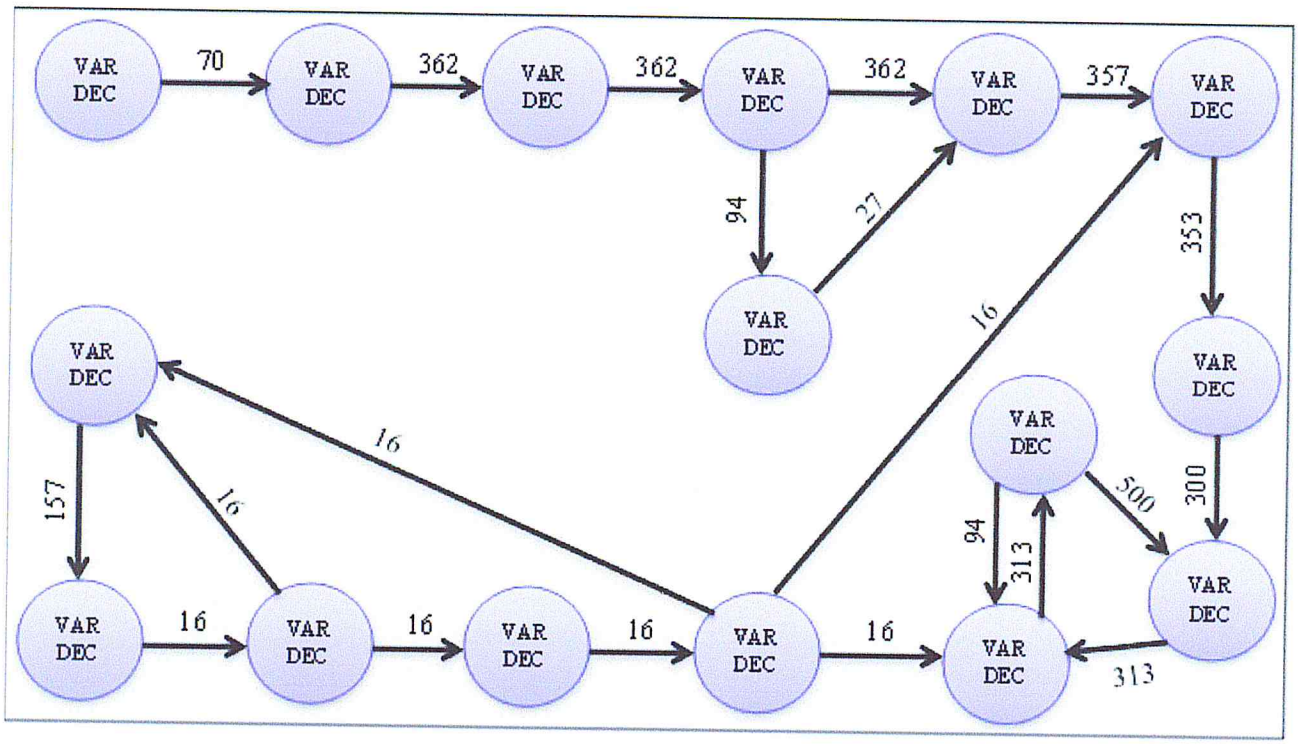

Figure33 : le graphe d'application du benchmark VOPD [61]

C'est trois benchmarks (MPEG4, MWD, VOPD) sont utilisés pour tester la technique de placement qu'on a choisit et implémenter (l'algorithme « DE »), mais par contre on ne peut pas les utilisés pour tester le placement et l'ordonnancement a la fois, parce que c'est des benchmarks qui contiennent des circuits dans leurs graphes d'application, et comme on avait dit dans le chapitre précédent les deux techniques d'ordonnancement qu'on a choisit (L'algorithme « ASAP » et « ALAP ») ne s'appliquent que sur des graphes d'application sans circuits, a moins d'implémenter une technique qui permet de détourner ce problème. C'est pour cela qu'on a choisit deux autres benchmarks qu'on va présenter par la suite pour tester le placement et ordonnancement a la fois.

C'est deux benchmark sont :

- Le benchmark (MJPEG) :

Le MJPEG ou Motion JPEG est un codec vidéo, qui permet de compresser des images JPEG une a une. Un codec vidéo (COmpresseurs - DECompresseurs) [site9] c'est un dispositif capable de compresser et/ou décompresser un signale numérique.

Ce benchmark est constitué de 5 IP qui communiquent entre eux à travers 7 liens. Le graphe d'application de ce benchmark est représenté par la figure suivante :

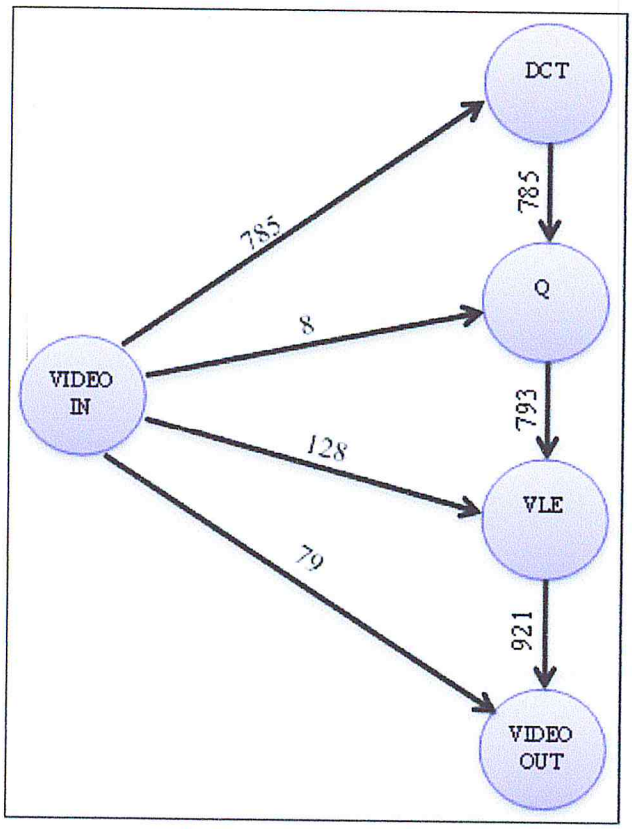

Figure34 : le graphe d'application du benchmark MJPEG [61]

- Le benchmark (PIP) :

PIP acronyme du mot « picture in picture » est une fonction qui permet de regarder une deuxième chaine sur l'écran d'une télévision. Une première chaine est affichée en plein écran alors que la deuxième est affichée dans une petite fenêtre [22].

Ce benchmark est composé de 8 IP reliés entre eux par 8 liens. La figure suivante décrit son graphe d'application.

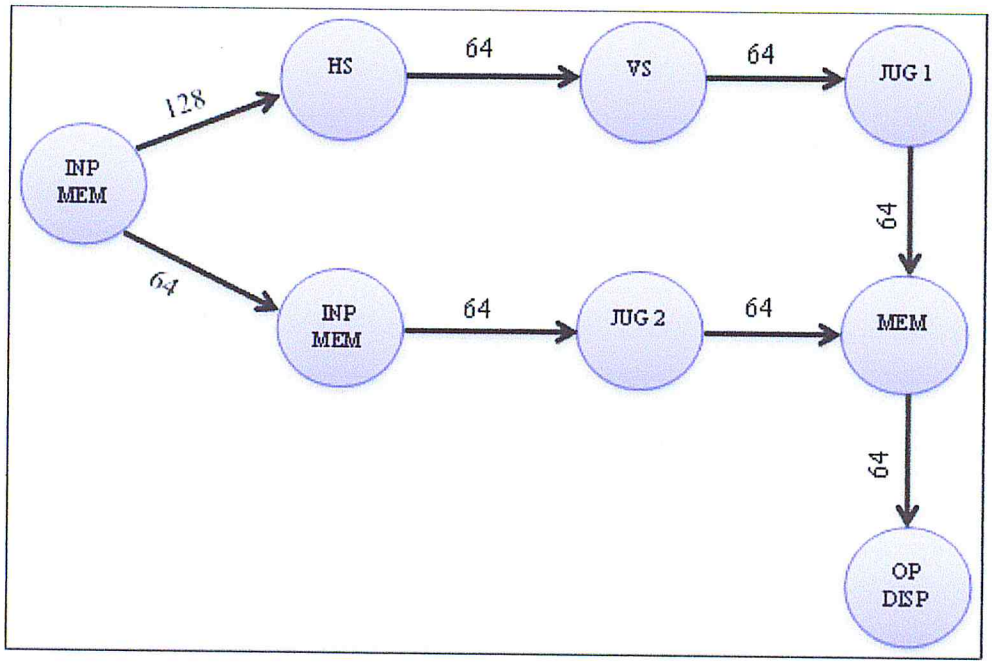

Figure35 : le graphe d'application du benchmark PIP [61]

C'est deux derniers benchmarks (PIP et MJPEG), comme on peut le voir dans les figures qui représentent les graphes d'applications de chacun d'entre eux, ils ne contiennent pas de circuit donc on pourra sans aucun problème tester les méthodes de placement et d'ordonnancement qu'on a implémenté sur ces benchmarks.

b. Réalisation des tests :

En ce qui concerne la réalisation des tests, on a réalisé ça en deux phases. La première phase consiste à tester la performance de notre méthode de placement (l'optimisation du cout de communication) a part, et pour cela on a utilisé tous les benchmarks qu'on a définie un peut plus haut.

La deuxième phase consiste à tester les méthodes des deux phases de placement et d'ordonnancement a la fois, et pour cela on à utilisé que les deux derniers benchmark (PIP et MJPEG) parce qu'ils ne contiennent pas de circuit, et comme on l'a déjà précisé la méthode d'ordonnancement qu'on a choisi ne s'applique que sur les benchmark sans circuit.

Pour la comparaison de nos résultats et l'évaluation des méthodes qu'on a utilisées, on a comparé nos résultats obtenus avec des résultats qui existent dans la littérature.

✓ Phase une des tests : (cout de communication)

Pour faire ces tests on se base sur Cinque critères essentiels dans la méthode de mapping qu'on choisit « DE » et qui sont :

- La taille de la population.
- Le facteur de mutation F.
- Le facteur de croisement CR.
- Et le type de mutation.
- Taille de l'architecture du NoC.

On va maintenant présenter des tableaux qui contiennent les résultats qu'on a obtenus tout en variant dans les critères cité ci-dessus.

❖ En variant dans la taille de la population :

Pour réaliser ce test on a fixé : -le facteur de mutation F=0.0001

-le facteur de croisement CR=0.85

-le type de mutation à « best1 »

-la taille du NoC choisit est 4*4

Et voila ce qu'on a obtenu :

En ce qui concerne les deux benchmarks (MJPEG et PIP), vu leurs simplicité on arrive à atteindre la solution optimale au niveau cout de communication au bout de quelques itérations.

Benchmark	Taille de la population	Résultats obtenus	Résultat optimal
MJPEG	10	3842	inconnu
	100	3586	
	1000	3586	
PIP	10	768	inconnu
	100	640	
	1000	640	

Tableau 3 : résultats selon la taille de la population pour MJPEG et PIP

Pour les trois autres benchmarks (MPEG4, MWD et VOPD) ils sont beaucoup plus complexe voici les résultats obtenus :

Benchmark	Taille de la population	Résultats obtenus	Résultat optimal
MPEG4	100	3740	3567 [62 - 63]
	1000	3563	
	10000	3470	
	20000	3470	
MWD	100	1408	1184 [22]
	1000	1216	
	10000	1184	
	20000	1184	
VOPD	100	4693	4119 [62 - 63]
	1000	4157	
	10000	4125	
	20000	4119	

Tableau 4 : de résultats selon la taille de la population pour MPEG4, MWD et VOPD

C'est résultats nous permettent de déduire que tant que la taille de la population est grande, tant qu'on tend beaucoup plus vers l'optimum.

❖ <u>En variant dans le facteur de mutation F :</u>

En ce qui concerne les deux premiers benchmarks MJPEG et PIP on va les utilisés dans ces tests parce que comme on vu dans les tests précédents ils sont simple et convergent rapidement. Donc les tests vont être faits sur le MPEG4, MWD et VOPD.

Pour réaliser ce test on a fixé : -la taille de la population 20000

-le facteur de croisement CR=0.85

-le type de mutation à « best1 »

-la taille du NoC choisit est 4*4

Benchmarks	Facteur de mutation	Résultats obtenus	Résultat optimal
MPEG4	0.2	3470	3567 [62 - 63]
	0.3	3470	

	0.4	3470	
	0.5	3470	
	0.6	3470	
	0.7	3470	
	0.8	3470	
	0.9	3550	
	1	3550	
	1.1	3595	
	1.2	3660	
	1.3	3660	
	1.4	3660	
	1.4	3660	
	1.5	3360	
	1.3	3660	
	1.6	3643	
	1.7	3726	
	1.8	3726	
	1.9	3740	
	2	3740	
MWD	0.0001	1184	1184 [22]
	0.1	1184	
	0.2	1184	
	0.3	1184	
	0.4	1184	
	0.5	1216	
	0.6	1280	
	0.7	1280	
	0.8	1216	
	0.9	1248	
	1	1312	
	1.1	1280	
	1.2	1344	
	1.3	1344	

	1.4	1344	
	1.5	1344	
	1.6	1344	
	1.7	1376	
	1.8	1408	
	1.9	1440	
	2	1408	
VOPD	0.0001	4119	4119 [62 - 63]
	0.1	4125	
	0.2	4135	
	0.3	4301	
	0.4	4466	
	0.5	4966	
	0.6	4984	
	0.7	4998	
	0.8	4884	
	0.9	4924	
	1	4871	
	1.1	5048	
	1.2	4818	
	1.3	4641	
	1.4	4926	
	1.5	5028	
	1.6	4927	
	1.7	5084	
	1.8	4871	
	1.9	4891	
	2	4960	

Tableau 5 : des résultats selon le facteur de mutation F

A travers c'est résultats on peut constater que tant que le facteur de mutation F tend vers le 0 tant qu'on obtient des résultats beaucoup plus proche de l'optimum.

❖ <u>En variant dans le facteur de croisement CR :</u>

Pour réaliser ce test on a fixé : -la taille de la population 20000

-le facteur de croisement F=0.0001

-le type de mutation à « best1 »

-la taille du NoC choisit est 4*4

On a obtenu les résultats suivants :

Benchmarks	Facteur de croisement	Résultats obtenus	Résultat optimal
MPEG4	0.1	3666	3567 [62 - 63]
	0.15	3666	
	0.2	3660	
	0.25	3470	
	0.3	3470	
	0.35	3470	
	0.4	3470	
	0.45	3470	
	0.5	3470	
	0.55	3470	
	0.6	3470	
	0.65	3470	
	0.7	3470	
	0.75	3470	
	0.8	3470	
	0.85	3470	
	0.9	3470	
	0.95	3470	
	1	3563	
MWD	0.1	1216	1184 [22]
	0.15	1216	

	0.2	1216	
	0.25	1280	
	0.3	1184	
	0.35	1184	
	0.4	1184	
	0.45	1184	
	0.5	1184	
	0.55	1184	
	0.6	1184	
	0.65	1184	
	0.7	1184	
	0.75	1184	
	0.8	1184	
	0.85	1184	
	0.9	1184	
	0.95	1184	
	1	1376	
VOPD	0.1	4735	4119
	0.15	4441	[62 - 63]
	0.2	4620	
	0.25	4484	
	0.3	4431	
	0.35	4263	
	0.4	4231	
	0.45	4189	
	0.5	4167	
	0.55	4135	
	0.6	4135	
	0.65	4125	
	0.7	4125	
	0.75	4125	
	0.8	4125	
	0.85	4119	

	0.9	4119	
	0.95	4119	
	1	4625	

Tableau 6 : résultats selon le facteur de croisement CR

D'après ces résultats on remarque que le cout de communication est optimal, tant que le CR tant vers 1 mais n'égale pas 1.

❖ <u>En variant dans le type de mutation :</u>

Pour réaliser ce test on a fixé : -la taille de la population 20000

-le facteur de croisement F=0.0001

-le facteur de croisement CR=0.85

-la taille du NoC choisit est 4*4

Benchmarks	Type de mutation	Résultats obtenus	Résultat optimal
MPEG4	Rand/1	3660	3567 [62 - 63]
	Rand/2	3666	
	Current to best/1	3667	
	Best/1	3470	
	Best/2	3470	
MWD	Rand/1	1280	1184 [22]
	Rand/2	1216	
	Current to best/1	1248	
	Best/1	1184	
	Best/2	1184	
VOPD	Rand/1	4701	4119 [62 - 63]
	Rand/2	4743	
	Current to best/1	4965	
	Best/1	4119	
	Best/2	4119	

Tableau 7 : résultats selon le type de mutation

D'après ces résultats c'est clair que les deux types de mutations qui nous permettent d'atteindre l'optimum sont sans aucun doute le « Best/1 » et le « Best/2 ».

❖ <u>En variant dans la taille de l'architecture du NoC :</u>

Pour réaliser ce test on a fixé : -la taille de la population 20000

-le facteur de croisement F=0.0001

-le facteur de croisement CR=0.85

-le type de mutation « Best/2 »

Benchmark	Taille du NoC	Résultats obtenus	Résulta optimal
MJPEG	3*2	3586	Inconnu
	3*3	3586	
	3*4	3842	
	4*4	5412	
PIP	3*3	640	Inconnu
	3*4	640	
	3*5	768	
	4*4	768	
MPEG4	3*4	3470	3567 [62 - 63]
	4*4	3470	
	4*5	3630	
	5*5	4329	
MWD	3*4	1184	1184 [22]
	4*4	1216	
	4*5	1504	
	5*5	1728	
VOPD	4*4	4119	4119 [62 - 63]
	4*6	4447	
	5*5	4721	

Tableau 8 : résultats selon la taille de la topologie

C'est résultats nous permettent de constater que tant que la taille de l'architecture du NoC est proche du nombre d'IP du graphe d'application on obtient de meilleurs résultats que si la taille de l'architecture du NoC été plus grande que le nombre d'IP.

D'après les résultats de tous les tests fait, on peut dire pour obtenir un résultat optimal ou très proche de l'optimum, les valeurs qu'on doit affecter au paramètres du DE sont :

- Taille de la population : la plus grande possible selon la complexité du graphe d'application.
- Le facteur de mutation F : le plus proche que possible du 0.
- Le facteur de croisement CR : éloigné du 0 en atteignant pas le 1
- Le type de mutation : soit le « Best/1 » ou le « Best/2 »
- La taille de l'architecture NoC : la plus proche du nombre d'IP de l'application que possible.

Maintenant on va entamer la deuxième phase des tests qui consiste à tester la méthode de placement et d'ordonnancement a la fois.

✓ Phase deux des tests : (temps d'exécution)

Dans cette partie des tests on va comparer les temps d'exécution, et comme on l'a dit auparavant pour calculer le temps d'exécution il faut faire l'ordonnancement, donc ces tests se basent sur la technique d'ordonnancement choisit : -l'algorithme ASAP

-l'algorithme ALAP

Mais pas seulement de ça, le calcul du temps d'exécution dépend aussi du résultat du placement, parce qu'on a considéré le problème de placement et d'ordonnancement comme étant un seule problème complexe, et aussi de la matrice des temps d'exécutions.

NB : pour la réalisation de ces tests on utilise les deux benchmarks qui ne contiennent pas de circuit dans leurs graphes d'application qui sont le MJPEG et PIP. Parce que comme on l'a précisé dans le chapitre précédent ces deux algorithmes ne s'appliquent que sur des graphes sans circuit.

❖ Les résultats des temps d'exécution selon le cout de communication par l'algorithme ASAP :

Benchmarks	Cout de communication	Temps d'exécution
MJPEG	3842	45
	3586	46
	4428	18
	3842	49
	3586	30
PIP	768	32
	896	59
	768	26
	640	37
	704	13

Tableau 9 : influence du cout de communication sur le temps d'exécution, résultat selon l'algorithme ASAP

Comme on voit sur le tableau, pour le benchmark MJPEG on a obtenu un temps d'exécution égale à 18 (ms) par le biais d'un résultat de mapping dont le cout de communication est égale à 4428, malgré que le cout de communication optimal pour le MJPEG d'après nos résultats est de 3586. Et les temps d'exécution obtenu par le biais du cout de communication optimale sont égaux à 46 et 30 (ms).

Pareil pour le benchmark PIP, le minimum coté temps d'exécution qui est de 13 (ms) on l'a obtenu par le biais d'un cout de communication égale à 704. Malgré que l'optimum coté cout de communication obtenu pour ce benchmark est de 604, qui nous a donné un temps d'exécution égale à 37 (ms).

❖ Les résultats des temps d'exécution selon le cout de communication par l'algorithme ALAP :

Benchmarks	Cout de communication	Temps d'exécution
MJPEG	4570	18
	3842	28
	3586	33

	3842	14
	4842	12
PIP	832	45
	640	62
	896	17
	768	38
	896	25

Tableau 10 : influence du cout de communication sur le temps d'exécution, résultat selon l'algorithme ALAP

Comme on voit sur ce tableau aussi, pour le benchmark MJPEG on a obtenu un temps d'exécution égale à 12 (ms) par un résultat de mapping dont le cout de communication est égale à 4842, malgré que le cout de communication optimal comme on l'a dit avant pour le MJPEG d'après nos résultats est de 3586. Et le temps d'exécution obtenu par le biais du cout de communication optimale est égal à 33 (ms).

Même chose pour le benchmark PIP, le minimum coté temps d'exécution qui est de 17 (ms) on l'a obtenu par le biais d'un cout de communication égale à 896, et on remarque qu'il y a un autre cout de communication qui est égal à 896 mais qui donne un temps d'exécution égal à 25. Pour l'optimum coté cout de communication obtenu pour ce benchmark est de 604, et il nous a donné un temps d'exécution égale à 37 (ms).

D'après ces résultats obtenus on remarque que, le cout de communication n'influence pas sur le temps d'exécution, mais c'est le résultat du placement des tâches sur la topologie du NoC (quelle tâche sur quelle tuile est placée ?) qui influence sur la valeur du temps d'exécution.

- Comparaison entre les temps d'exécution selon ASAP et ALAP :

Ici on essaye de comparer entres des temps d'exécution obtenue par les deux algorithmes selon des solutions de mapping. Ces dernières les mêmes pour les deux algorithmes.

On a réalisé ce test sur le benchmark MJPEG et voila les résultats :

Solutions	Temps d'exécution selon ASAP	Temps d'exécution selon ALAP
Sol 1	13	15
Sol 2	9	9
Sol 3	13	17

Sol 4	17	16
Sol 5	16	15
Sol 6	17	14
Sol 7	18	18
Sol 8	15	16
Sol 9	14	16
Sol 10	16	16

Tableau 11 : comparaison du temps d'exécution selon l'algorithme d'ordonnancement

D'après ce tableau on peut remarquer que dans 10 solutions de mapping, le « ASAP » et meilleur dans 4 d'entre elles, le « ALAP » dans 3, et pour les 3 autres qui reste les deux algorithmes donne le même résultat. Alors on peut conclure que le « ASAP » et légèrement favorable par rapport au « ALAP »

c. Comparaison entre les résultats obtenus :

✓ Pour le cout de communication :

Après avoir effectué les tests selon différents critères, on va maintenant comparer les résultats obtenus.

- Selon la taille de la population :

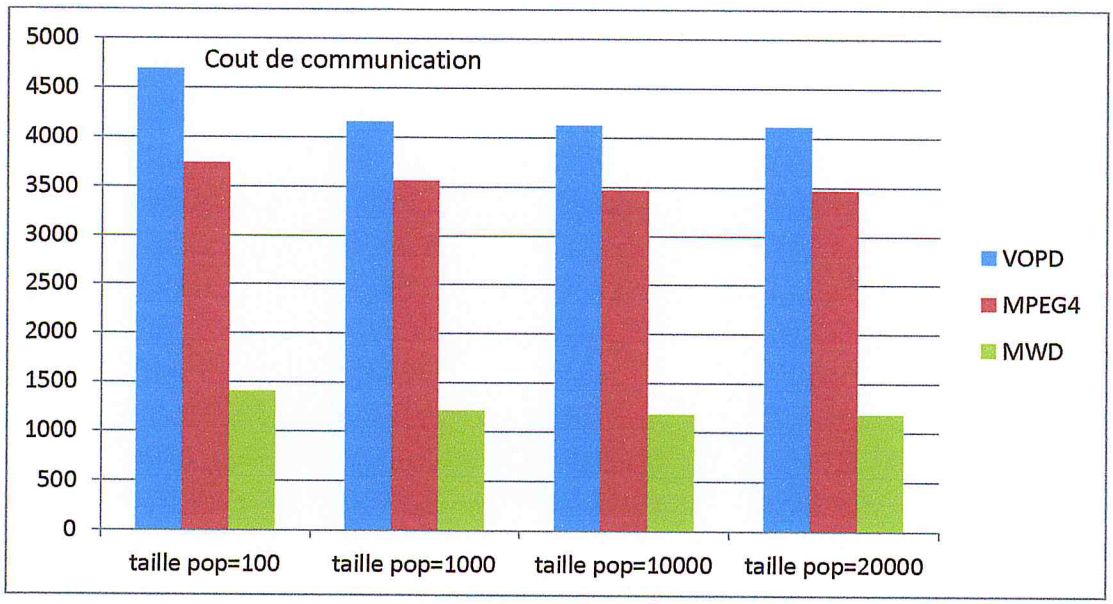

Figure 36 : comparaison selon la taille de la population pour VOPD, MPEG4 et MWD

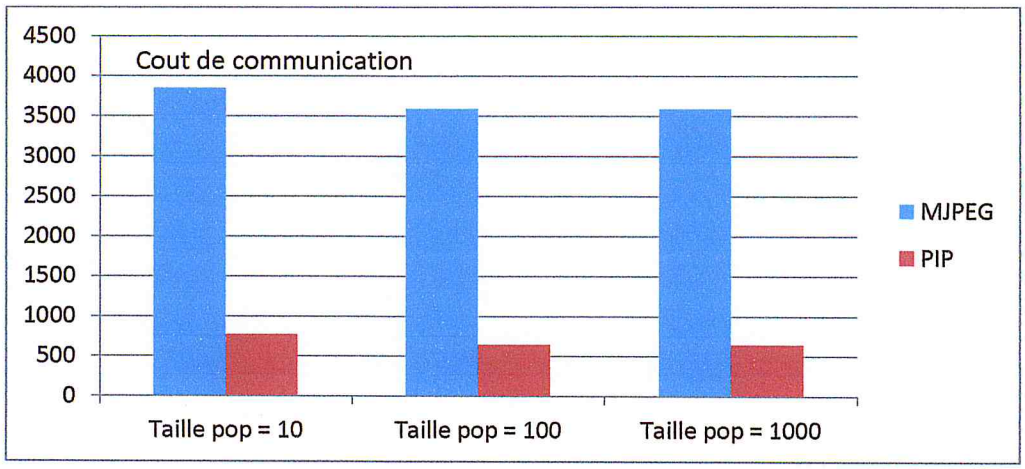

Figure 37 : comparaison selon la taille de la population pour MJPEG et PIP

D'après la figure on remarque que tant que la taille de la population augmente tant qu'on obtient de meilleurs résultats.

- Selon le facteur de mutation F :

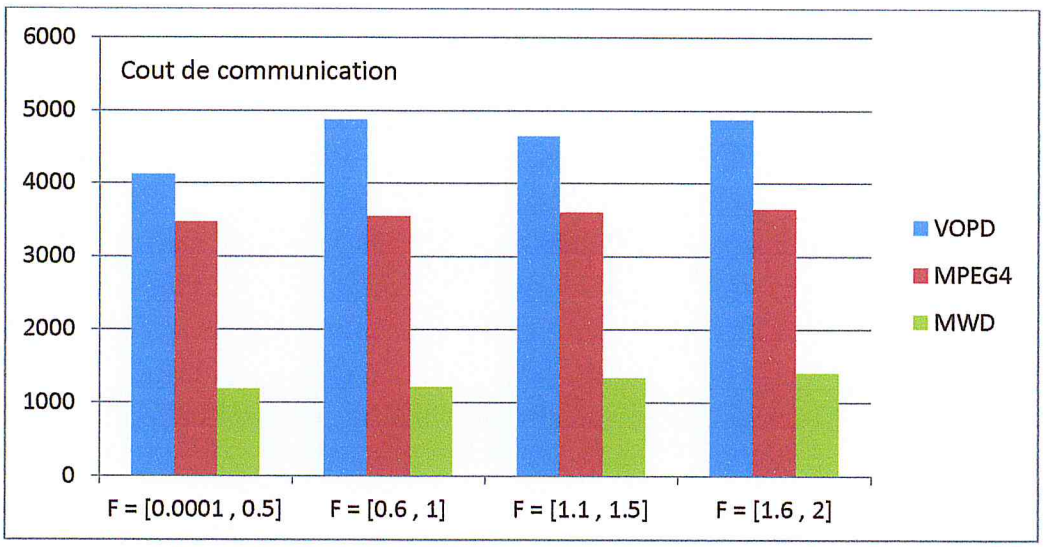

Figure 38 : comparaison selon facteur de mutation F

D'après ce diagramme on constate que, plus F est proche du 0 plus en a de meilleurs résultats.

- Selon le facteur de croisement :

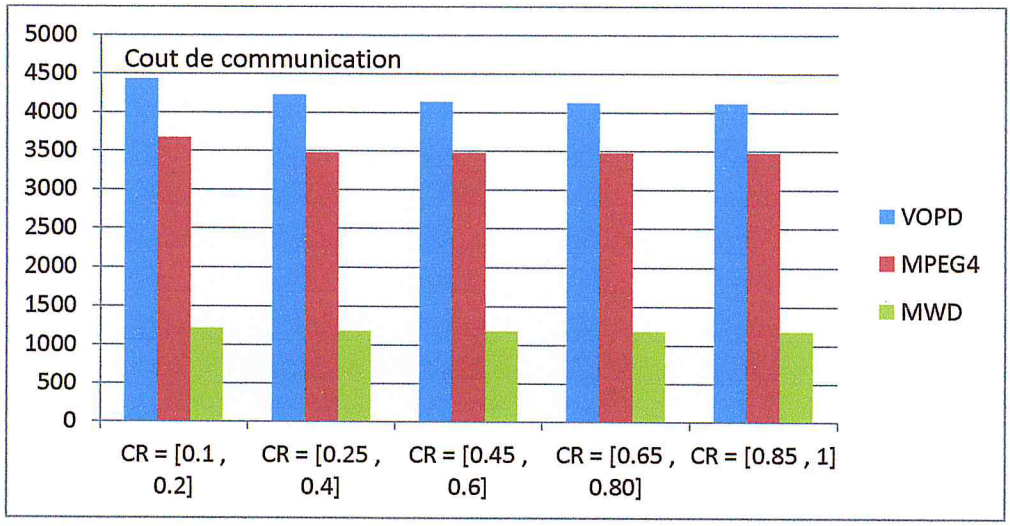

Figure 39 : comparaison selon facteur de croisement CR

D'après les résultats représenté par cette figure on constate que tant que le CR s'éloigne du 0 tant que les résultats son bons.

- Selon la technique de mutation :

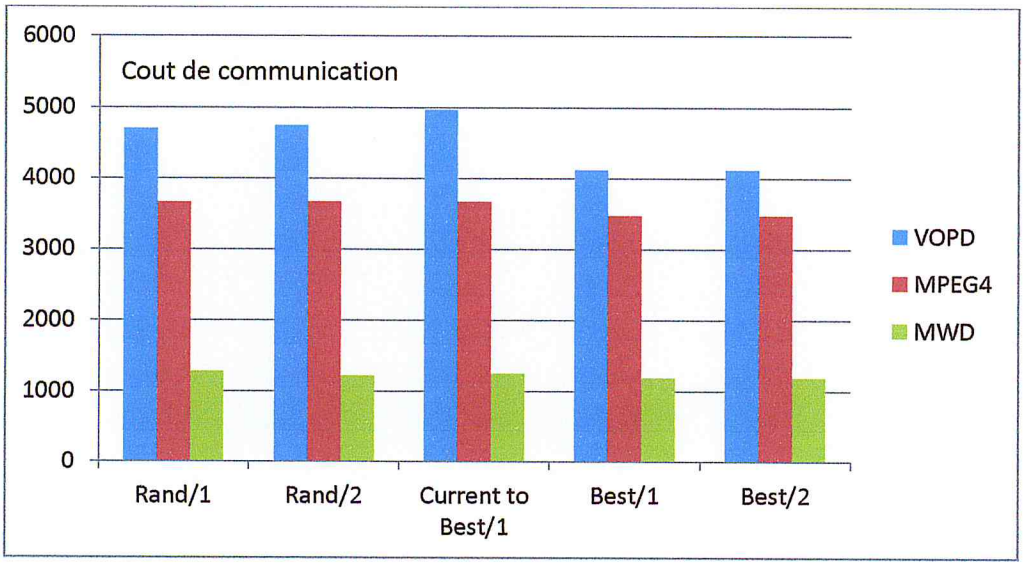

Figure 40 : comparaison selon type de mutation

D'après cette figure on remarque clairement que les meilleurs type de mutation son les deux Best/1 et Best/2.

- Selon la taille du NoC :

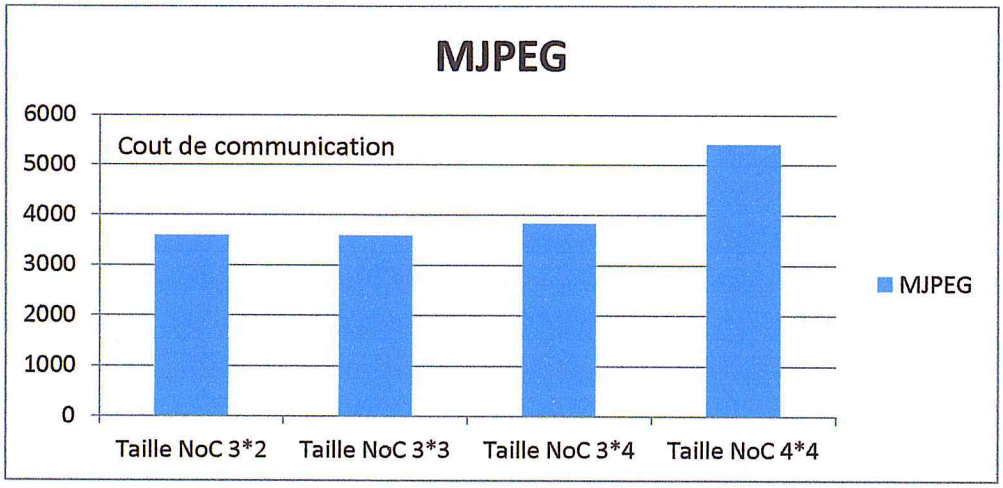

Figure 41 : comparaison selon taille du NoC pour MJPEG

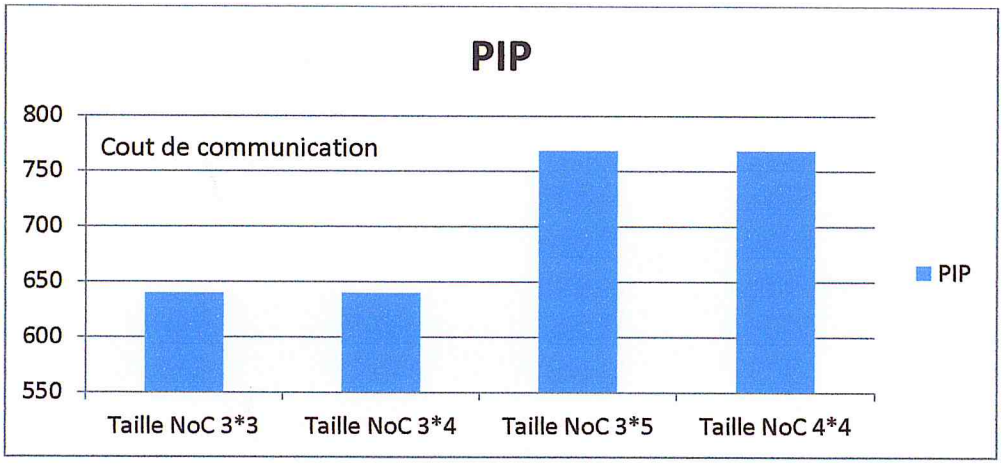

Figure 42 : comparaison selon taille du NoC pour PIP

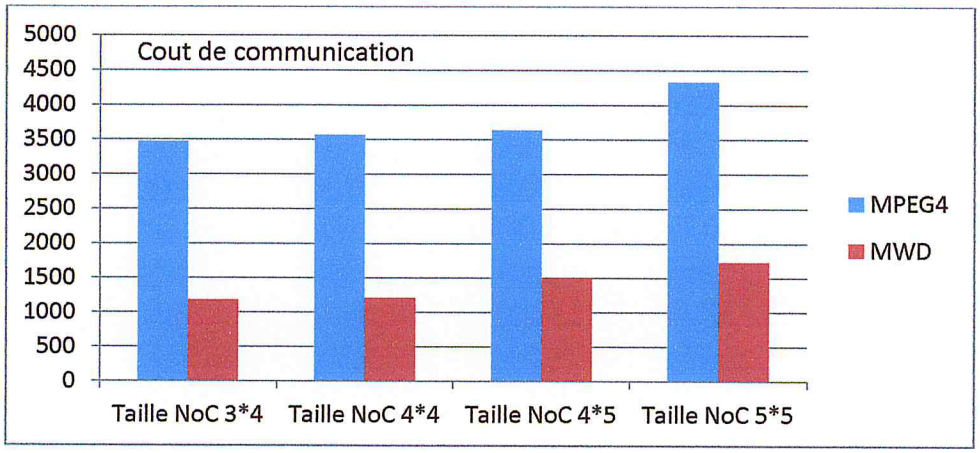

Figure 43 : comparaison selon taille du NoC pour MPEG4 et MWD

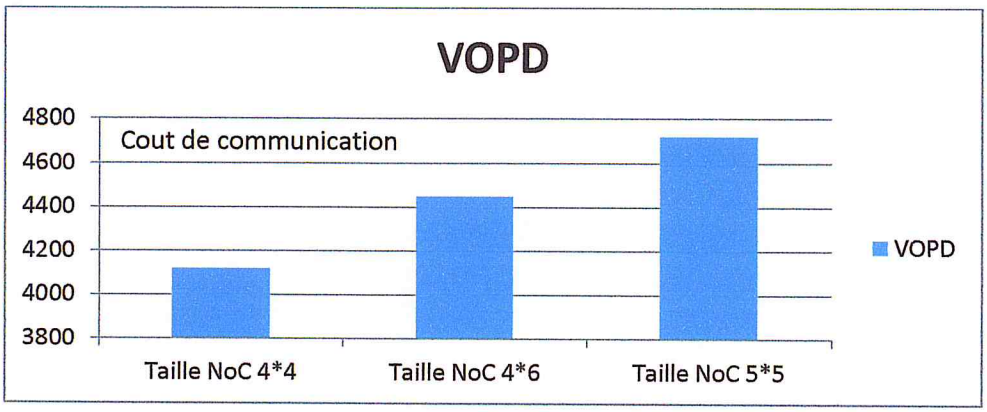

Figure 44 : comparaison selon taille du NoC pour VOPD

D'après ces résultats on constate que tant que la taille de la topologie du NoC est égale ou beaucoup plus proche du nombre d'IP du graphe d'application, tant que les résultats sont meilleurs.

- Selon les résultats obtenus dans la littérature :

A fin de prouvé que l'algorithme de mapping « DE » qu'on a utilisé a apporté des améliorations par rapport au technique déjà existante, on a comparé les résultats qu'on a obtenus avec les résultats de d'autres techniques existante. On n'a pas implémenté ces techniques, mais on a juste utilisé leurs résultats publiés pour faire la comparaison.

C'est techniques avec les quelles on a comparé nos résultats obtenus par le benchmark VOPD sont CMAP [22], GMAP [22], CGMAP [22], SPIRAL [22], GBMAP [64], Citrine [62], Onyx [63].

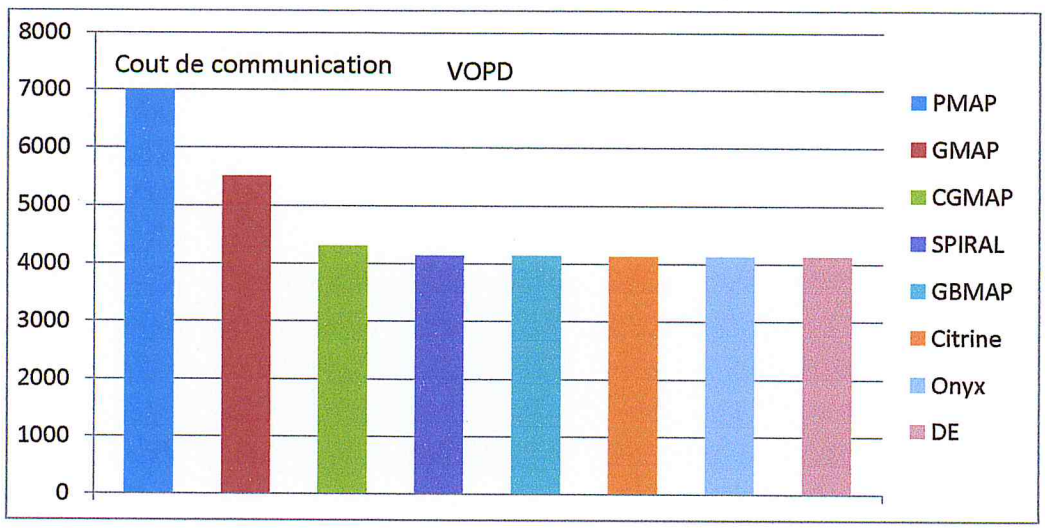

Figure 45 : comparaison entre les résultats obtenus et ceux de la littérature pour VOPD

Et pour les résultats obtenus par le benchmark MPEG4 on les a comparés avec les techniques suivantes NMAP [64], GBMAP [64], Citrine [62], Onyx [63]

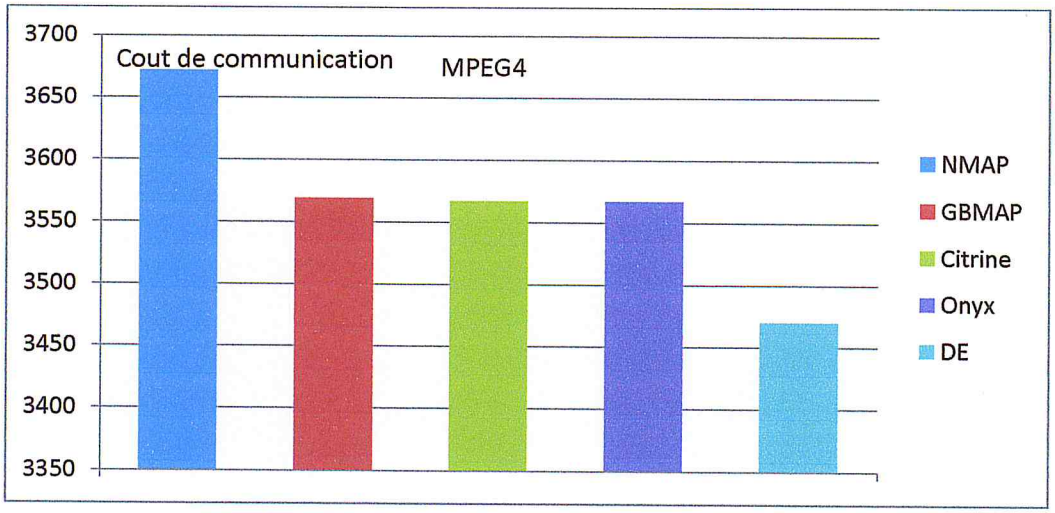

Figure 46 : comparaison entre les résultats obtenus et ceux de la littérature pour MPEG4

On constate que la méthode de placement qu'on a utilisé et qui se base sur l'algorithme évolutionnaire « DE » est une bonne méthode de placement, car elle nous permit d'obtenir les résultats optimum qui existe dans la littérature jusqu'à présent, et même d'obtenir un meilleur résultat pour le MPEG4 que celui qui existe.

✓ <u>Pour le temps d'exécution :</u>

- selon le cout de communication

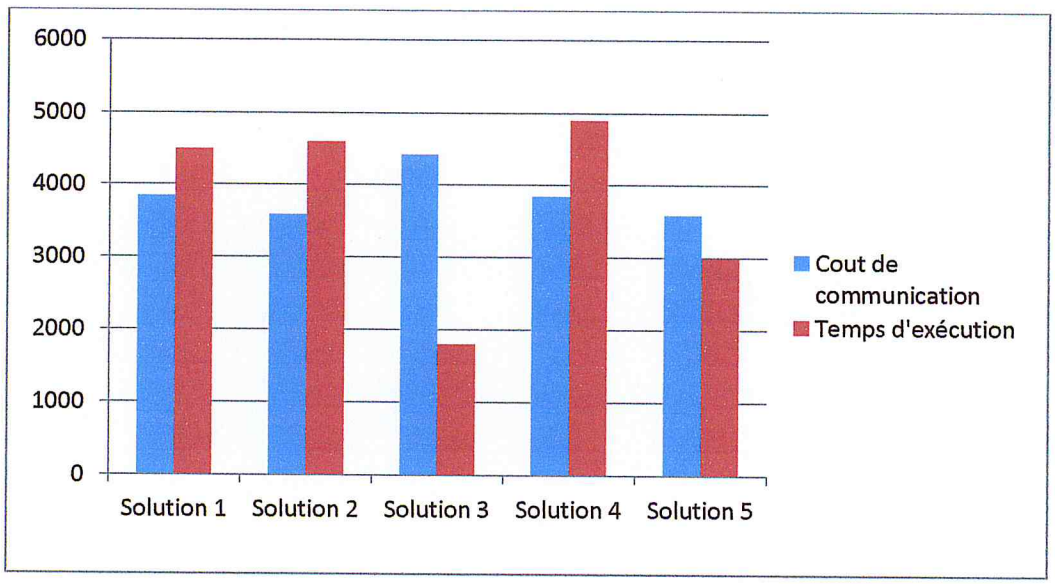

Figure 47 : Temps d'exécution de MJPEG selon ASAP

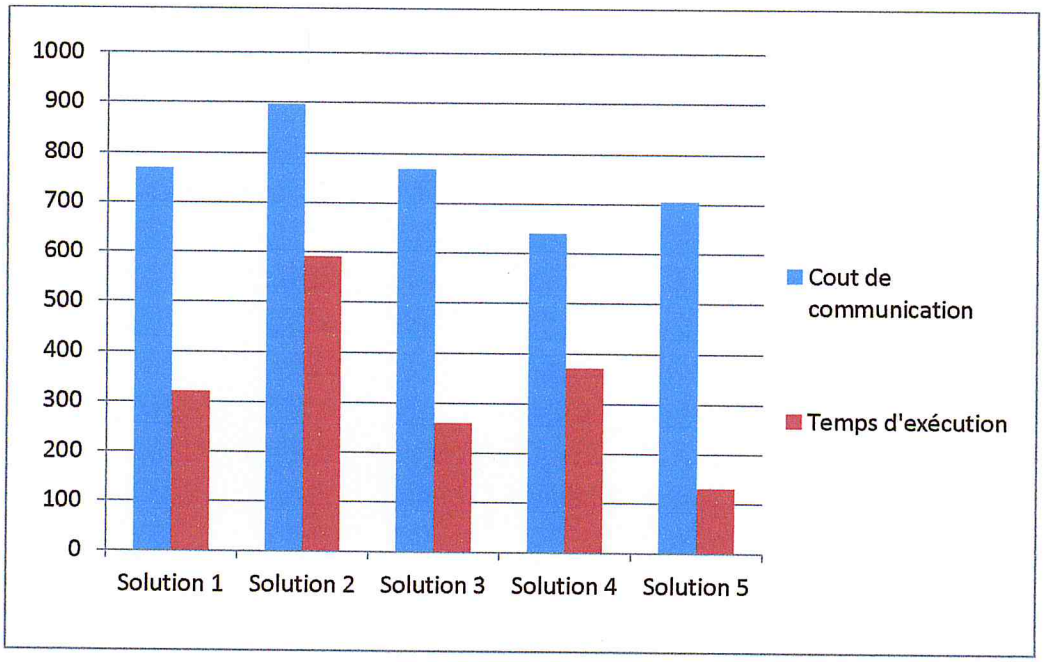

Figure 48 : Temps d'exécution de PIP selon ASAP

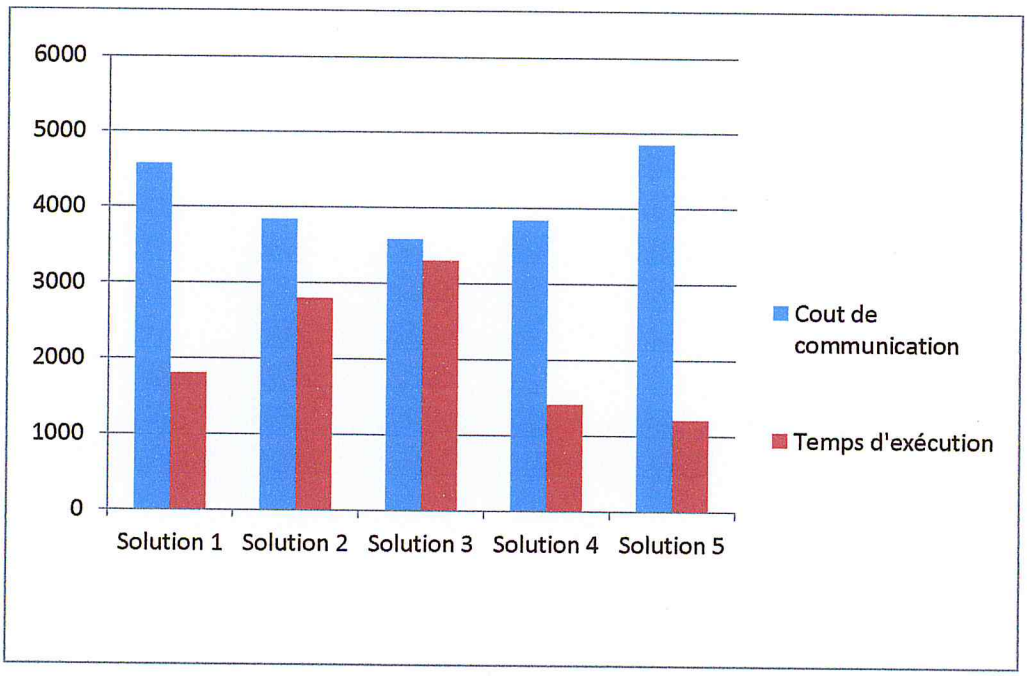

Figure 49 : Temps d'exécution de MJPEG selon ALAP

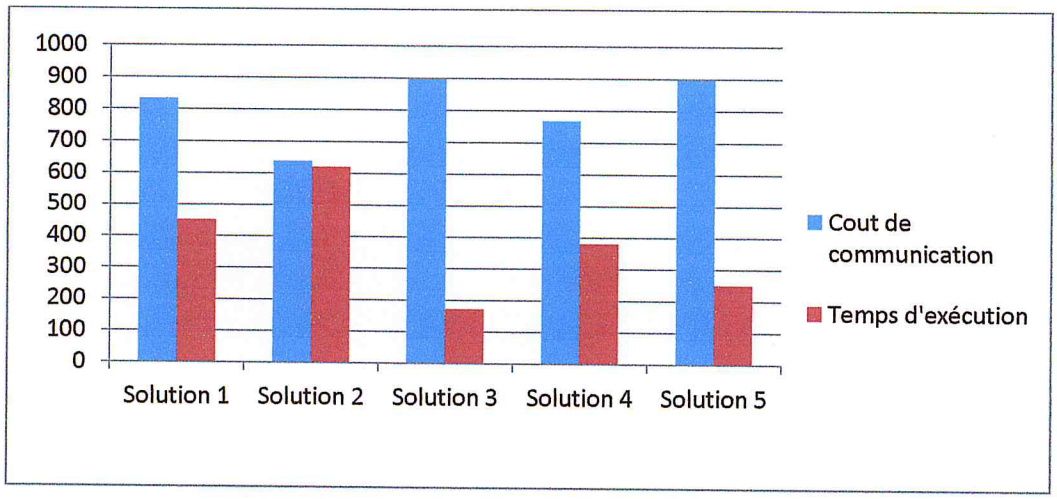

Figure 50 : Temps d'exécution de PIP selon ALAP

D'après ces résultats on peut voir clairement que le cout de communication n'a pas d'influence sur le temps d'exécution, mais c'est le placement des tâches qui est en relation directe avec le temps d'exécution.

- Selon les algorithmes « ASAP » et « ALAP » :

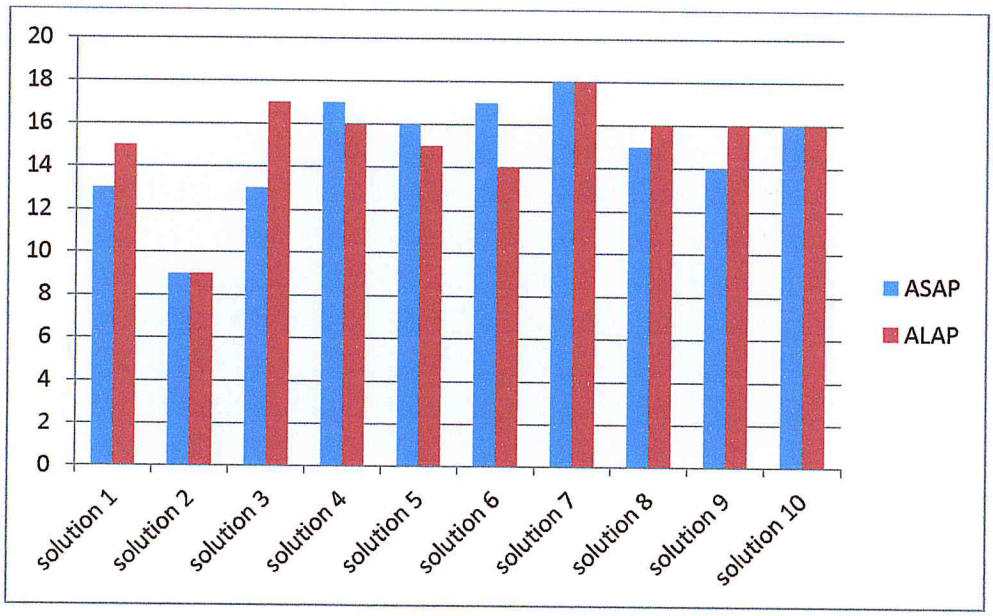

Figure 51 : Temps d'exécution selon « ASAP » et « ALAP »

D'après la figure on peut voir qu'il n'y a pas une grande différence entre les temps d'exécution obtenus par « ASAP » et ceux de « ALAP », quoi que « ASAP » soit légèrement meilleur car dans 10 solutions, il y a 4 ou il est meilleur, contre 3 pour le « ALAP ».

9. Conclusion :

Notre but était d'optimiser le temps dans la conception des NoC, ce temps qu'on veut minimiser c'est le cout de communication + le temps d'exécution, pour ce fait on a utilisé une méthode qui se base sur les algorithmes évolutionnaires qui est l'algorithme « DE » ainsi que deux algorithme d'ordonnancement qui sont le « ASAP » et « ALAP ».

Après l'implémentation de tous ces algorithmes et la réalisation des tests sur les résultats obtenus, on a constaté que la méthode de placement utilisée est performante au niveau cout de communication, et elle nous permit d'obtenir les résultats optimums qui existent dans la littérature jusqu'à présent, mais pour cela il faut qu'un bon réglage des paramètres liés à cet algorithme soit respecté.

Tandis que pour le temps d'exécution le facteur qui influence sur ce dernier c'est le résultat du placement des tâches de l'application sur les tuile de l'architecture. Et en ce qui concerne les algorithmes d'ordonnancement « ASAP » et « ALAP » ils permettent de faire l'ordonnancement des tâches de l'application, chacun selon son principe mais à la fin ils donnent tous les deux le séquencement d'exécution des tâches.

Conclusion générale :

Aujourd'hui, pour répondre aux besoins de nouvelles applications, les concepteurs intègrent de plus en plus d'unité de traitement dans les systèmes sur puce. La quantité de donnée échangée dans les systèmes a augmenté. Ainsi, la gestion au sein de l'architecture de l'application devient complexe. Dans ce contexte, les problèmes de mapping et d'ordonnancement apparaissent, c'est l'un des sujets de recherche le plus exploité dans le domaine de l'informatique. Comme c'est des problèmes NP-Difficiles qui tentent à optimiser un ou plusieurs objectifs. L'utilisation de méthodes exactes en vus de l'obtention de solution optimale semble non réalisable.

L'objectif de notre travail consiste à traiter le problème de mapping et d'ordonnancement d'une manière simultané. Pour cela, on a choisi les deux algorithmes « ASAP » et « ALAP » pour réaliser l'ordonnancement, comme c'est des algorithmes simple a mettre en œuvre et qui donne de bon résultats. Tandis que pour le mapping, d'après les recherches effectuées on a remarqué que l'algorithme évolutionnaire « DE » n'a pas été beaucoup utilisé auparavant dans notre problème, c'est pour cela qu'on a décidé d'utiliser cet algorithme pour résoudre le problème de mapping, et il c'est avérer que c'est un très bon algorithme qui donne de bons résultat pour l'optimisation du cout de communication. Et cela pour optimiser notre fonction objective *f* qui consiste à minimiser la somme entre le cout de communication obtenu par le mapping et le temps d'exécution relatif a ce dernier.

Et comme il n'existe pas de plateforme qui permet de tester nos algorithme, on a essayé d'implémenté une petite application pour réaliser ces tests.

Coté mémoire, on a commencé par définir dans un premier temps toutes les notions essentielles qui sont en relation avec le contexte de notre travail, et par la suite, nous avons décrit d'une manière détaillé les techniques choisis qui sont ASAP /ALAP pour la phase ordonnancement et l'algorithme DE (Differential evolution) pour le mapping.

Enfin, plusieurs expérimentations sont effectuées sur des Benchmarks retenu comme échantillons de test pour prouver l'efficacité des algorithmes utiliser et implémenter.

Perspectives :

Un nombre d'enrichissements pourra être apporté a notre travaille comme :

- Proposer une amélioration pour l'algorithme de mapping :

- Faire une hybridation du « DE » avec une méta-heuristique performante, pour obtenir de meilleurs résultats.

- Proposer une amélioration pour l'ordonnancement :

- Implémenter l'algorithme d'ordonnancement « Liste scheduling » qui se base sur les résultats du « ASAP » et « ALAP » pour réaliser l'ordonnancement.

- calculer le cout de communication entre les tâches des cycles du résultat de l'ordonnancement.

- Faire une optimisation multicritères.
- Utiliser une architecture de réseaux sur puce hétérogène.

Bibliographie :

[1] : Mohamed akli REDJEDAL, Abdelhak OULHACI « Ordonnancement Hi´erarchique d'Application Temps R´eel Sur Une Architecture NOC » M´EMOIRE DE FIN D'´ETUDE pour l'obtention du Dipl^ome d'Ing´enieur d'Etat en Informatique, 2009

[2] : Christophe BLANC « Architecture des systèmes à processeurs – IUT GEII (ISI-II2) -2 » http://www.christophe-blanc.info/ IUT de Montluçon Département Génie Electrique et Informatique Industrielle 2009

[3] : Multiprocessor System-on-Chip (MPSoC) Technology Wayne Wolf, Fellow, IEEE, Ahmed Amine Jerraya, and Grant Martin, Senior Member, IEEE

[4] : ATAT Youssef « Conception de haut niveau des MPSoCs à partir d'une spécification Simulink : Passerelle entre la conception au niveau Système et la génération d'architecture ». Thèse de doctorat. Institut National Polytechnique de Grenoble, 2007.

[5] : BOUCHHIMA Aimen « Modélisation du logiciel embarque à différents niveau d'abstraction en vue de la validation et la synthèse des systèmes monopuces ». Thèse de doctorat. Institut National Polytechnique de Grenoble (INPG), 2006

[6] : AROUI Abdelkader « Ordonnancement multiobjectifs d'applications intensives sur architectures régulières embarquées » Thèse de magister en informatique. UNIVERSITE D'ORAN ES-SENIA, 2012

[7] : Romaine Lemaire « conception et modélisation d'un système de contrôle d'applications de télécommunication avec une architecture de réseau sur puce(NoC) », thèse de doctorat, institut national polytechnique de Grenoble, 2006

[8] : DELORME Julien « Méthodologie de modélisation et d'exploration d'architecture de réseaux sur puce appliquée aux télécommunications », thèse de doctorat, 'Institut national des sciences appliquées de Rennes, 2007

[9] : Réseau d'Interconnexion pour les Systèmes sur Puce : le Réseau HERMES, Séverine Riso, Lionel Torres, Gilles Sassatelli, Michel Robert, Laboratoire d'Informatique de Robotique et de Microélectronique de Montpellier, Université de Montpellier II / CNRS

[10] : Antoine SCHERRER « Analyses statisitiques des communications sur puce » thèse de doctorat l'Ecole Normale Sup´erieure de Lyon, 2006

[11] : Samuel Evain « µSpider Environnement de Conception de Réseau sur Puce » thèse de doctorat institut national des sciences appliqués de Renne, 2006

[12] : Nicolas Ngan, « Etude et conception d'un réseau sur puce dynamiquement adaptable pour la vision embarquée », THESE pour obtenir le grade de Docteur de l'Universit e Paris-Est, 2011

[13] : Abdellah Medjadji Kouadri Mostéfaoui, «Architectures Flexibles pour la Validation et l'Exploration De Réseaux Sur Puce» INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE 2009

[14] : JESSICA ALLARD BERNIER, «MÉTHODE DE RECONFIGURATION DYNAMIQUE POUR UN RÉSEAU-SUR-PUCETOLÉRANT AUX FAUTES» Mémoire de maîtrise, École Polytechnique de Montréal. 2011

[15] : Xuan-Tu TRAN «Méthode de Test et Conception en Vue du Test pour les Réseaux sur puce Asynchrones : Application au Réseau ANOC», thèse de doctorat, INSTITUT POLYTECHNIQUE DE GRENOBLE, février 2008

[16] : MCKINLEY Philip and NI Lionel « A Survey of Wormhole Routing Techniques in Direct Networks ». Computer, 1993

[17] : Initiation à la modélisation et co-simulation comportementale C-VHDL d'un réseau de communication sur Puce (Network on Chip) C.Tanougast1, S. Jovanovic2, F. Monteiro1, C. Diou1, A. Dandache1 Laboratoire des Interfaces Capteurs et Microélectronique (L.I.C.M.) Université Paul Verlaine de Metz, LICM-ISEA, 7 rue Marconi, 57070 Metz Technopole Université Henri Poincaré – Nancy Université, BP 239, 54506 Vandœuvre-Les-Nancy

[18] : Adrijean A NDRIAHANTENAINA Alain G REINER, « M ICRO - RÉSEAU POUR SYSTÈMES INTÉGRÉS : RÉALISATION D ' UN RÉSEAU SPIN À 32 PORTS », Département ASIM, laboratoire LIP6, Université Pierre et Marie C URIE

[19] : QNoC: QoS architecture and design process for Network on Chip, Evgeny Bolotin, Israel Cidon, Ran Ginosar and Avinoam Kolodny, lectrical Engineering Department Technion—Israel Institute of Technology, Haifa 32000, Israel

[20] : Meriem ZIDOUNI, « Modélisation et analyse des performances de la bibliothèque MPI en tenant compte de l'architecture matérielle », thèse de doctorat, Université Joseph Fourier - Grenoble, 2010

[21] : SCHERRER Antoine « Analyses statistiques des communications sur puce ». Thèse de doctorat, Ecole Normale Supérieure de Lyon, 2006

[22] : Yousra Abir LARIBI « Environnement de Mapping pour la Conception des Réseaux sur Puce (NoCs) » Mémoire de fin d'études Pour l'obtention du diplôme d'Ingénieur d'Etat en Informatique,école nationale supérieure d'informatique,2010

[23] : SUNMAP: A Tool for Automatic Topology Selection and Generation for NoCs Srinivasan Murali, Giovanni De Micheli Computer Systems Lab Stanford University Stanford, CA-94305, USA

[24] : ANDRE Françoise et PAZAT Jean-Louis « Le placement de tâches sur des architectures parallèles ». Revue TSI : Technique et science informatiques.

[25] : CARVALHO Ewerson, MARCON César, CALAZANS Ney and MORAES Fernando « Evaluation of Static and Dynamic Task Mapping Algorithms in NoC-Based MOEPCs ».Proceedings of the 11th international conference on System-on-chip, 2009.

[26] : DELORME Xavier « Optimisation combinatoire et problèmes de capacité d'infrastructure ferroviaire ». Thèse de doctorat, Université de Valenciennes et du Hainaut Cambrésis, Valenciennes, France, Décembre 2003.

[27] : FREVILLE Arnaud « Méthodes de recherche locale ». Journée AFPLC - École des Mines de Nantes, 2000.

[28] : Srinivasan Murali, Giovanni De Micheli. «Srinivasan Murali, Giovanni De Micheli. «Bandwidth-Constrained Mapping of Cores onto NoCArchitectures».Journal. Computer Systems Lab Stanford University.2004

[29] : MOEIN-DARBARI Fahime, KHADEMZADE Ahmad and GHAROONI-FARD Golnar « CGMAP: A new approach to network-on-chip mapping problem ». IEICE Electronics Express, 2009.

[30] : KOZIRIS Nectarios, ROMESIS Michael, TSANAKAS Panayiotis and PAPAKONSTANTINOU George « An Efficient Algorithm for the Physical Mapping of

Clustered Task Graphs onto Multiprocessor Architectures ».Proc. of 8th Euromicro Workshop on Parallel and Distributed Processing, IEEE, 2000.

[31] : MEHRAN Armin, SAEIDI Samira and KHADEMZADEH Ahmed « DSM:A heuristic dynamic Spiral mapping algorithm for network on chip ».IEICE Electronics Express,2008.

[32] : Zhou Wenbiao, Yan Zhang « Link-load Balance Aware Mapping and Routing for NoC».journal.Harbin Institute of Technology Shenzhen Graduate Schoo.2007.

[33]: A.Khademzadeh,S.Pourkiani,M.Yaghobi."GBMAP:An Evolutionary Approch to Mapping Cores onto a Mesh-based Noc Architecture".Journal of Communication and Computer,ISSN 1548-7709,USA.2010.

[34]: M.Janidarmian,A.Khademzadeh,M.Tavanpour."Onyx: A new heuristic bandwidth-constrainted mapping of cores onto tile-base Network on Chip".CE Department,Islamic Azad University,Science and Research Branch,Tehran,Iran,2009.

[35]: M.Janidarmian,A.Khademzadeh,A.Ferk,V.Bokharaei," Citrine: A Methodologie for Application-Specific Network-on-Chips Design".Proccedings of the World Congress on Engineering and Computer Science .San Francisco,USA.2010.

[36] : Olivier Liess, Serigne Gueye, « Problématique d'ordonnancement Ferroviaire»,18 juin 2008.

[37] : Amokrane Samah « Algorithme Génétique pour le problème d'ordonnancement dans la synthèse de haut niveau pour contrôleur dédié », en vus de l'obtention du diplôme du magister, faculté science de l'ingénierie, département informatique, Université Batna.

[38] : N.Hamani, « Application des fourmis quantiques au partitionnement avec Ordonnancement en Codesign ».en vus de l'obtention du grade du Magister. Institut national d'informatique D'Alger.

[39] : R.MarculescuG.Varatkar. Communication, complexity and critically issue in designing silicon networks. In Date, 2006.

[40] : T.Lei and S. Kumar. A two-step genetic algorithm for mapping task graphs to A network on chip architecture. In Euromicro Symposium on, editor, Digital System Design, pages 180-187. 0-7695-2003-0, 2003.

[41] : J.Hu and R.Marculescu. Energy-aware communication and task scheduling for Network-on-chip architectures under real-time constraints. Design, Automation And Test in Europe Conference and Exhibition, 1:10234, 2004.

[42] : D.Shin and J.Kim. Power-aware communication optimization for networks-on- Chips with voltage scalable links. Proceedings of the 2nd IEEE/ACM/IFIP international conference on Hardware/software code- sign and system synthesis, pages 170{175, New York, NY, USA, 2004.

[43]: Cédric Joncour, « Cours d'optimisation Combinatoire et Méta heuristiques-Approche avancé de résolution exacte ».Master 2 -semestre 2.2012

[44]: S.BenIsmail « Introduction à l'optimisation combinatoire», Majeure informatique. INF413-2eme semestre.2012.

[45]: Clarisse Dhaenes, El-Ghazali Talbi, « Optimisation multicritère: approche par méta heuristiques », laboratoire d'informatique fondamentale de Lille. Université de lille1.France.

[46]: Walid Tfaili, « Conception d'un algorithme de colonie de fourmis pour l'optimisation continue dynamique ». Thèse de doctorat. Université Paris 12-val de marine.13/12/2007.

[47]: Abdesselem.Layeb, « Introduction au méta –heuristique ».2009

[48]: Gilles Schaeerr, « Recherche Locale », Enseignement polytechnique Informatique. INF550.Cours1112.

[49]: Alain Heatz, « le méta heuristiques ». Ecole polytechnique .Canada.

[50]: Ben Mohamed Ahmed. « Résolution approchée du problème de BIN-PACKING ».thèse de doctorat. Université havre.03/12/2009.

[51]: Sébastien verel .Manuel Clergue. « Méta heuristiques pour l'optimisation combinatoire », mode de comptabilité.2002.

[52]: Benallal.H. « Résumé de la recherche Tabou ».Master1.2013.

[53]: François Gueguen. « Rapport de projet Recuit-Simulé ».Master I IL .2011

[54]: Eric « Recuit-Simulé ».Master I.25/06/2010

[55]: ABDUL STTAR ISMAIL WDAA, « DIFFERENTIAL EVOLUTION FOR NEURAL NETWORKS LEARNING ENHANCEMENT », UNIVERSITI TEKNOLOGI MALAYSIA, 2008/2009

[56]: COSTANZO Andrea, LUONG Thé Van., MARILLGuillaume, « Optimisation par colonies de fourmis ».19 mai 2006

[57] : Majid.Janidarmian,Atena.Roshan Fekr. « A Survey of Meta-heuristic Solution Méthods for Mapping problem in network on chips ». CE department Science and Research Branch,Islamic Azad University,Tehran,Iran.

[58]: BENYAMINA Abou Elhassen, « Ordonnancement Hierarchique Multi-Objectif D'Application Embarquees Intensives », THESE De Doctorat D'Etat, Universite d'Oran-Es-Senia, 2009

[59]: Abbas EL DOR, « Perfectionnement des algorithmes d'Optimisation par Essaim Particulaire. Applications en segmentation d'images et en électronique», THÈSE DE DOCTORAT EN INFORMATIQUE, UNIVERSITÉ PARIS-EST, décembre 2012

[60]: BERTOZZI Davide, JALABERT Antoine, MURALI Srinivasan and Student Member « NoC Synthesis Flow for Customized Domain Specific Multiprocessor Systems-on-Chip ». IEEE Transactions on parallel and distributed systems, Vol 16, N° 2, February 2005.

[61]: Jae Young Hur · StephanWong · Todor Stefanov « Design Trade-offs in Customized On-chip Crossbar Schedulers », 2008

[62]: Majid Janidarmian1, Ahmad Khademzadeh2, Atena Roshan Fekr1, Vahhab Samadi Bokharaei3, « Citrine: A Methodology for Application-Specific Network-on-Chips Design »,Proceedings of the World Congress on Engineering and Computer Science 2010 Vol I WCECS 2010, October 20-22, 2010, San Francisco, USA

[63]: Majid Janidarmian1a), Ahmad Khademzadeh2, and Misagh Tavanpour1 « Onyx: A new heuristic bandwidth-constrained mapping of cores onto tile-based Network on Chip », IEICE Electronics Express, Vol.6, No.1, 1–7, 2009

[64]: Misagh Tavanpour1, Ahmad Khademzadeh2, Somayyeh Pourkiani3 and Mehdi Yaghobi3 « GBMAP: An Evolutionary Approach to Mapping Cores onto a Mesh-based NoC

Architecture », Mar. 2010, Volume 7, No.3 (Serial No.64), Journal of Communication and Computer, ISSN 1548-7709, USA

Webographie :

[site1]: http://webee.technion.ac.il/people/bolotin/QNoC/main.htm

[site2]: http://www3.pucrs.br/portal/page/portal/pucrs/Capa/

[site3]: http://www.arteris.com/

[site4]: http://rd.springer.com/chapter/10.1007/978-1-4020-6488-3_12

[site5]: http://reussirlem1info.files.wordpress.com/2013/02/colonie-dabeilles-resume.docx

[site6]: www.scholarpedia.org/article/Artificial_bee_colony_algorithm

[site7]: http://fr.wikipedia.org/wiki/Optimisation_par_essaims_particulaires

[site8]: http://fr.wikipedia.org/wiki/MPEG-4

[site9]: http://www.clashinfo.com/dico/definition-c/art44-codec.html