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 :

ERRAHMANI Hossein

En vue d'obtenir le diplôme de Master

Domaine : Mathématique et Informatique

Filière : Informatique

Spécialité : GSI

Thème :

MAPPING AUTOMATIQUE DANS LES RÉSEAUX SUR PUCES

Encadré pardevant :

- *M. Bougherara*
- *Mlle. Zahra*

le jury composé de :

- Mme. Boumahdi
- Mme. Rezoug
- Mme toubaline

Année Universitaire

2012/2013

Dédicace

Je dédie ce travail à :

A la plus ma merveilleuse mère dans le monde

Et a mon cher père

A mes chères sœur Ghania, Zahra, Aicha, Yamna, Omelkayer et Loubna

Et à mes frères Mohamed, Djillali, Rabeh, Jaifer et Athman.

A mon promoteur Mr.Bougherara

A mon encadreur Melle.Zahra

A mes amies et mes collègues

A tout qui m'aiment et qui j'aime

Remerciement

Nous remercions ***« Dieu »*** *de nous avoir accorde les connaissances de la science et de nous avoir aidé à réaliser ce travail.*

Au terme de ce modeste travail nous tenons à remercier chaleureusement et respectivement tous ceux qui ont contribués de prés ou de loin à la réalisation de ce modeste projet de fin d'étude, à savoir l'encadreur ***M.Bougherara.***

Nos vifs remerciements vont tous d'abord à ***Mlle.Zahra*** *qui nous a aides à développer notre travail avec l'aide de* ***Mlle Khadidja*** *qu'on remercie également.*

Nous tenons à remercier tous ***les Enseignants*** *qui nous ont suivis durant notre formation.*

Et Merci à Tous...

Abstract

Network on chip is a new concept in SoC interconnections. This structure facilitate complex components integration and seems adapt to applications evolution. However, as it's a new technology, it requires efforts in research, especially for the acceleration and simplification of design phases.

Mapping is a central step in the flow of NoC conception. It consists of placing different IPs of the application on NoC's tiles. The goal is to minimize communications cost energy consumption or system's latency while respecting bandwidth or real time constraints. It's necessary to develop tools and methods to automate this process.

It's in this context that fits our work. It consists of proposing a new technique for mapping applications onto NoC architecture in order to minimize communications cost. This new solution is based on hybridization of evolutionary methods genetic algorithm and ant colony optimization.

To evaluate the new technique, we propose to implement plate-forme for tests and simulations. It can perform mapping, compare different techniques of solving this problem and simulate traffic on NoCs in order to evaluate their performances.

Keywords

Network on chip, System on Chip, communication, mapping, genetic algorithm ant colony optimization.

Résume

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.

La phase de mapping représente une phase centrale lors de la mise en œuvre d'un réseau sur puce. Elle permet de placer les éléments d'une application sur l'architecture. Le but est de minimiser le coût de communications, la consommation d'énergie ou la latence du système tout en respectant les contraintes de bande passante ou de temps réel. Il devient nécessaire d'élaborer des outils et des méthodes qui automatisent ce processus.

C'est dans ce cadre que s'insère notre travail. Il consiste à proposer des nouvelles techniques de placement des IPs d'une application sur les différents éléments d'une architecture de réseau sur puce afin de minimiser le coût de communications. Ces nouvelles solutions sont basée sur l'optimisation par les algorithmes génétiques, les colonies de fourmis et l'hybridation des deux algorithmes (GANT).

Afin d'évaluer les performances de cette technique, développer une nouvelle plateforme de tests et de simulations qui permet d'effectuer un mapping.

Mots clés :

Réseaux sur puce, Systèmes sur puce, communication, mapping, les algorithmes génétiques et Optimisation par colonies des fourmis.

Table des matières

Liste des Figures

Liste des Tableaux

Introduction Générale

Introduction Générale

Les semi-conducteurs constituent les briques élémentaires des industries modernes. Les technologies qui leur sont associées jouent un rôle essentiel dans tous les secteurs. Leurs perpétuelles évolutions a donné naissance à des systèmes composés de millions de transistors sur une seule et même puce. C'est ce que l'on appelle les systèmes sur puce ou SoC (pour « *System On Chip* »). Ces systèmes intègrent différents éléments comme des processeurs, des mémoires, des blocs d'entrée/sortie ou encore des médias de communications.

Les progrès de la technologie et l'évolution des besoins font que la durée de vie de ces architectures diminue. Afin d'y remédier, de plus en plus de fonctionnalités sont introduites dans un même système, ce qui conduit à la mise en communication d'un grand nombre de blocs fonctionnels de natures hétérogènes. Cependant, les liens de communication n'évoluent pas à la même vitesse et deviennent un goulot d'étranglement.

Les solutions de communication actuelles telles que le bus partagé trouvent leurs limites en termes de bandes passantes et d'extension à mesure que le nombre d'éléments communicants augmente. Cette structure est la plus utilisée de nos jours mais ne semble pas s'adapter aux applications futures. C'est dans ce contexte que le concept des réseaux sur puce ou NoC (*Networks On Chip*) a vu le jour.

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.

La phase de mapping (placement physique) est une phase centrale dans la conception des NoCs, dont le résultat a un impact direct sur les performances du système. Elle consiste à placer chaque élément de l'application sur l'architecture du réseau sur puce de sorte à minimiser le coût de communications.

Notre travail consiste à étudier certaines des principales techniques de mapping existantes et proposer des nouvelles solutions qui permettent de mapper une application sur une architecture de NoC tout en minimisant le coût de communications.

Le document est organisé en 5 chapitres :

Le premier chapitre va traiter les concepts de base de systèmes sur puce, MPSoCs, réseaux sur puce, les différentes structures d'interconnexions ainsi que les limites de chacune.

Le second chapitre introduit le problème de placement physique lié aux réseaux sur puce. Les méta-heuristiques qui permettent de résoudre ces problèmes, quelques techniques proposées dans la littérature sont introduites et une étude comparative est présentée.

Le troisième chapitre est consacré à présenter les méthodes des résolutions, présentées les solutions qui sont base sur les algorithmes génétiques, colonies des fourmis, les algorithmes génétiques avec croisement shift et l'hybridation des colonies des fourmis avec les algorithmes génétiques et la conception de la plate-forme développée.

Dans le dernier chapitre, des tests sont effectués afin de comparer les techniques proposées aux techniques de littérature.

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

Chapitre 1 : les systèmes sur puce

Introduction

Dans ce chapitre nous définirons les notions de base du système sur puce, son architecture de communication ancienne et ses limites, puis nous allons définir la nouvelle architecture de communication qui est inspirée du réseau classique « internet », ses topologies, le routage et les outils d'aide à la conception, ainsi que les problèmes de cette architecture aux différents niveaux.

I. Les Systèmes Mono puce

Les systèmes sur puce ou circuits est de plus en plus présente dans la vie quotidienne. En effet, on trouve des puces dans nos cartes bancaires, cartes vitales, les voitures, les téléphones portables, les avions, les satellites, …etc. Les circuits conçus actuellement sont d'une grande complexité et leur conception et fabrication nécessitent des techniques toujours plus sophistiquées.

La structure de communication sur puce constitue un point essentiel dans la conception. En effet, les structures de communications traditionnelles deviennent inefficaces lorsque les systèmes sur puce présentent un grand degré de parallélisme, elles ne rendent plus leurs utilisations viables. Dans ce contexte, les réseaux sur puce ont récemment vus le jour en tant que nouveau paradigme permettant de résoudre ces problèmes.

Avant d'expliciter les réseaux sur puces nous allons définir les systèmes sur puce, son évolution, et leurs contraintes de conception, en particulier les réseaux sur puce (NoC).

1. Les systèmes sur puce

Un système sur puce (System-on-Chip ou SoC en anglais) désigne un système complet embarqué sur une même puce. Les technologies actuelles permettent d'embarquer sur une même puce de silicium plusieurs millions de transistors. Cette densité importante permet de placer des systèmes de plus en plus complexes sur une unique puce, ces systèmes pouvant accueillir des unités de calcul, de la mémoire, des réseaux d'interconnexion …etc. On peut donc considérer qu'un système sur puce est l'équivalent d'un ordinateur complet. [1]

1.1 Du transistor aux systèmes sur puce

Nous présentons dans cette section un (bref) historique de la micro-électronique ainsi que les spécificités des générations de circuits conçus actuellement.

1.1.1 Les circuits intégrés

L'apparition de la micro-électronique (est récemment, de la nano-électronique) est la conséquence de l'invention du transistor par les laboratoires Bell en 1948 puis de la mise au point de la technologie d'interconnexion «Planar» en 1959 qui ont permis la conception du premier circuit intégré, constitué de six transistors en 1961. Les progrès technologiques, qui ont été réalisés depuis, permettent d'intégrer plusieurs dizaines de millions de transistors sur une surface de silicium de 10 à 300 mm2, la surface d'un transistor ayant été divisée par 10000 depuis les années cinquante.

Selon la loi de Moore [2], qui a été formulée en 1965, les progrès technologiques permettent de doubler le nombre de transistors intégrés dans une puce tous les deux ans. Cette croyance s'est révélée à peu près variée même si on peut imaginer que les industriels se sont attachés à suivre cette tendance. On s'attend aujourd'hui à ce qu'elle soit valide pendant encore une dizaine d'années avant que les progrès technologiques se heurtent à des barrières physiques dues, en particulier, à des effets quantiques.

1.1.2 Des circuits spécifiques aux systèmes sur puce

Ces immenses progrès technologiques ont permis d'intégrer sur une même puce toujours plus de fonctions et les systèmes conçus aujourd'hui embarquent fréquemment plusieurs microprocesseurs, des fonctions de traitement analogique ou numérique du signal, des interfaces multiples avec le monde extérieur, un système d'exploitation, etc. L'ensemble des traitements informatiques nécessaires pour réaliser une fonction complexe étant rassemblés au sein d'une seule et même puce, on parle de systèmes sur puce (ou SoC, pour System on Chip).

Les SoCs sont les successeurs des circuits spécialisés : les ASICs (pour Application Specic Integrated Circuit). Les caractéristiques de ces circuits sont :
- une réduction importante des coûts de production au détriment d'un coût de conception important.

- une augmentation de l'habilité des systèmes. [3]

1.2 le monoprocesseur

Le système monoprocesseur typique se compose de trois éléments principaux: la mémoire principale, l'unité centrale de traitement (CPU) et l'entrée-sortie (E / S) de sous-système.

1.2.1 Architecture monoprocesseur de base

Le CPU, la mémoire principale et les sous-systèmes (E/S) sont tous connectés à un bus commun d'interconnexion de fond de panier synchrone (SBI synchronous backplane interconnect) au travers de ce bus, tous dispositif (E/S), périphériques, mémoire et CPU peuvent être reliés directement à la SBI par un mono bus et son contrôleur ou par l'intermédiaire d'un bus de masse et son contrôleur, pour la communication chaque composants doit faire un balayage pour prendre la parole . [3]

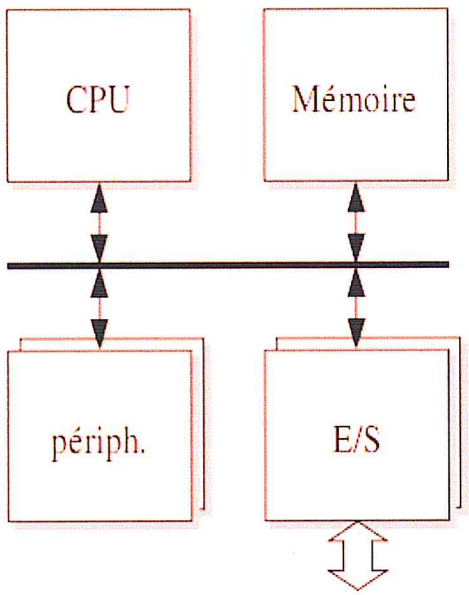


Figure 1.1: Architecture monoprocesseur

1.3 Les Multiprocesseurs sur puce

1.3.1 Les architectures multiprocesseurs de première génération

Les architectures multiprocesseurs de première génération constituent une évolution directe des architectures monoprocesseurs où des traitements de données qui au paravent on été confiés à des accélérateurs matériels dédiés sont désormais pris en charge par un ou

plusieurs processeurs spécialisés (DSP) (figure 1.2). Cette solution constitue un compromis qui exploite à la fois la puissance et la flexibilité des processeurs spécialisés. [4]

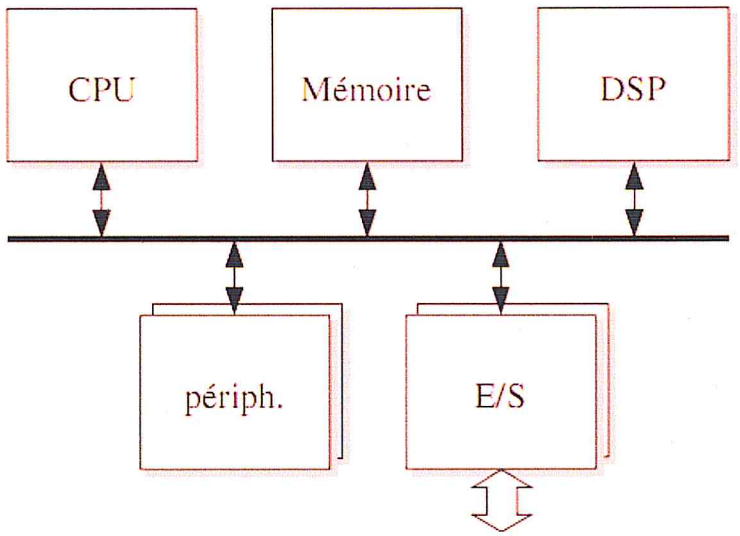


Figure 1.2: Architecture multiprocesseur de première génération

La communication est de type maître/esclave

1.3.2 Les architectures multiprocesseurs de deuxième génération

L'évolution du besoin en terme de puissance de traitement d'une part, et la percée importante de la capacité d'intégration d'une autre part, ont contribué à la mise au point d'architectures hautement parallèles intégrant un nombre important de processeurs et de composants matériels sur une même puce. [4]

2. Système multiprocesseur mono puce (MP Soc)

Un système multiprocesseur mono puce (MP Soc de l'anglais Multi-Processor System on Chip) est un SoC à fort composants logiciels. Le composant logiciel qui représente la partie programmable du système est dédié à un ou plusieurs processeurs. Ces processeurs peuvent être génériques (GPP pour general purpose processor) ou spécifiques comme les processeurs de signaux numériques (DSP pour digital signal processor) ou les processeurs dédiés aux réseaux de communication … etc.

La quête de programmabilité dans les MPSoC et motivée essentiellement par deux raisons qui ne sont pas complètement indépendantes.

La première concerne la flexibilité des systèmes, la deuxième est liée au coût de ces systèmes.

La flexibilité d'un système mesure la facilité de le faire évoluer pour l'adapter à des nouvelles exigences.

Ainsi, il est généralement connu que recompiler le logiciel embarqué après avoir changé ou adapté l'application est beaucoup plus facile que de concevoir un nouveau circuit (comme les ASIC) spécifique aux nouvelles exigences de l'application.

2.1 Méthodologies de Conception SOC et IPs

Les méthodes de conception traditionnelles sont de moins en moins adaptées aux circuits de grandes complexités. Les concepteurs sont amenés à gérer, non seulement un nombre énorme de transistors, mais aussi une incroyable complexité dans le *design*. Les possibilités architecturales pour concevoir de tels systèmes sont extrêmement nombreuses. Par conséquent, l'évaluation des différentes solutions architecturales devient un passage obligé de la conception moderne de circuits intégrés. Ces facteurs, couplés à une demande de plus en plus forte de réduction du temps de mise sur le marché avec des performances de plus en plus pointues, font qu'il n'est plus possible de développer un système en partant de zéro. La solution à ces problèmes est la réutilisation de blocs déjà conçus et validés. La notion de composants génériques réutilisables s'impose ainsi de plus en plus.

2.2 Les composants réutilisables IPs

Nous entendons par composant ou bloc réutilisable un élément d'une bibliothèque dont le concepteur dispose et qu'il peut directement inférer ou instancier sans avoir à le concevoir. L'un des problèmes majeurs concernant la réduction du temps de conception (TTM : Time To Market) réside dans la bonne ou la mauvaise réutilisation des blocs existants [5].

3. Exigences des futures structures d'interconnexion

Pour faire face aux évolutions que nous venons de présenter, les architectures de communication devront répondre aux différentes exigences que nous allons détailler :

3.1 Flexible et extensible

L'architecture de communication est conçue pour interconnecter de nombreux blocs d'IP de nature hétérogène, de plus en plus complexes intégrant des fonctions ayant des caractéristiques très diverses. Le protocole d'échange des données doit donc être

suffisamment flexible pour s'adapter à ces modules qui peuvent être développés par différentes équipes, avec différents outils, dans différents contextes et qui sont destinés à être intégrés dans le même circuit. Cette flexibilité peut être obtenue en spécifiant de manière rigoureuse des interfaces qui permettent de découpler les fonctions de traitement et les fonctions de communication.

On utilise le terme d'interface-based design [5]. Une architecture de communication flexible permet aussi de gagner en productivité. En effet, elle pourra être réutilisée d'un circuit à l'autre et supporter les besoins de nombreuses applications. La notion d'extensibilité est également un élément important.

Il s'agit de la possibilité d'augmenter le nombre de blocs interconnectés tout en maintenant le même niveau de performance dans les communications. [6]

3.2 Simplicité

Minimiser le nombre de ressources nécessaires à la réalisation d'un réseau sur puce garantit le plus souvent de meilleures performances en termes de latence et de bande passante.

4. Les solutions d'interconnexion actuelles et leurs limites

Traditionnellement, les structures de communication utilises pour les systèmes sur puce sont toujours basées soit sur des connexions point-à-point entre les éléments communicants, soit sur des bus partagés [7].

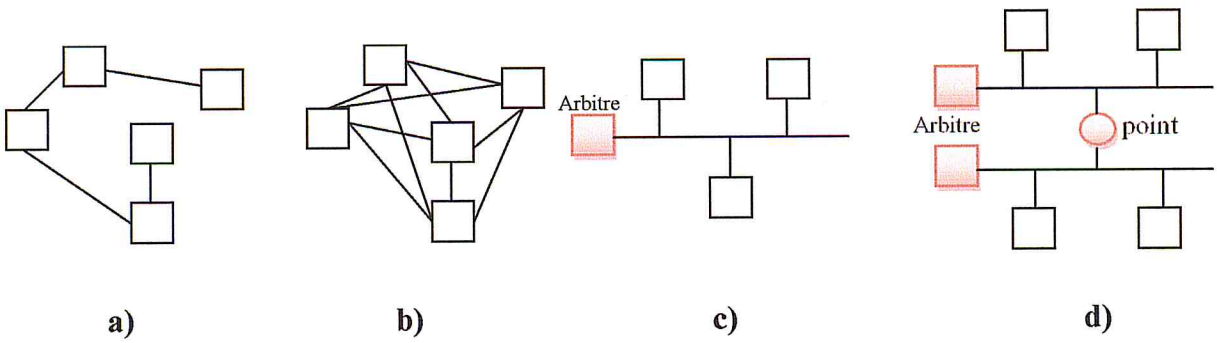


Figure 1.3 — Structures de communication traditionnelles

4.1 La connexion point à point

Dans une connexion point à point (figure 1.3(a)), des liens dédiés sont établis entre chaque couple d'éléments communicants. Cette solution offre de nombreux avantages en termes de performances car les contraintes en latences et en débits peuvent être parfaitement

respectées. De plus il ne peut y avoir de congestion et aucun arbitrage n'est nécessaire pour l'accès au médium de communication. [8]

Cependant, la connexion point à point ne fournit aucune tolérance aux fautes/pannes pouvant survenir sur un lien. D'autre part ce mode de connexion ne présente aucune flexibilité car elle n'est adapté qu'à une architecture donnée et donc difficilement réutilisable, enfin une grande partie de la bande passante est typiquement perdue car les liens ne sont que très peu utilisés [9]

4.2 La connexion complète

Si tous les éléments communicants sont reliés entre eux par des connexions point à point, la connexion est dite complète [10]. (Figure 1.3(b)). Dans ce cas, la structure tolère mieux les pannes mais les liens sont encore plus sous-exploités. En outre, le coût en termes d'interconnexions est le plus élevé, ce qui limite énormément l'extensibilité car il n'est pas concevable d'utiliser ce type d'architectures lorsque le nombre d'unités à interconnecter est très grand.

1.1 Le bus partagé

Dans les systèmes sur puce complexes, le moyen de communication actuellement le plus répandu dans l'industrie est le bus partagé (figure 1.3(c)). En effet, ils sont beaucoup plus flexibles et mieux réutilisables, comparés aux liaisons point à point.

En revanche l'inconvénient principal des bus provient du fait qu'ils n'autorisent qu'une seule connexion à la fois. La bande passante allouée à chaque élément diminue donc avec le nombre d'éléments communicants. De plus, les bus requièrent un mécanisme supplémentaire d'arbitrage permettant de traiter les demandes d'accès concurrentes. du point de vue physique, les longueurs de fils dans les bus sont souvent plus grandes, ce qui aboutit à des a consommations d'énergie plus élevées et des délais de transmission pouvant passer la période d'horloge du système [11].

1.2 Le bus hiérarchique

Dans le but d'augmenter l'extensibilité et les performances des bus partagés, il existe une structure appelée bus hiérarchique [12] (figure 1.3(d)) qui permet de connecter plusieurs

bus partagés à l'aide de ponts (bridges). Ces derniers peuvent servir de convertisseur de protocoles si les bus n'utilisent pas le même protocole.

Cependant, les connexions entre les bus ne sont optimales que lorsqu'il n'y a pratiquement pas des communications sur les deux bus. En outre le problème d'extensibilité est réduit mais pas résolu.

1.3 Le crossbar

Une alternative offrant un compromis intéressant entre les topologies bus et celles des réseaux est le crossbar. Dans ce cas, tous les blocs fonctionnels de l'application sont reliés les uns aux autres par l'intermédiaire du crossbar (figure 1.4). Celui-ci a l'avantage de permettre des communications parallèles (contrairement au bus) et d'offrir une grande bande passante pour chaque communication car celles-ci ne sont plus partagées avec les autres communications comme c'est le cas dans la topologie bus standard ou hiérarchique.

Cependant, la complexité de câblage d'une telle architecture croit en fonction du nombre d'IP qui composent l'application. Celle-ci augmente avec le carré du nombre d'éléments communicants soit une complexité de O (n2), ce qui devient vite exorbitant.

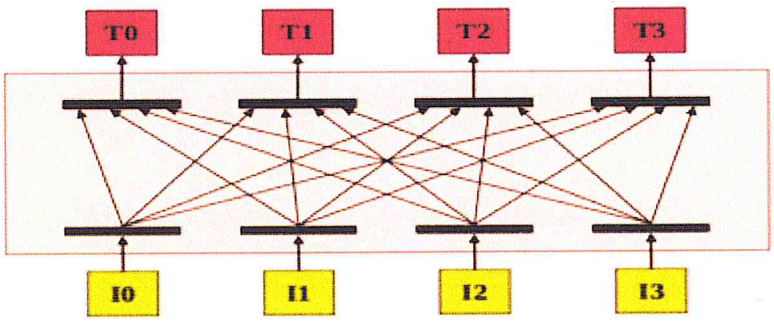


Figure. 1.4 – Topologie d'un crossbar 4 vers 4

1.4 réseaux sur puce

Un réseau sur puce (NoC pour Network on Chip) est comme un réseau de machine constitué d'un ensemble d'éléments de routage (commutateurs ou Switch en anglais) interconnectés par des liens (ensemble de fils).

La tâche première d'un NoC est d'échanger des informations d'un point vers un autre point en offrant des meilleures performances que le bus, et ce non seulement au niveau de la bande passante, mais aussi du point de vue évolutif, tolérance aux pannes, etc.

Les travaux sur les NoCs restent du domaine de la recherche académique. Il n'existe pas de solution industrielle car les approches ont été jugées manquantes de maturité [13].

1.5 Tableaux de Comparaison entre les Types d'Interconnexion

Tableau 1.1 : Comparaison entre les différentes structures d'interconnexion dans les SoCs [1]

Connexion	Parallélisme	Consommation	Scalabilité	Réutilisation
Point à point	++	+	--	--
Bus partagé	--	--	--	++
Bus hiérarchique	+	-	-	++
NoC	++	++	++	++

++ : Très bon

\+ : Bon

-- : Très mauvais

\- : Mauvais

Nous avons présenté, dans cette première partie, le système sur puce et définie ses topologies de communication est nous concluions que la topologie NoC offre de meilleures performances que le bus, non seulement au niveau de la bande passante mais aussi du point de vue évolutif, tolérance au pannes et …etc.

C'est pourquoi, dans la partie suivant, nous allons préciser architectures de communication réseau sur puce.

II. Les Réseaux sur puce (NoCs)

Les limitations des bus décrites dans la partie précédente en terme de flexibilité, extensibilité et Simplicité car les applications aujourd'hui demande plus en plus l'augmentation du nombre de composants sur les puces. Il est donc nécessaire pour les SoCs à venir de concevoir des interconnexions extensibles, très performantes, peut consommatrices.

Dans cette partie, nous intéresserons aux réseaux sur puce, ses composants,

ses caractéristiques et quelques architectures académiques puis les phases de conception.

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

Depuis une dizaine d'années, les premiers concepts et prototypes de réseau de communication sur puce appelé Network-on-Chip (NoC), sont apparus.

Selon la définition proposée par Dally [11], un réseau d'interconnexion correspond à un système permettant de transporter des données entre des terminaux. La figure 1.5 illustre un exemple avec trois terminaux T0, T1 et T2.

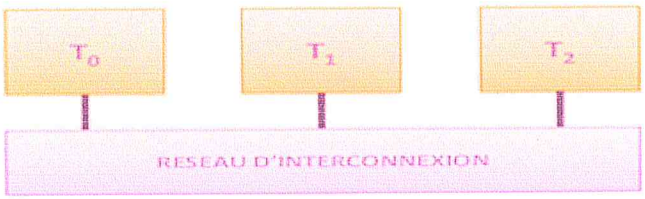


Figure 1.5: Réseau d'interconnexion du point de vue fonctionnel avec trois terminaux

Les terminaux se communiquent en se transmettant des messages (données) à travers le réseau. Un terminal correspond à un point source ou un point de destination. Le réseau peut effectuer plusieurs connexions simultanées entre les terminaux autorisant ainsi plusieurs communications en parallèle et les modifier à tout instant. Un réseau est défini comme un système car il est composé de différentes ressources :interface, canal de communication et d'un élément de commutation de données appelé routeur qui s'organisent pour transmettre les messages entre les terminaux.

1.1 Structure d'un message

Chaque message, illustré par la figure 1.6, est fractionné en plusieurs paquets de données [11]. Le paquet est l'unité d'information du réseau de communication, ils sont en général composés d'un entête (header) contenant des informations de contrôle et de la donnée utile à transporter. Afin de pouvoir être transmis dans le réseau, ces paquets sont divisés en paquets élémentaires appelés

flits (contraction de flow control digits). Ces flits peuvent être encore subdivisés en phits (contraction de physical units) correspondant à une unité de donnée qui peut être transférée physiquement à travers un canal de données entre deux routeurs en un cycle d'horloge. Les performances (i.e. latence, bande passante, consommation) d'un réseau sur puce dépendent dans un premier temps du choix de la granularité de chaque unité de données (message, flits, phits).

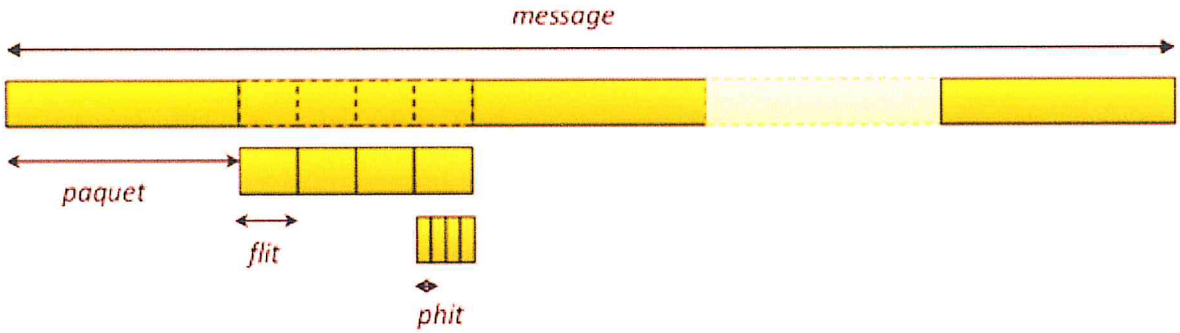


Figure 1.6: Structure d'un message dans un réseau NoC [11]

1.2 Les NI (Network Interface) ou interface réseau

Elles réalisent l'interface entre le protocole du NoC et celui des blocs IP qui sont connectés au routeur, leurs rôle est de séparer le traitement (effectué dans les IP) des communications (gérées par le réseau). Un adaptateur réseau peut être composé de deux parties, l'une prenant en charge l'interface réseau en elle même, l'autre réalisant l'adaptation de protocole aussi appelée "wrapper".

1.3 Les routeurs

Ils acheminent les paquets de données dans le réseau en fonction du protocole choisi en respectant une stratégie de routage qui constitue l'arbitrage du routeur.

1.4 Les liens

Ils relient les routeurs entre eux ou les routeurs aux NI (Network interface). Ils offrent la bande passante pour les communications entre la source et la destination, celui-ci peut posséder plusieurs canaux virtuels et peut être monodirectionnel ou bidirectionnel. Dans la plupart des NoC que nous verrons dans la section 1.5, les liens

sont de type bidirectionnel.

1.5 Unité de traitement

Composant fait une fonction spécifique par exemple DSP, RAM, ASIC...etc.

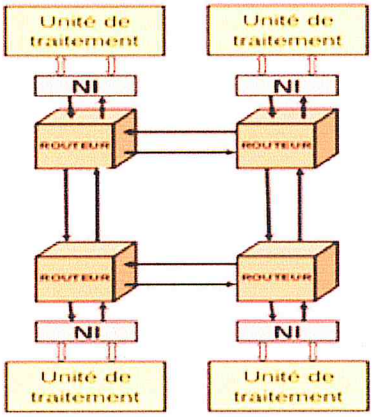


Figure. 1.7 – Topologie Réseau

2. Les caractéristiques des réseaux sur puce

On peut trouver dans la littérature différentes topologies de réseau apportant des avantages et les inconvénients maintenant nous allons donc voir les principales structures que l'on peut trouver dans les NoC.

2.1 quelques Topologie du réseau sur puce

La topologie d'un réseau sur puce spécifie l'organisation physique du réseau et définit la disposition structurelle de ses éléments. Plusieurs topologies sont utilisées dans la littérature. Les plus répandues sont la 2D maillée, la topologie en tore et la topologie en anneau.

- **Topologie 2D maille**

La topologie la plus souvent employée est celle de structure 2D régulière représentée sur la figure 1.8. Elle est simple de mise en œuvre et facilement implantable sur une technologie silicium.

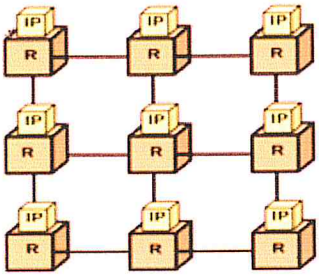


Figure. 1.8 – Topologie 2D

❖ **Topologie de torus**

Le torus (figure 1.9) est une topologie dérivée de la structure 2D qui possède la particularité d'un reliment des bords extérieurs sur eux même, offrant une bande passante légèrement supérieure.

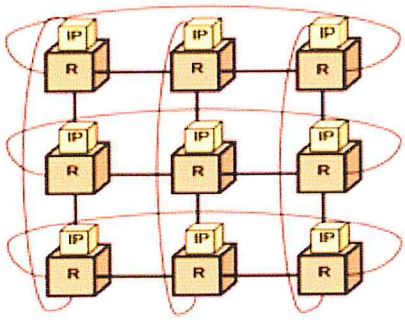


Figure. 1.9 – Topologie 2D Torus

❖ **La topologie 3D**

La topologie 3D (figure 1.9) quand à elle offre une plus grande bande passante que les deux autres mais elle souffre d'un inconvénient majeur qui est la difficulté de son implantation sur silicium (routage complexe). [14]

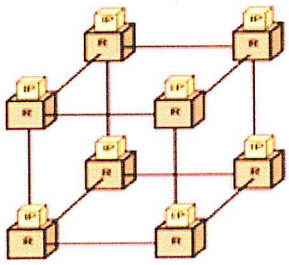


Figure. 1.10 – Topologie 3 D

❖ La topologie arbre

La topologie en arbre élargi représentée sur la figure 1.11 (dite "Fat-tree") a pour intérêt d'être scalable et d'offrir une latence pouvant être plus faible que la topologie en grille 2D. Cette architecture nécessite un bon placement des IP sur le réseau car les routeurs intermédiaires deviennent rapidement des goulots d'étranglement lorsque plusieurs feuilles d'une même branche veulent communiquer avec des feuilles d'autres branches. C'est précisément dans ce cas que cette topologie trouve ses limites. De nombreux travaux sur ce type d'interconnexion sont menés par le LIP6 (Université Pierre et Marie Curie).

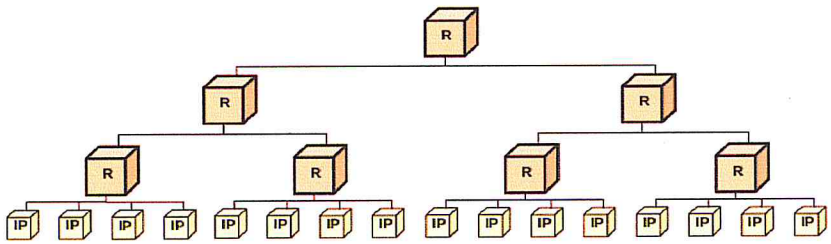


Figure. 1.11 – Topologie arbre

❖ La topologie en anneau

La topologie en anneau, représentée sur la figure 1.12, est facile à mettre en œuvre au niveau de l'algorithme de routage et de l'implémentation physique car les routeurs sont reliés uniquement à leurs deux voisins par des liens unidirectionnels, mais cette topologie souffre d'un inconvénient majeur, car pour un réseau de grande taille, le message doit traverser un nombre important de routeurs, engendrant inévitablement une limite de bande passante.

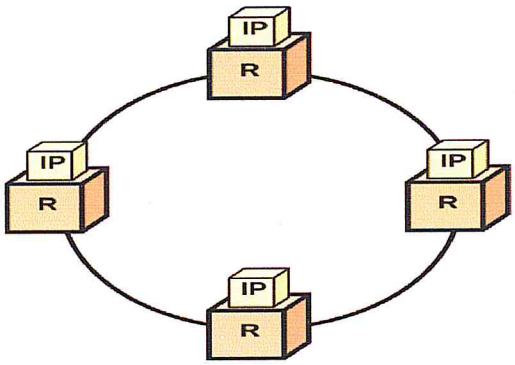


Figure. 1.12 – Topologie anneau

❖ **La topologie octogone**

Une alternative proposée à la limitation de la topologie en anneau, représentée sur la figure 1.13. [6], elle permet de réduire la latence en offrant la garantie de ne traverser au maximum que deux routeurs pour n'importe quelle communication au sein du réseau.

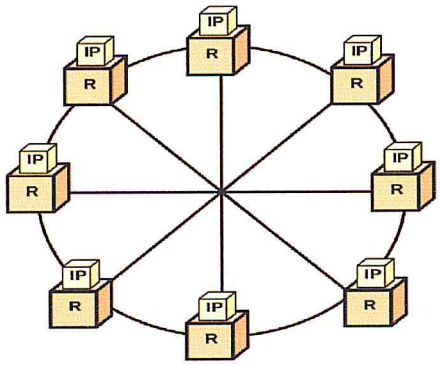


Figure. 1.13 – Topologie octogone

4.3 Les modes de commutations

Il existe deux modes de commutation, commutation des circuits et commutation des paquets.

❖ **La commutation des circuits**

Dans ce cas une connexion est établie entre la source de données et sa destination, cela signifie que des éléments du réseau (liens, nœuds) sont alloués pour un transfert de données et ne peuvent pas être utilisés par d'autres nœuds durant le temps de la connexion. Il existe un lien physique entre les deux nœuds, des informations de contrôle (pour le début et la fin de connexion par exemple) sont séparées par des données. Ce mécanisme est bien adapté lorsque l'on veut effectuer des transferts de données importants puisqu'il faut rentabiliser le temps passé à négocier la connexion. Physiquement les nœuds sont simples puisqu'ils établissent seulement un lien entre un port d'entrée et un port de sortie, il n'y a pas de blocage. [15]

❖ La commutation de paquet

Avec ce mécanisme, la structure d'échange au niveau du réseau est le paquet. Un paquet peut contenir une fraction, un ou plusieurs messages. Les paquets sont transmis sans qu'il y ait de connexion établie préalablement, l'avantage est de pouvoir accéder au réseau plus rapidement et de partager les ressources. Le réseau est localement commuté entre deux nœuds mais pas entre deux ressources comme dans la commutation de circuit, les paquets doivent contenir des informations sur leurs destinations : les informations de contrôle sont envoyées en même temps que les données, les nœuds sont souvent plus complexes ils ont parfois à modifier le contenu des paquets pour mettre à jour des informations de routage par exemple, la latence est plus grande. Il faut introduire des systèmes de gestion des blocages et des priorités, les paquets peuvent arriver de façon désordonnée et il y a moins de garantie temporelle sur le transfert. Dans le cas de la commutation par paquet, il faut définir le mode de commutation c'est à dire la façon dont les paquets vont transiter d'un nœud du réseau au suivant. Les trois principaux modes sont [15] :

- **Store-and-forward**

Ce mode de commutation consiste à faire transiter le paquet dans son ensemble d'un nœud à l'autre (Figure 1.14) cela implique que chaque nœud doit être en mesure de stocker la totalité d'un paquet. Il existe donc une taille de paquet maximale. De plus la latence des échanges est importante puisqu'il faut multiplier le temps de transfert d'un paquet entre deux nœuds par le nombre de nœuds du réseau traversés par le paquet de sa source à sa destination. [15]

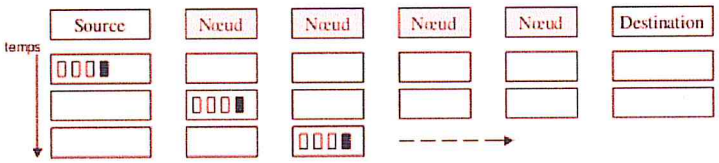


Figure 1.14 : Commutation Store and Forward

- **Virtual cut through**

Avec ce mode un nœud peut commencer à envoyer un paquet si le nœud suivant lui garantit qu'il peut stocker le paquet dans sa totalité, dans le cas contraire le nœud doit pouvoir garder le paquet. La capacité de mémorisation du nœud est donc la même que pour le mode Store-and-forward mais la latence est diminuée puisqu'il n'est plus nécessaire d'attendre la réception complète du paquet pour qu'il passe d'un nœud à l'autre. [15]

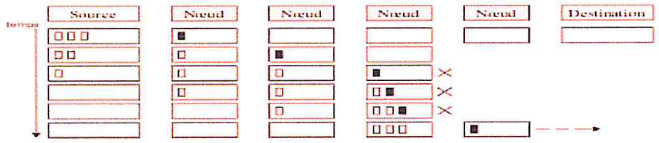


Figure 1.15 : Commutation Virtual Cut Trough

- **Wormhole**

Ce mode a pour but de réduire la taille des mémoires dans les nœuds sans limiter la taille des paquets. Les paquets sont découpés en éléments de taille fixe appelés flits (flow control unit). Les flits passent de nœud en nœud dès qu'il y a de la place pour un flit et pas nécessairement pour un paquet complet (Figure 1.16). Le flit de tête15 contient des informations de contrôle, en particulier sur la destination du paquet. Tous les flits qui suivent doivent emprunter le même chemin que le flit d'entête. Un même paquet peut donc être réparti sur plusieurs nœuds du réseau. La commutation Wormhole permet donc de réduire la latence. [15]

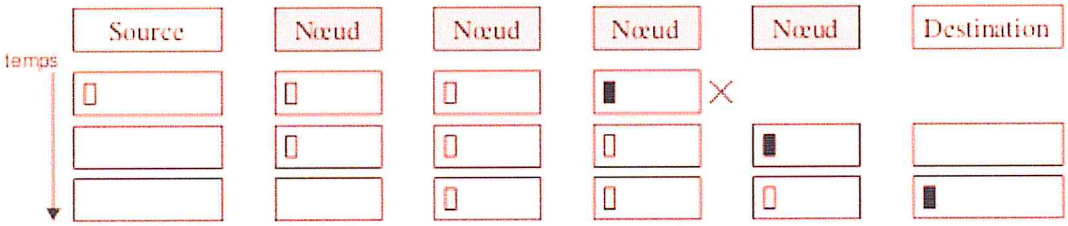


Figure 1.16 : Commutation Wormhole

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

Tableau 1.2 : Comparatif de différents modes de commutation (D'après [15])

2.3 L'algorithme de routage

Détermine quelle série de nœuds les données vont emprunter entre la source émettrice à la source réceptrice, donc tout les algorithmes sont censés de trouver le meilleur compromis entre une utilisation optimale des canaux de communication du réseau et être simple ne nécessité pas de matérielle trop importante.

Algorithme de routage XY :

L'algorithme XY est un algorithme déterministe et qui garantit une solution pour toute situation de blocage (deadlock) **[16]**. Les flits sont routés d'abord dans la direction X ensuite dans la direction Y. Si un saut est utilisé dans le NoC par un autre paquet, le flit reste bloqué dans le routeur (buffers) jusqu'à ce que le chemin soit libéré. La Figure 1.17 illustre le fonctionnement de cette technique.

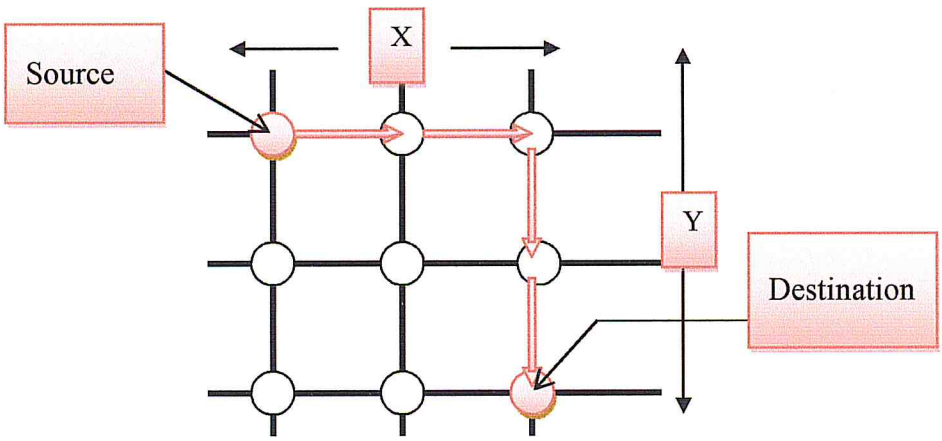


Figure 1.17 : Routage ordonné X-Y [16]

2.4 Contrôle de Flux

❖ mécanisme de communication HANDSHAKING

L'interaction entre les routeurs est en fait très simple, dans ce protocole de communication le routeur source envoie les données est attend jusqu'à la réception de l'accusé du routeur destinataire pour reprendre la transmission dans le canal.

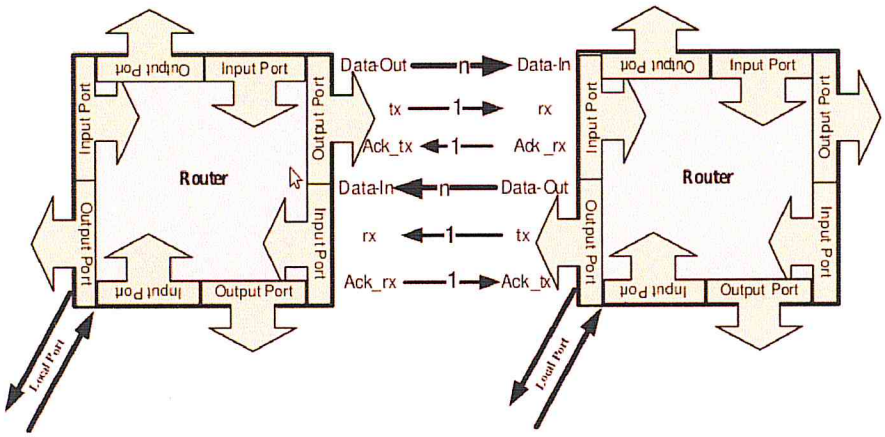


Figure1.18: Communication signals in handshake protocol between two NoC Routeurs [5]

❖ Credit Based

C'est un mécanisme de contrôle de flux où le nœud a un compteur pour suivre le nombre de slots disponibles dans la file d'attente. L'état du compteur est décrémenté à chaque fois qu'un flit est transmis et incrémenté à chaque fois qu'un signal de crédit est envoyé, la complexité d'échange dans le crédit implique une augmentation de la consommation d'énergie, ce mécanisme est simple à implémente et améliore la bande passant et le débit car le transfert des données ne nécessité qu'un seul cycle d'horloge.

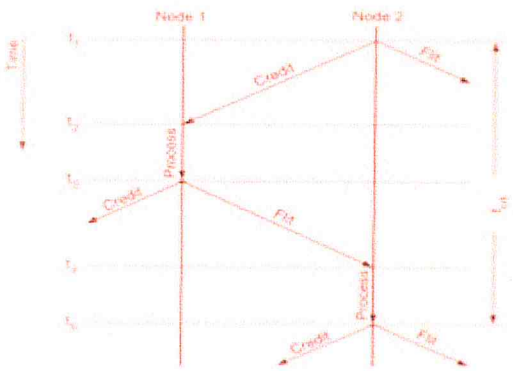


Figure 1.19: crédit based

3. Les architectures académiques NoCs

1.1 SPIN

Le réseau SPIN est développé par le laboratoire LIP6 de l'Université Pierre et Marie Curie de Paris.

C'est historiquement une des premières propositions concrètes de NoC par commutation de paquet. Ce réseau s'appui sur une topologie en arbre élargi qui offre une latence limité grâce à un faible diamètre. Cette topologie est par contre moins flexible à intégrer et peut extensible. La commutation est de type Wormhole avec un algorithme de routage adaptatif et distribué ce qui impose un réordonnancées des paquets à la réception.[15]

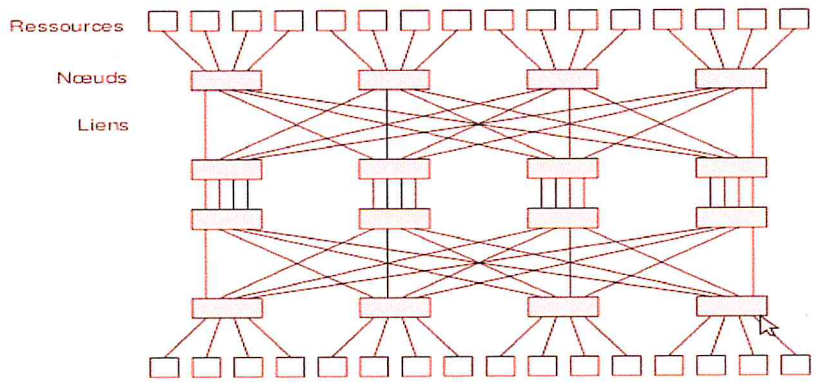


Figure 1.20 – Topologie du réseau SPIN à 32 ports

1.1 QNOC

L'architecture QNOC est proposée par l'institut technologique du Technion (Is- raël) [17]. La topologie est maillée à deux dimensions. Les ressources de traitement sont hétérogènes. L'algorithme du routage est de type XY ou YX en fonction des positions de l'émetteur et du récepteur de manière à ce que le chemin soit le même lors d'échanges bidirectionnels. Un mécanisme de crédit permet le contrôle le flux de messages.

1.2 ANOC

L'architecture de communication ANOC est composé de nœuds, les liens entre les nœuds et les ressources de calcul Figure. 1.7. Les nœuds asynchrones NoC sont la base éléments de commutation du réseau: ils sont responsables de la gestion du protocole Wormhole.

1.3 Hermès

la Hermès NoC suppose que chaque commutateur comporte un ensemble de ports bidirectionnels associés à d'autres commutateurs et à un noyau IP, chaque commutateur dispose d'une nombre différent de ports, en fonction de sa position par rapport aux limites de la réseau la communication de type Wormhole et l'arbitrage fait par algorithme de « Round Robin »

II. Limites des Réseaux sur Puce

Les réseaux sur puce sont bien d'un côte mais malheureusement il y a des limites pour les réseaux sur puce a cause des :

4.1 Cohérence des caches

Si on partage des données en plusieurs copies et on les stocks dans la mémoire caché des IPs, lorsque les IPs change ils sont copies alors il y a un problème de cohérente des données, il existe plusieurs solutions :

- La solution propose par **LOGHI Mirko** basé sur l'espionnage mais il y a une augmentation de consommation d'énergie.
- La deuxième solution ne stocke pas les données partagées dans la cache IPs mais cela implique une grande quantité d'informations partagées et la consommation d'énergie reste importante.
- La troisième solution la plus employée basée sur l'utilisation de mémorisation centralise des données cohérentes pour la réduction d'énergie et le temps d'exécution.

4.2 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 [12].

4.3 Ordre des communications

La commutations dans les réseaux ne conservée pas l'ordre d'émission des paquets, donc il faut des techniques de ré-ordonnancement dans la réception. Cette opération est autant plus importante lorsqu'il s'agit des requêtes de lecture/écriture qui doivent être exécutées selon leurs ordres d'envoi. [12]

4.4 Conception

La conception d'un architecture de réseau sur puce éffication dépende de satisfaction des contraintes de l'application, réduire temps de mise sur le marché et automatise les taches de conception.

Le réseau sur puce inspiré du réseau traductionnel est composé a des blocs de calcule, plusieurs routeurs communiquent via des liens .L'architecture 2D maille la plus employée dans les réseaux sur puce.

La commutation par circuit garantir une latence et début précis par contre la commutation par paquets ouvre meilleur utilisation des ressources de réseaux.

Le contrôle de flux fait soit synchronise par mécanisme handshake ou par Based Cridet qui permis d' a synchronisation des données envoie

La partie suivante traiter les points de conception et des outils qui simplifié la conception.

III. Conception des Réseaux sur puce

Les réseaux sur puce permis d'interconnecter plusieurs modules dans la puce de silicium et Représenter une solution pour les problèmes actuelles, mais la conception resté complexe et délicate donc il est nécessaire d'automatisé les taches de conception par des méthodes et des outilles.

Ce partie traiter la phase de conception de réseau sur puce et les outils qui permit de résoudre les problèmes de conception de réseau sur puce.

1. Les 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. [15]
- 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. [15]
- 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. [15]

2. Approches de Conception des Réseaux sur puce

Les réseaux sur puce sont classés selon le modèle de trafic pour lequel ils sont conçus [17].

2.1.réseau sur puce pour Multi Processors

Dans un MPSoCs à usage général, le modèle de trafic est complètement imprévisible jusqu'à ce moment de l'exécution.la meilleure solution consiste à planifier un modèle

uniforme où chaque module communique avec les autres selon une probabilité de communication P [17].

2.2. NoCs pour SoCs (Systems On Chip)

La spécification NoCs désigné pour SoCs il est nécessaire de connu le modèle d'utilisation avant de l'exécution de l'application, l'utilisation doit être spécifique et leurs fonctionnalités peuvent être simulées au moment de la conception.

3. Phases de Conception d'un Réseau sur Puce pour Systèmes Sur Puce

Conception d'un réseau sur puce efficace satisfaisante les critères de l'application est un processus complexe et portant plusieurs niveau abstracts de niveau modélisation jusqu'à l'implémentation physique.

La conception d'un NoC comporte les phases suivantes [18] :

- Modélisation du trafic de communication
- Détermination de la topologie NoC pour l'application
- Ordonnancement des taches.
- Place les nœuds de l'application dans la topologie.
- Alloué des schémas de routage et réservation des ressources.
- Vérification de performance de l'application.
- Développement des modèles de synthèse et simulation.

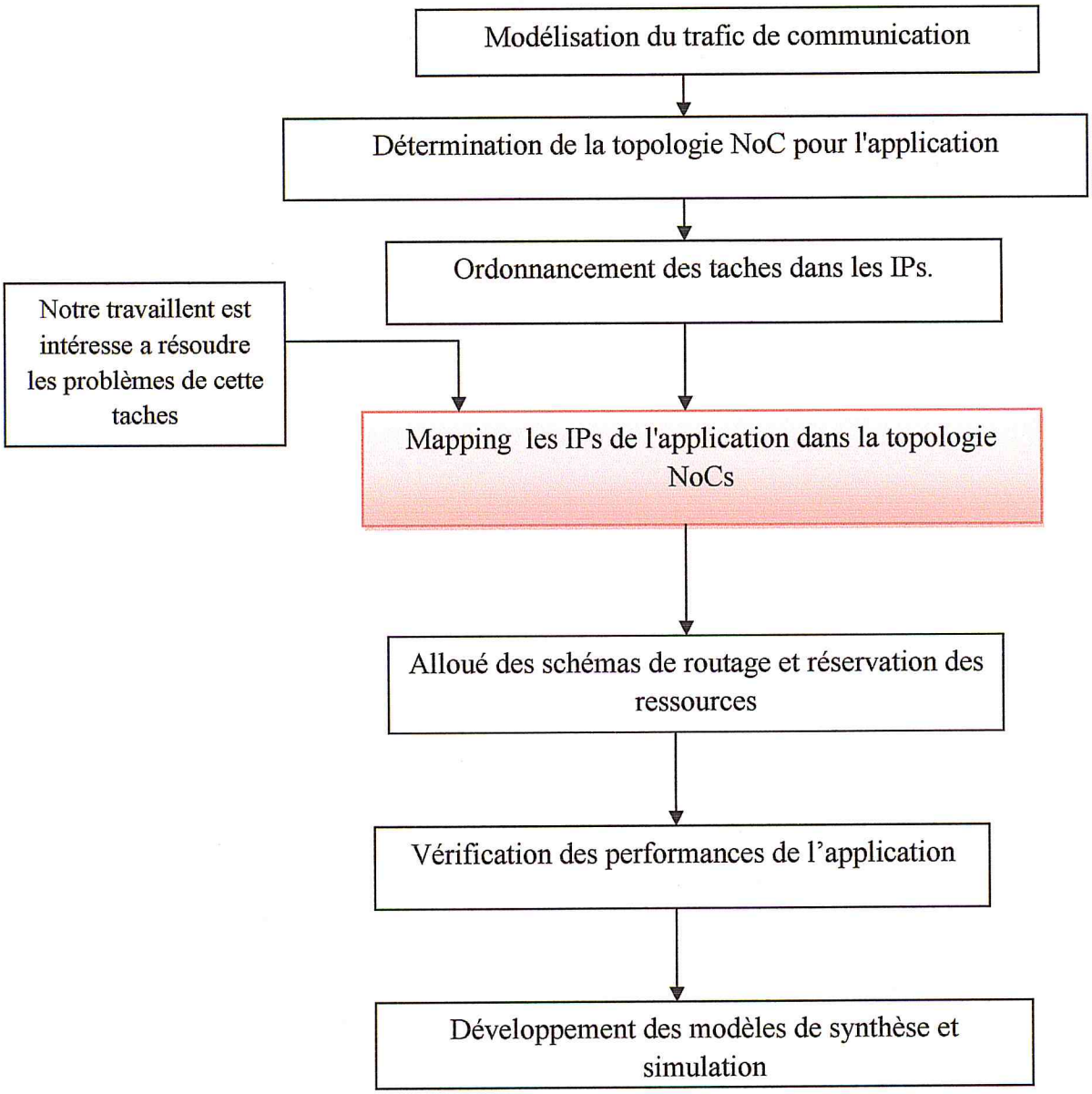


Figure 1.21: les phases de conception d'un réseau sur puce [16]

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

a. Méthodologie de conception SNMAP

SUNMAP analyse statique la quantité de transfert de données entre les nœuds, il base sur trois phases :

Dans la première phase on choisit une fonction de routage, on suite un mappage sur diverses topologies du réseau. Pour chaque affectation on vérifie les contraintes de la zone et la bande passante est évaluée afin de produire une mapping réalisable.

Dans la deuxième phase les différentes topologies (avec mapping produites à partir de la première phase) sont évalués pour plusieurs objectifs de conception et choisi la meilleure topologie.

Dans la troisième phase l'outil génère description SystemC des composants du réseau.

b. NoCExplorer

NOCExplore est un outil de réseau sur puce, il est basé sur SystemC. L'outil d'exploration NoCexplorer fournit un environnement intuitif et robuste pour saisir les besoins de flux de données des blocs IP et permet au concepteur d'analyser rapidement les différentes options de topologie NoC pour optimiser les performances et la mise en œuvre. [19]

c. MAIA

Environnement permet de génère un fichier descriptive a partir des paramètres de l'utilisateur pour réaliser la simulation du trafic et l'évaluation de performance du réseau. [7]

d. XpipesCompiler

L'outil XPipesCompiler **[JAL 04]** s'intègre aussi dans le flot de conception du réseau sur puce QNoC. 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. [20]

5. La Phase de Mapping

La phase de mapping consisté a placé les IPs dans la structure .la complexité de ce processus dépende de nombre de module IP et tuiles.

5.1 Types de Mapping

Mapping peut être fait avant l'application ou moment l'exécution de l'application :

i. mapping statique

Toutes les IPs sont affectées aux tuiles de l'architecture avant que l'application ne soit Exécutée [21].

ii. Mapping dynamique

L'affectation des IPs fait durant l'application, une tache peut être ajouté ou supprimes Durant l'application. Le but est de reconfigurer la plateforme de communication tout en Minimisant le temps de replacement des tâches [21].

5.2 Résolution du problème de Mapping

Le problème consister à placer m° IPs sur n° tuiles s'avère être un problème NP-difficile [22] , il n! solutions si n° devient c'est impossible d'essaiyer toutes les solutions existentes.

Alors l'algorithme exacte prend trop de temps.donc c'est obléige d'utiliser des méthodes approchées pour résoudre ce problème. [21]

5.2.1 Méthodes exactes

Garantir la solution optimale mais le temps d'exécution augument de manière exponentielle en fonction de la taille du problème. [21]

5.2.2 Méthodes approchées

Consister a trouver (optimale ou proche de l'optimale), l'avantage est réduire le temps lors qu'a la taille du problème très grand.

- Méthodes constructives telles que les algorithmes gloutons et la méthode Pilote.
- Méta-heuristiques pour l'optimisation recuit simulé, recherche tabou.
- Méthodes évolutives telles que les algorithmes génétiques et les fourmis artificielles.
- Réseaux de neurones (Modèle de HopfieldTank, machine de Boltzmann, réseau auto adaptatif, réseau élastique).

- Heuristiques Bayésiennes (optimisation globale, optimisation discrète).
- Superposition (perturbation des données, perturbation des paramètres d'une heuristique).

Conclusion

Dans ce chapitre, on a expliquer les notions suivante :système sur puce ,caractéristique des réseaux sur puce ,quelque architecture académique ,les phases de conception et les outils d'aide à la conception des réseaux sur puce.

Nous intéressons dans le chapitre suivant à la phase de mapping qui est le point essentiel de notre travail et l'un des problèmes de conception des réseaux sur puce. En effet, le chapitre suivant sera consacré à la présentation de quelques méta-heuristiques et son utilisation pour résoudre le problème de mapping.

Chapitre 2 : Mapping dans un réseau sur puce

Introduction

Les réseaux sur puce permettent d'interconnecter plusieurs modules sur une puce et représentent une solution pour les limites des architectures actuelles, mais la conception reste complexe et délicate donc il est nécessaire d'automatisé les tâches de conception par des méthodes et des outils.

L'un des problèmes est comment placer l'ensemble des taches d'application dans des tuiles matérielle (ensemble des unîtes de calculer) de telle sorte que le coût de communication doit être minimisé. Ça ressemble à un problème d'affectation quadratique donc on ne peut pas le résoudre par une méthode exacte car son temps d'exécution est très lent, donc les méthodes approchées comme les méta-heuristiques sont utilisées.

Les méta-heuristiques donnent une solution de ce problème avec temps d'exécution réalisable et des contraintes sur la solution trouvée.

Ce chapitre est consacré à définir quelques méta-heuristiques qui peuvent résoudre le problème de mapping.

I. Les métas heuristiques

Les méta-heuristiques forment un ensemble de méthodes utilisées en recherche opérationnelle pour résoudre des problèmes d'optimisation difficiles.

A ces problèmes de minimisation, les méta-heuristiques permettent dans un temps de calcul raisonnables de trouver des solutions peut-être pas toujours optimales en tout cas très proches de l'optimum.

Nous proposons de fournir un panorama de quelque méta-heuristique.

1. L'optimisation difficile

1.1 définition

L'optimisation difficile consiste à trouver la meilleure solution entre un nombre fini de choix. Autrement dit, à minimiser une fonction, avec ou sans contraintes, sur un ensemble fini de possibilités. Quand le nombre de solution possibles devient exponentiel par

rapport à la taille du problème, le temps de calcul devient rapidement critique.

Ce temps de calcul devient une problématique quand ne connaît pas algorithme exact de complexités polynomial, c'est-à-dire dont le temps de calcul soit proportionnel à $\mathbf{m^n}$ où n Désigne le nombre de paramètres inconnus du problème et n est une constante entière.

1.2Heuristique et méta-heuristiques

Une méthode heuristique (du verbe grec heuriskein, qui signifie « trouver ») permet de guider par un processus dans la recherche des solutions optimales.

Feignebaum et Feldman (1963) définissent une heuristique comme une règle d'estimation, une stratégie, une astuce, une simplification, ou toute autre sorte de système qui limite drastiquement la recherche des solutions dans l'espace des configurations possibles. Newell, Shaw et Simon (1957) précisent qu'un processus heuristique peut résoudre un problème donné.

Dans la pratique, certaines heuristiques sont connues et ciblées sur un problème particulier.

La méta-heuristique, elle se place à un niveau plus général encore et intervient dans toutes les situations où l'ingénieur ne connaît pas d'heuristique efficace pour résoudre un problème donné.

En 1996, I.H.Osman et G.Laporte définissaient la méta-heuristique comme « un processus itératif qui subordonne et qui guide une heuristique, en combinant intelligemment plusieurs concepts pour explorer et exploiter tout l'espace de recherche. Des stratégies d'apprentissage sont utilisées pour structurer l'information afin de trouver efficacement des solutions optimales, ou presque Optimales ».

En 2006, le réseau Meta-heuristiques (metaheuristics.org) définit les méta-heuristiques comme « un ensemble de concepts pouvant être appliqués à une grande nombre des problèmes. On peut voir la méta-heuristique comme une « boîte d'outils » algorithmique, utilisable pour résoudre différents problèmes d'optimisation et ne nécessitant que peu de modifications pour qu'elle puisse s'adapter à un problème particulier .

Elle a donc pour objectif de pouvoir être programmée et testée rapidement sur un problème.

Comme l'heuristique, la méta-heuristique n'offre généralement pas de garantie d'optimalité.

Elle fait parfois usage de l'expérience accumulée durant la recherche de l'optimum, pour mieux guider la suite du processus de recherche.

2. Classification des métas heuristiques

Les méta-heuristiques distinguées en deux approches :

- les approches constructives « basé sur une solution»

Ces algorithmes partent d'une solution initiale (obtenue de façon exacte, ou par tirage aléatoire) et s'en éloignent progressivement pour réaliser une trajectoire.

Dans cette catégorie, se rangent :

- la méthode de descente
- le recuit simulé
- la méthode Tabou

Le terme de recherche locale est de plus en plus utilisé pour regroupe ces méthodes.

- les approches évolutives « base sur une population »

Elles consistent à travailler avec un ensemble de solutions simultanément, que l'on fait évoluer progressivement. L'utilisation de plusieurs solutions simultanément permet naturellement d'améliorer l'exploration de l'espace des solutions. Dans cette seconde catégorie on trouve :

- les algorithmes génétiques
- les algorithmes des colonies des fourmis
- l'optimisation par essaim particulaire
- les algorithmes à estimation de distribution

Notons que ces méta-heuristiques évolutionnaires seront probablement plus gourmandes en calculs, mais on peut supposer aussi qu'elles se prêteront bien à leur parallélisassions.

Certains algorithmes peuvent se ranger dans les deux catégories à la fois, comme la méthode GRASP qui construit un ensemble de solutions et améliore ensuite avec une recherche locale. La plupart des méthodes développées ces dernières années sont d'ailleurs souvent à chevé sur ces deux approches.

Une autre manière, plus intuitive, de classifier les méta-heuristiques consiste à séparer celles qui sont inspirées d'un phénomène naturel, de celles qui ne le sont pas.

Les algorithmes génétiques ou les algorithmes par colonies de fourmi entrent clairement dans la première catégorie tandis que la méthode de descente ou la recherche Tabou vont dans la seconde.

On peut également raisonner par rapport à l'usage que font les méta-heuristiques de la fonction objectif. Certaines la laissent « telle quelle » d'un bout à l'autre du processus de calcul, tandis que d'autres la modifient en fonction des informations collectées au cours de l'exploration l'idée étant toujours de « s'échapper » d'un minimum local, pour avoir l'avantage de chance de trouver l'optimal.

Enfin, il faut distinguer les méta-heuristiques qui ont la faculté de mémoriser des informations à mesure dans leur recherche de celles qui fonctionnent sans mémoire. On distingue la mémoire à courte terme (celles des derniers mouvements effectués) et la mémoire à long terme on trouve par exemple la recherche Tabou et sans mémoire il y a le recuit simulé par exemple.

3. Analyse des principaux méta-heuristiques

Dans ce chapitre, nous allons voir des principaux méta-heuristiques en commençant par celles qui exploitent séquentiellement un seul voisinage (approche constructive), avant d'étudier celles qui exploitent plusieurs solutions à la fois (approche évolutive).

3.1 La méthode de descente

Le principe de la méthode de descente (dite aussi basic local search) commence à partir d'une solution s puis choisir un voisinage s' de s. tels que la fonction objectif f (s') < f(s).

On peut décider soit d'examiner toutes les solutions du voisinage et prendre la meilleure de toutes (ou prendre la première trouvée), soit d'examiner un sous-ensemble du voisinage.

3.2 La méthode du recuit simulé

Le recuit simulé (simulated annealing) est souvent présenté comme la plus ancienne des méta-heuristiques. Elle s'inspire d'une procédure utilisée depuis longtemps par les métallurgistes pour obtenir un alliage sans défaut, chauffent d'abord les morceaux de métal avant de laisser l'alliage refroidir très lentement. Pour simuler cette évolution d'un système physique vers son équilibre thermodynamique à une température T, la méthode du recuit simulé exploite l'algorithme de Métropolies.

Dans l'algorithme de métropolies on part d'une configuration donnée et on fait subir au système une modification élémentaire, si cette perturbation a pour effet de diminuer la fonction objectif (ou énergie) du système elle est acceptée, sinon elle est acceptée avec la probabilité expo (ΔE/T), en appliquant itérativement cette règle jusqu'a engendre une séquence de configurations qui tendent vers l'équilibre thermodynamique.

3.3 La méthode Tabou

La méthode Tabou est une technique de recherche dont les principes ont été proposés pour la première fois par Fred Glover dans les années 80 et elle est devenue très classique en optimisation combinatoire, elle se distingue par rapport au méthodes de recherche locale simples par le recours à un historique des solutions visitées pour rendre la recherche un peu moins « aveugle » il devient donc possible de sortir d'un minimum local. A l'inverse du recuit simulé qui génère de manière aléatoire une seule solution voisine $s' \in (s)$ à chaque itération, Tabou examine un échantillonnage de solutions de N(s) et retient la meilleure s' même si f (s')>f(s). La recherche Tabou ne s'arrête donc pas au premier optimum Trouvé.

3.4 La méthode GRASP

La méthode grasp (pour greedy randomized adaptative search procedure) est une méta-heuristique simple développée à la fin des années 90 par Feo et Resende.

Son algorithme contient deux phases :

-une phase de construction d'une solution

-une étape d'amélioration de la solution

La phase d'amélioration sera souvent faite grâce à une autre méta-heuristique.

3.5Les algorithmes génétiques

nous passons à une autre catégorie de méta-heuristiques, celles des méthodes dites évolutionnaires, qui manipulent un ensemble de plusieurs solutions simultanément.

La méta-heuristique la plus connue dans cette branche est celle reposant sur un algorithme génétique, inspiré du concept de sélection naturelle élaboré par Darwin.

Des individus qui représente des solutions cet ensemble des individus formera une génération /population que nous ferons évoluer pendant une certaine succession d'itérations jusqu'à ce qu'un critère d'arrêt soit vérifié, pour passer d'une génération à une autre nous soumettrons la population à des opération de sélection, croisement et mutation qui permettront de favoriser la convergence a des meilleurs individus, nous intéressons dans notre travailler sur les algorithmes génétiques et colonies des fourmis

Principe de l'algorithme :

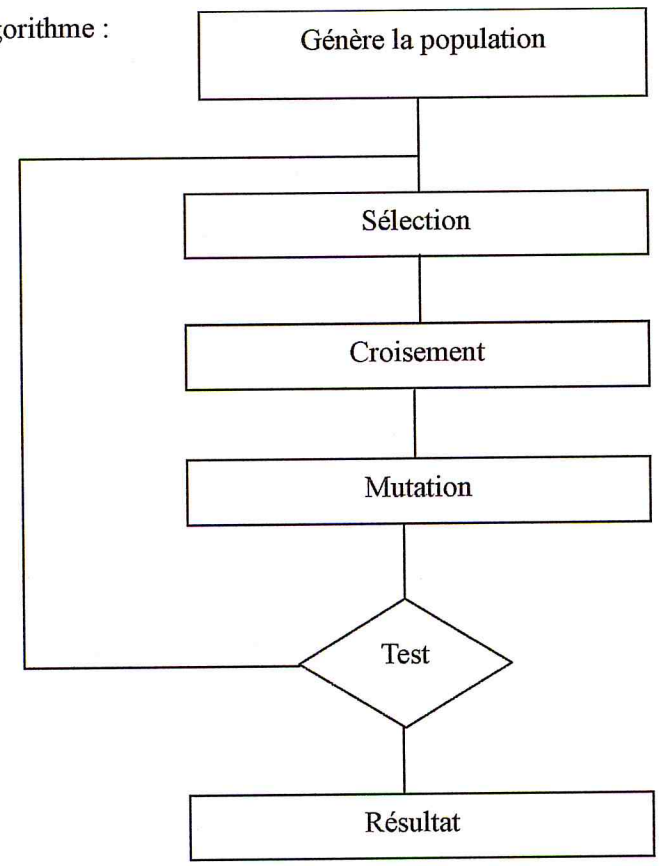


Figure 2.1 : Principe de l'algorithme génétique

1. **génère la population** : fait de manière heuristique ou aléatoire.

2. **La sélection :** consiste à retenir un individu parmi un ensemble selon un stratège par exemple aléatoire, élitiste, tournoi, … etc.

II. Techniques pour résoudre le problème de mapping

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 dégrade les performances de système.

Dans cette partie nous formulons le problème mathématiquement et racontons quelques travaux à l'intérieur :

1. Formulation mathématique de problème du mapping

Nous commençons par définir graphe de communication et le graphe de NoC, le graph de communication est un graphe oriente G (V, E) où :

- chaque sommet $v_i \in V$ représente une tache.
- $e_{ij} \in E$ représente une communication entre v_i et v_j .
- w_{ij} représente le poids d'arête e_{ij} montre les besoins en bande passante de la communication.

Le réseau sur puce représenté par un graphe N (T, L) :

- chaque sommet $t_i \in T$ représente une tuile.
- $e_{ij} \in E$ représente un lien physique entre t_i et t_j .

Le mappage de graphe de communication entre les taches sur le graphe de l'architecture NoC fait lorsque $|V| \leq |T|$, il est définie par:

$$map\ (v_i) = t_j \text{ tels que } \forall\ v_i \in V\ \exists\ t_j \in T \ldots\ldots\ldots (3)$$

La fonction pour calcule de la bande passante est définie par :

$$bw_k = H_{ij} \times w_{ij} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (4)$$

$$BW = \sum_{k=1}^{|E|} bw_k \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (5)$$

H_{ij} : représente le nombre de sauts nécessaires pour atteindre v_j a partir de v_i .

Entre deux tuiles t_i et t_j, des coordonnées respectives (X_{t_i}, Y_{t_i}) et (X_{t_j}, Y_{t_j}) dans la NoC, la distance de Manhattan est définie par :

$$\left|X_{t_i} - X_{t_j}\right| + \left|Y_{t_i} - Y_{t_j}\right| \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (6)$$

3. **Croisement** : consister a croise deux individus échangent des parties entre eux il existe plusieurs stratège par exemple un point, deux point, uniforme.

4. **mutation** : consiste a prendre un chromosome (individu) et change l'ordre des gènes selon plusieurs stratège par exemple un point, deux point, uniforme, … etc.

3.6.Les algorithmes des colonies des fourmis

Cet algorithme est encore inspiré de la nature elle été mis au point par Doringo au début des années 90. Son principe repose sur le comportement particulier des fourmis lorsqu'elles quittent leur colonie pour recherche la nourriture et finissons la rechercher par l'élaborer des chemins qui s'avèrent fréquemment être les plus courts pour aller de la colonie à une source de nourriture.

Description formelle :

La règle de déplacement est appelée « règle aléatoire de transition proportionnelle », et est écrite mathématiquement sous la forme suivante :

$$P_{ij}^{k}(t) = \begin{cases} \dfrac{\tau_{ij}(t)^{\alpha}.n_{ij}^{\beta}}{\sum_{l\epsilon J_i^k}\tau_{il}(t)^{\alpha}.n_{il}^{\beta}} & \text{si } j\epsilon J_i^k \\ 0 & \text{si } j \in J_i^k \end{cases} \quad \ldots\ldots\ldots\ldots\ldots\ldots (1)$$

Où J_i^k est la liste des déplacements possibles pour une fourmi k lorsqu'elle se trouve sur une solution i, η_{ij} la visibilité (c'est une matrice détermine par une heuristique)

Une fourmi k dépose une quantité $\Delta\tau_{ij}^{k}$ de phéromone:

$$\Delta\tau_{ij}^{k}(t) = \begin{cases} \dfrac{Q}{L^k(t)} & \text{Si } (i,j)\epsilon T^k(t) \\ 0 & \text{si } (i,j) \notin T^k(t) \end{cases} \quad \ldots\ldots\ldots\ldots\ldots\ldots (2)$$

L'algorithme des colonies des fourmis :

Initialisation des pistes de phéromone ;

While tant que critère d'arrêt non atteint :

Construire les solutions aléatoire ou guidé

Mise à jour phéromone ;

Fin.

Quelque Techniques Proposées pour la résolution du problème de Mapping utilise l'énergie comme fonction objectif et autre utilisé la bande passante ce la même chose si on minimise l'énergie ou bande passante.

2. Quelques Techniques Proposées pour la résolution du problème de Mapping

➢ NMAP

Est une heuristique qui permet de mapper IP sur la topologie 2D maille, sont but est de minimise le cout de communication, elle procède en trois etapes :

1- Placé les IPs qui ont une communication maximale avec leurs voisins.
2- Placé les IPs qui communiquent plus avec les IPs placés.
3- Placé les IPs qui communiquent plus avec les IPs déjà placés. [27]

➢ CGMAP

La technique CGMAP permet de placer des IPs sur une topologie 2D maillée en minimisons le coût de communications. Elle combine les deux idées des algorithmes génétiques et algorithme chaotique. [26]

➢ BMAP (BINOMIAL MAPPING)

Heuristique permet de placé les IPs sur un architecture 2D maille en minimisant le coût de communications, elle utilisée l'idée de transféré chaque IPs a un sous-arbre. [28]

➢ PLBMR

Heuristique permet de placé les IPs sur un architecture 2D maille en minimisant la consommation d'énergie, elle utilisé l'idée d'optimisation par l'algorithme d'essaim des particules (PSO). [29]

➢ MILP

Heuristique permet de placé les IPs sur une topologie 2D maille en minimisant l'énergie, il présente une nouvelle approche basé sur l'utilisation de la programmation linéaire (ILP integer linear programming). Il présente également une technique de clustering heuristique permettant de réduire le temps d'exécution de la formulation ILP. [30]

➢ CMGA

Heuristique permet de placé les IPs sur une topologie 2D maille basé sur l'utilisation du Chaos et les algorithmes génétiques, il optimisé la consommation d'énergie. [17]

➢ A3MAP (Architecture-Aware Analytic Mapping)

Heuristique permettant de place les IPs sur une topologie 2D maille basé sur l'utilisation de la méthode de sur relaxation successive (est une variante de la méthode de Gauss Seidel pour résoudre un système d'équation linéaire) et les algorithmes génétiques pour l'optimisation d'énergie. [41]

➢ PMAP (Physical Mapping)

Heuristique qui permet de mapper les clusters d'une application sur une topologie 2D maille, son but est de minimiser le coût des communications, elle procède en deux phases :

- La première phase consiste à placer les IPs qui communiquent le plus entre elles sur des tuiles adjacentes.
- La seconde étape permet de mapper le reste des IPs en fonction de celles déjà placées.

Tableau 2.1 : Résume des techniques proposée

Algori-thme	Benchmark	Méthode	Fonction Objectif	Remarque	Année
NMAP [27]	(Video Object Plane Decoder) VOPD	Heuristique	la bande passante équation (5)	· Placé les IPs qui ont une communication maximale avec leurs voisins. · Placé les IPs qui communiquent plus avec les IPs placés. · Placé les IPs qui communiquent plus avec les IPs déjà placés.	2004
BMAP [28]	VOPD (16 cores) , MPEG-4	Méta-heuristique de Binomial mapping	la bande passante équation (5)	-Calaculer le IP ranking -marge IP-set -actualisé IP-set	2005
PLBMR [29]	NWD, PIP, VOPD, MPEG	PSO	Optimisation d'énergie	-chaque solution est une particule -rechercher le meilleur positon dépende de sa expérience et l'expérience de sont voisin -mémorise le meilleur de particule	2007
MILP [30]	mp3 encoder mp3 dec oder	Heuristique programmation linéaire et technique de clustering	Optimisation d'énergie	-mélange les deux techniques de optimisation programmation linéaire et la méta-heuristique de clustering.	
CMGA [31]	M-JPEG encoder system (16, 18, 14)	Les algorithmes génétiques	Optimisation d'énergie	-initialise la population en chaos -calcule la fonction fitness -compare avec la meilleure solution -sélection, mutation, permutation, opération chaos.	2012
SPIRAL [32]	VOPD	Heuristique base sur l'utilisation de forme spirale	la bande passante équation (5)	Consister a place les cores de l'application à une forme spirale	2008

3. Discussion sur les Techniques de Mapping

Les méthodes proposé a minimise le cout de communication et le temps de placement de l'application on peut dire que les méthodes évolutive permet d'éviter la stagnation par contre la méthode de recherche locale reste toujours boucle dans la même solution donc elle exploitée pas bien l'espace de rechercher, alors la meilleure solution est de hybridé les deux technique évolutive et recherche locale pour exploiter bien l'espace de solution.

Conclusion

Nous avons présenté dans ce chapitre quelques méthodes de mapping. Elles utilisent les méta- heuristiques pour trouver une solution optimale ou proche de l'optimale. En effet on va proposer dans le chapitre suivant d'utiliser d'autres méta-heuristiques pour résoudre le problème de mapping.

Chapitre 3 : solution proposé

Introduction

L'objectif de notre travail était, en premier lieu de concevoir une technique de mapping permettant de placer chaque élément d'une application sur une architecture de réseau sur puce. Cependant pour tester cette technique et la comparer avec d'autres existantes un outil de tests et de simulation s'est avéré nécessaire. En l'absence d'une plateforme libre permettant de réaliser ces fonctionnalités, la conception et la réalisation d'un outil qui permet de tester les différentes techniques de mapping.

I. Résolution du problème

1. Adaptation de l'algorithme génétique pour la résolution du problème de mapping

Fonctionnement générale de l'algorithme génétique :

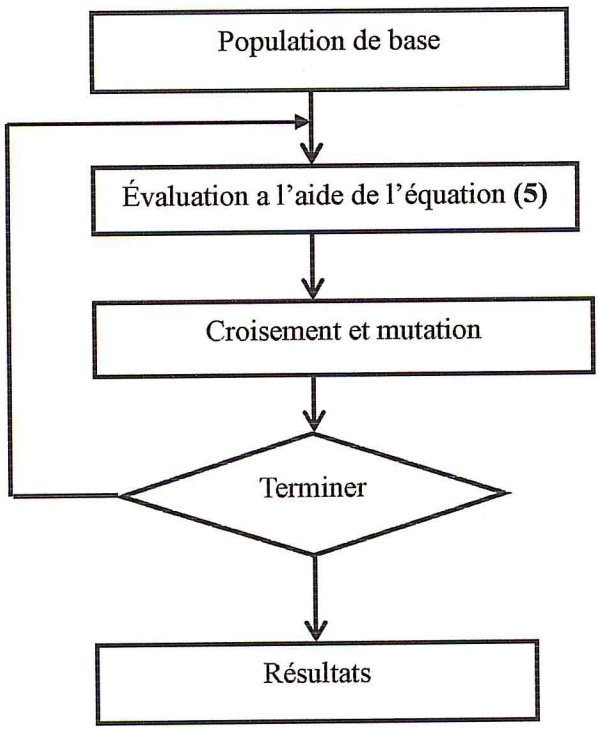


Figure 3.1 : Algorithme génétique

1.1. Phases d'adaptation de l'algorithme

La phase d'initialisation est exécutée au début de l'algorithme. Cependant, les phases de mouvement, de mises à jour et de confinement sont réitérées un certain nombre de fois.

➢ **Phases initialisation :**

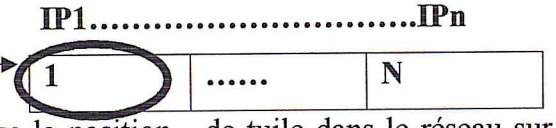
La population initiale est constituée d'un ensemble de chromosomes avec des gènes Représentent une position sur le réseau sur puce.

Les taches d'application :$IPs = \{IP_1, IP_2, \dots, IP_n\}$

Les tuiles de NoCs : $T = \{t_1, t_2, \dots\dots\dots\dots, t_m\}$

Structure chromosome (solution) : les gêné représenter des numéros de tuile :

La position dans la topologie de NoC et représente les gènes de chromosome



Où N se la position de tuile dans le réseau sur puce et chaque indice de tableau représente une tache (IPs).

➢ **Phase Evaluation :**

Chaque chromosome évalué par l'équation (5) (se la fonction fitness).

➢ **Phases sélections :**

Dans cette phase nous utilisant trois méthodes de sélection:

1. Sélection par La loterie biaisée :

Choisir deux individus à partir de population aléatoirement.

2. La méthode élitiste :

Choisir les individus possédant les meilleurs scores par rapport la fonction fitness (équation (5)).

3. La sélection par tournois :

Choisir n individu aléatoirement et prendre les deux meilleurs en fonction de l'équation (5).

➢ **Phase de croisement :**

1. **point de croisement :**

Consister a divise le chromosome en deux parties et combine les partie de deux chromosomes comme la figure suivant :

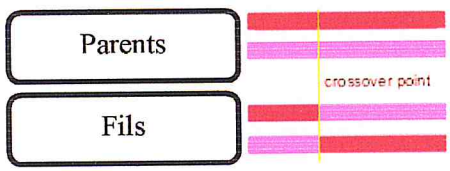


Figure 3.2 : point de croisement

2. **Deux points croisés :**

Consister a divise le chromosome en trois parties et combine les partie de deux chromosomes comme la figure suivant :

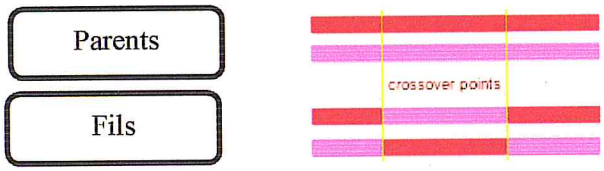


Figure 3.3 : Deux points croisés

3. **croisement uniforme :**

Consister a divise le chromosome en plusieurs parties et combine les partie de deux chromosomes comme la figure suivant :

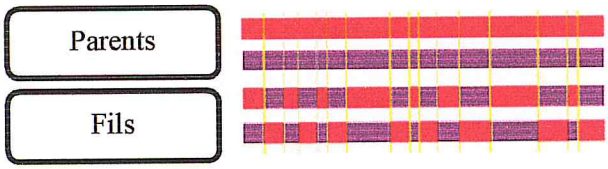


Figure 3.4 : croisement uniforme

4. **croisement shift:**

Consister a divise le chromosome en deux parties et combine les parties de chromosome comme la figure suivant :

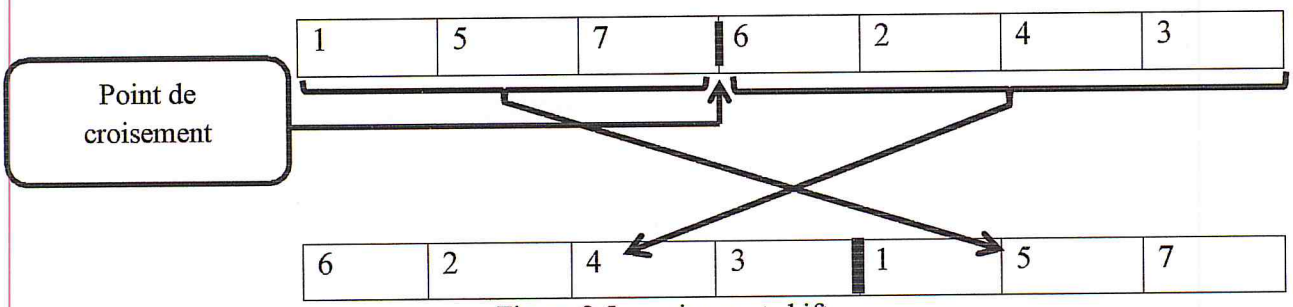


Figure 3.5 : croisement shift

➢ **Phase de mutation :**

1. **un point de mutation :**

Consister a inverse la position de deux gènes de chromosome commet la figure suivant :

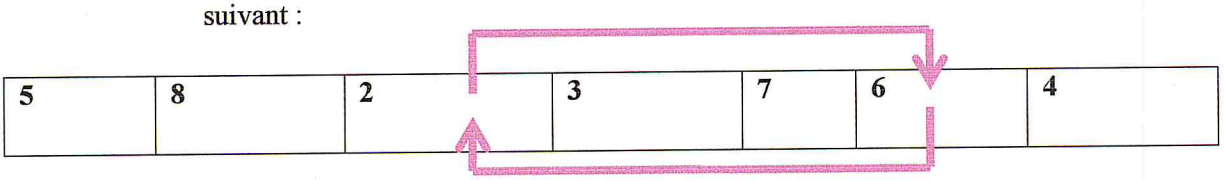


Figure 3.6 : un point de mutation

2. **deux points de mutation :**

Consister a inverse la position de quatre gènes de chromosome commet la figure suivant :

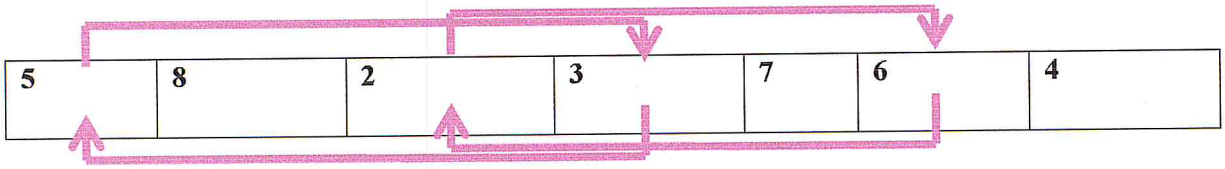


Figure 3.7 : deux points de mutation

3. **mutation Uniform :**

Consister a inverse la position de plusieurs gènes de chromosome commet la figure suivant :

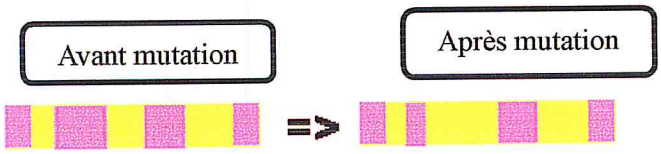


Figure 3.8 : mutation Uniform

- **Phase de remplacement :**

Consister a compare la fonction fitness (de l'équation (5)) de deux fils s'ils sont plus petites que sont parents on remplace les parents par les fils, cette stratégie améliore les performances des algorithmes évolutionnaire dans certains cas mais présente aussi un désavantage en augmentant le taux de convergence prématuré.

1.1. paramètres des algorithmes génétiques :

- nombre d'itération.
- taille de population.
- Taux croisement
- Taux mutation
- méthode de sélection
- méthode de croisement
- méthode de mutation

Optimisation par les algorithmes génétiques :

```
Initialiser
Répéter
        Pour (i=0 ; i<taille de population ; i++)
                Sélection parent (méthode de sélection)
                If (random1>taux croisement)
                        Croisement (méthode de croisement)
                If (rand1>taux mutation)
                        Mutation (méthode de mutation)
                Remplace () ;
Jusqu'à ce que  (nombre d'itération)
```

2. Les algorithmes génétiques avec croisement shift

Paramètres d'algorithme :

- nombre d'itération.
- taille de population.

Optimisation par GA avec shift :

```
Initialiser
j=0 ;
Répéter
        Pour (i=0 ; i<taille de population ; i++)
                If (j%2==0)
                        If (i< (taille de population/2))
                                Croisement shift ;
                        If (i > (taille de population/2))
                                Mutation ;
                If (j%2= =1)
                        If (i > (taille de population/2))
                                Croisement shift ;
                        If (i < (taille de population/2))
                                Mutation ;
                Remplace () ;

j + + ;
Jusqu'à ce que  (j >nombre d'itération)
```

3. Adaptation de l'algorithme des colonies des fourmis pour la résolution du problème de mapping

La fourmi désigne l'élément de base de l'algorithme des colonies des fourmis, la fourmi construire la solution du problème à traiter, a partir de ses informations sur la phéromone, la fourmi décide de son prochain mouvement.

Chaque itération, la fourmi qui construire la meilleure solution mettre a jour la matrice globale des phéromones.

Optimisation par colonies des fourmis :

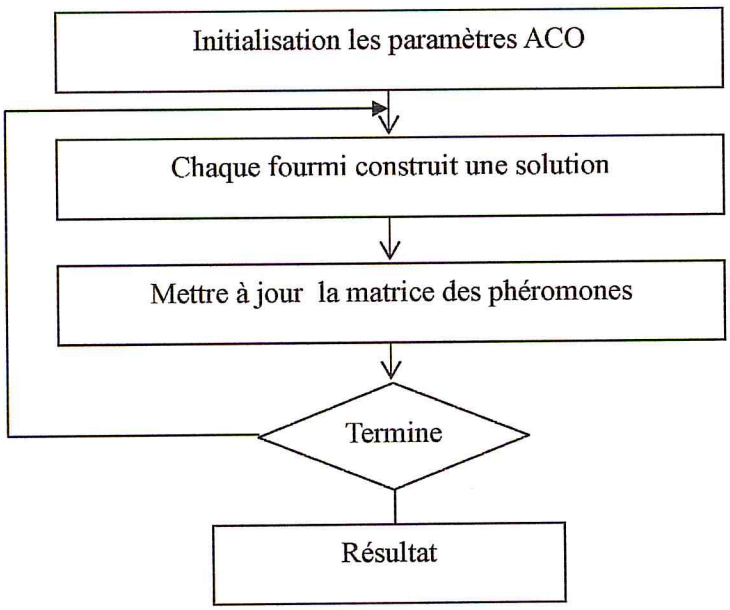


Figure 3.9 : Optimisation par colonies des fourmis

2.1. Phases d'adaptation de l'algorithme

Pour ces phase d'adaptation nous inspirons la solution propose par Jain Wang.

- On initialise le nombre d'itération NC, le nombre de fourmis k, la probabilité (phéromone) τ entre les tuiles T et les tâches d'application(IPs) , la valeur d'évaporation ρ.
- Les fourmis vont placer les tâches sur les tuiles un par un en fonction de plus grand probabilité de phéromone.
- Après toutes les fourmis terminons leur distribution des taches sur les tuiles la fourmi qui obtient la meilleur solution mettre à jour la phéromone.

2.2. paramètres de l'algorithme colonne de fourmis

- Nombre d'itération NC
- Nombre de fourmis k
- La valeur initiale de phéromone τ

- La valeur d'évaporation ρ

Optimisation par ACO :

Initialiser
Répéter

Pour (i=0 ; i< nombre de fourmi; i++)

Initialise la table de rechercher de fourmi ;
Pour (k=0 ; k<nombre d'IPs ; k++)
Affecter une IPs k a un tuile j, en
fonction de la plus grand phéromone
fin pour
Fin pour

Rechercher la meilleure fourmi qui minimises l'équation (5) la valeur de B^k ($B_{\text{meilleur}} = \min(B^k)$) ;

Mettre à jour la phéromone de la meilleure fourmi

$$\tau_{ij} = \rho\tau_{ij} + \Delta\tau_{ij}^{meilleur}$$

$$\Delta\tau_{ij}^{meilleur} = \begin{cases} \frac{1}{B^{mielleur}} & \text{Assigner une IPs j À une tuile i} \\ 0 & \text{Les autres} \end{cases}$$

Jusqu'à ce que (nombre d'itération)

1. Hybridation des colonies des fourmis avec les algorithmes génétiques

L'hybridation des deux approches permet de réunir dans un seul algorithme les avantages de chacune d'ils afin d'améliorer les résultats obtenus en exécutant chaque approche à part.

- **Phase des algorithmes génétiques :**

  Dans cette phase nous utilisons l'algorithme génétique pour attendre une initialise la matrice des phéromones.

- **Phase initialisation de matrice des phéromones :**

  Consiste a divise la matrice des phéromones en deux parties la première partie remplir par les solutions trouve par l'algorithme génétiques et la deuxième partie remplir aléatoirement.

Fonctionnement générale :

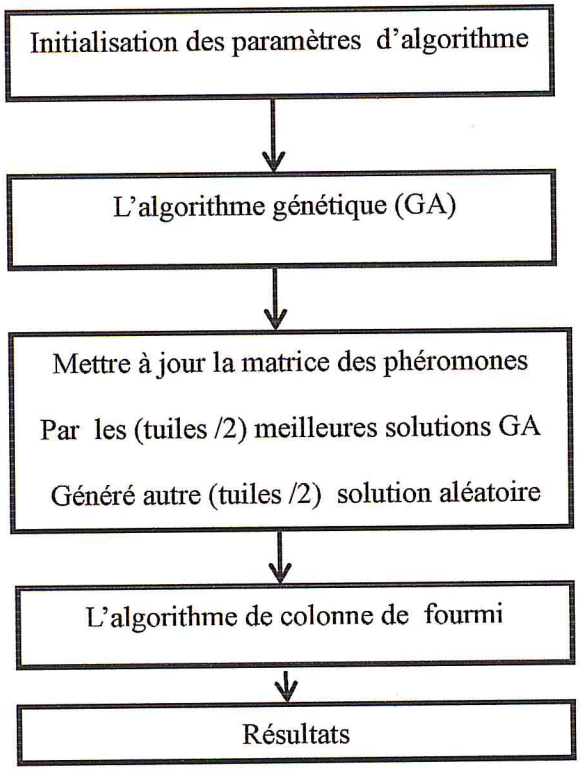


Figure 3.10 : Fonctionnement générale de l'algorithme GAANT

a. paramètres de l'algorithme colonne de fourmis

Cet algorithme inspire de l'algorithme [33] :

- nombre d'itération.
- taille de population.
- Taux croisement
- Taux mutation
- méthode de sélection
- méthode de croisement
- méthode de mutation
  - Nombre d'itération NC
  - Nombre de fourmis k
  - La valeur initiale de phéromone

GAANT :

Initialiser les paramètres de GA et ACO
Répéter
    Pour (i=0 ; i< taille de population ; i++)
        Sélection parent (méthode de sélection)
        If (random1>taux croisement)
            Croisement (méthode de croisement)
        If (random2>taux mutation)
            Mutation (méthode de mutation)
        Remplace () ;
    Fin pour
Jusqu'à ce que (nombre d'itération)
//initialisation la matrice globale des phéromones
Solution [] population ={les meilleur les (nombre des tuiles /2) solutions, généré (nombre des tuiles /2) solution aléatoire}

Pour (i=0 ; i<nombre des tuiles ; i++)
    Pour (j=0 ; j<nombre des IPs ; j++)

        Phéromone[j] [position] = $\frac{B_j}{B_{totale}}$;

    Fin pour
Fin pour

Répéter
    Pour (i=0 ; i< nombre de fourmi; i++)

        Initialise la table de rechercher de fourmi ;
            Pour (k=0 ; k<nombre d'IPs ; k++)
                Affecter une IPs k a un tuile j, en
            fonction de la plus grand phéromone
        fin pour
    Fin pour

    Rechercher la meilleure fourmi qui minimises la valeur de B^k
    ($B_{\text{meilleur}} = \min(B^k)$) ;

    Mettre à jour la phéromone de la meilleure fourmi : $\tau_{ij} = \rho\tau_{ij} + \Delta\tau_{ij}^{meilleur}$

$$\Delta\tau_{ij}^{meilleur} = \begin{cases} \frac{1}{B^{mielleur}} & \text{Assigner une IPs j à une tuile i} \\ 0 & \text{Les autres} \end{cases}$$

Jusqu'à ce que (nombre d'itération)

Conclusion

Ce chapitre nous fournissons 4 solution propose pour la résolution de problème de mapping sur une structures de NoC dans le but de minimiser le coût de communications.

Ces approches se basent sur les algorithmes génétiques, colonies des fourmis et nouvelle hybridation des deux approches pour trouver une bonne solution en fonction de l'équation (5).

Nous entament dans le chapitre suivant la conception de plate forme développée.

Chapitre 4 : Test et Résultats

Introduction

Après avoir implémenté les différentes techniques étudiées dans notre partie conception, nous présentons les résultats obtenus lors de nos expérimentations.

Nous commençons par présenter les différents benchmarks sur lesquels nous avons effectué nos tests, puis nous étudions les résultats obtenus puis le test la meilleur technique avec les techniques de littérature.

1. Présentation des benchmarks

1.2. Présentation du benchmark Video Object Plane Decoder (VOPD)

L'application VOPD est composée de 16 IPs qui communiquent entre elles à travers liens. Le nombre total de paquets envoyé est de **3731.**. L'architecture NoC choisie est une maille de taille 4x4. La Figure 5.1 décrit cette application sous forme de graphe.

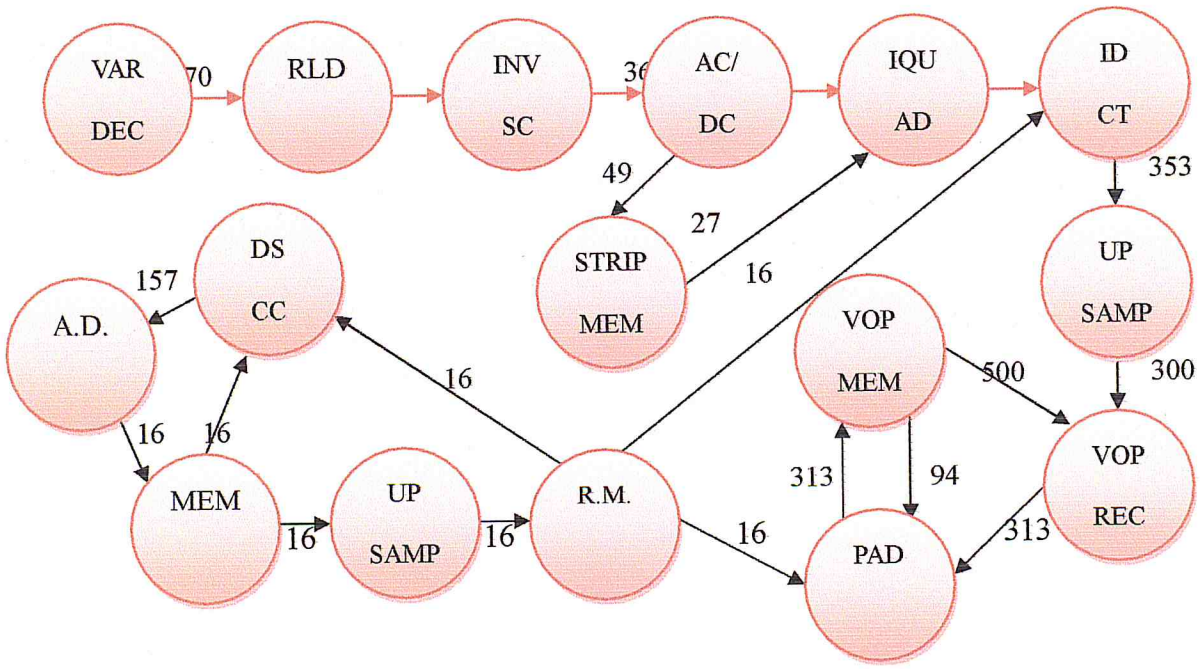


Figure 4.1 : Graphe d'application du benchmark VOPD [21]

1.1 Présentation du benchmark MPEG 4 Decoder in Decoder :

L'application MPEG 4 est composée de 12 IPs qui communiquent entre elles à travers liens. Le nombre total de paquets envoyé est de **3466.** L'architecture NoC choisie est une maille de taille 4x4. La **Figure 5.2** décrit cette application sous forme de graphe.

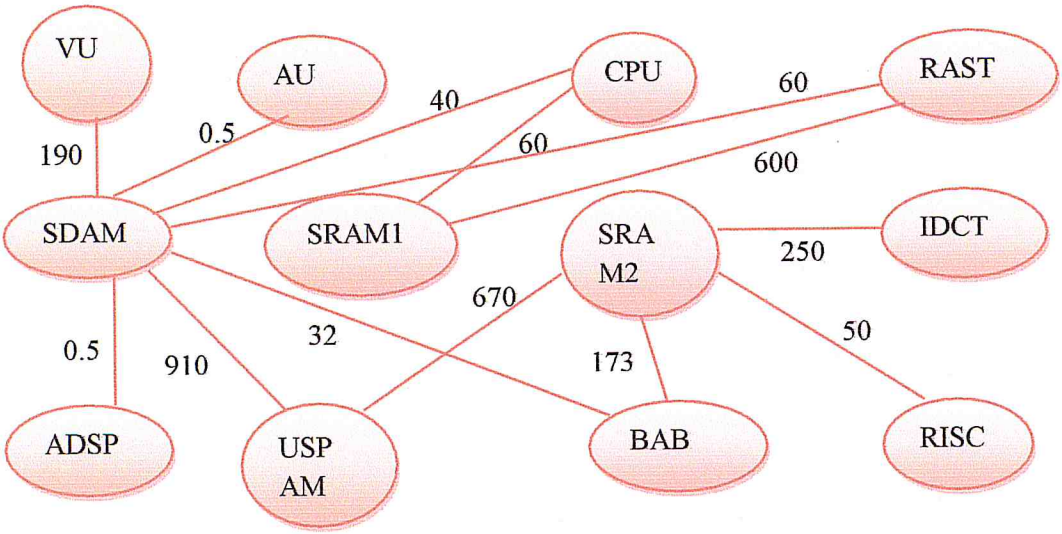


Figure 4.2 : Graphe d'application du benchmark MPEG [35]

I. Réalisation des tests

Nous étudions en premier lieu les résultats obtenus avant l'exécution de l'application sur le réseau. Le but de cette étude est de comparer les différentes techniques de mapping proposer par apporte la littérature.

La métrique prise en considération dans ce cas est le coût des communications pour cela nous utilisons les benchmarks le plus utilisé dans ce domaine VOPD et MPEG4.

1. Etude paramétrique pour GA

Cette étude est effectuée en réalisant les tests sur les deux benchmarks VOPD, MPEG4. Toutes les techniques arrivent à trouver la solution proche de l'optimale (4135 selon la littérature).nous commencé par étudie l'influence de nombre d'itération, méthode de croisement et méthode de mutation sur les performances de l'algorithme.

On a fixe la méthode de sélection élitiste, Pc=0.95, Pm=0.5, taille de population par 30

Tableau 4.2: Tableau Comparatif entre les techniques de croisement et mutation

Benchmark	Méthode de croisement	Méthode de mutation	Cout de communication
VOPD	**Deux points**	**Aléatoire**	**4125**
	Uniform	**Deux points**	**4246**
	Un point	**Aléatoire**	**4247**
MPEG	**Deux points**	**Aléatoire**	**3568**
	Uniform	**Deux points**	**3633**
	Un point	**Aléatoire**	**3752**

2. Etude paramétrique pour GA shift

Pour l'algorithme shift il y seulement deux paramètres varie :

Tableau 4.3: Tableau Comparatif de variation de taille de population et nombre d'itération

Benchmark	Nombre d'itération	Taille de population	Cout de communication
VOPD	**100**	**30**	**5100**
	500	**200**	**4819**
	1000	**200**	**4393**
MPEG	**100**	**30**	**4742**
	500	**200**	**4167**
	1000	**200**	**4000**

Nous remarquons que les résultats obtenus ne sont pas aussi bons que l'optimale. En effet, ceci explique que il y a pas une stagnation, en effet il n'exploite pas l'espace de rechercher locale.

3. Etude paramétrique pour colonies des fourmis :

Etude d'influence des nombre d'itération et nombre de fourmis sur la performance de l'algorithme, on a fixons $\tau = 0.0002$ et $\rho = 20$

Tableau 4.4: Tableau Comparatif d'influence de nombre d'itération et nombre de fourmis

Benchmark	Nombre d'itération	Nombre de fourmis	Cout de communication
VOPD	**1000**	**500**	**4167**
	1000	**10**	**4205**
	1000	**30**	**4186**
MPEG	**1000**	**500**	**3900**
	1000	**10**	**4131**
	1000	**30**	**4151**

Nous remarquons que les résultats obtenus avec ne sont pas aussi bons que l'optimale, en effet le nombre de fourmi favoriser l'exploration de l'espace de recherche et ainsi augmenter les chances de s'approcher le plus de la solution optimale.

I. Etude paramétrique pour GANT (les algorithmes génétiques et colonies des fourmis).

Nous fixons la méthode de sélection élitiste, Pc=0.95, Pm=0.5, taille de population par 30 $\tau = 0.0002$ et $\rho = 20$.

Tableau 4.5: Tableau Comparatif d'influence de nombre d'itération

Benchmark	Nombre d'itération GA	Nombre d'itération ACO	Cout de communication
VOPD	**1000**	**200**	**4135**
	1000	**120**	**4183**
	1000	**100**	**4157**
MPEG	**1000**	**200**	**3772**
	1000	**120**	**3632**
	1000	**100**	**3567**

Nous remarquons que les résultats obtenus avec une sont pas aussi bons que l'optimale, en effet l'algorithme génétique favoriser l'algorithme des colonies des fourmis d'augmenter les chances de s'approcher a la solution optimale.

II. Etude comparative entres les technique implémente :

Après avoir effectué une étude paramétrique, nous comparons maintenant les résultats obtenus.

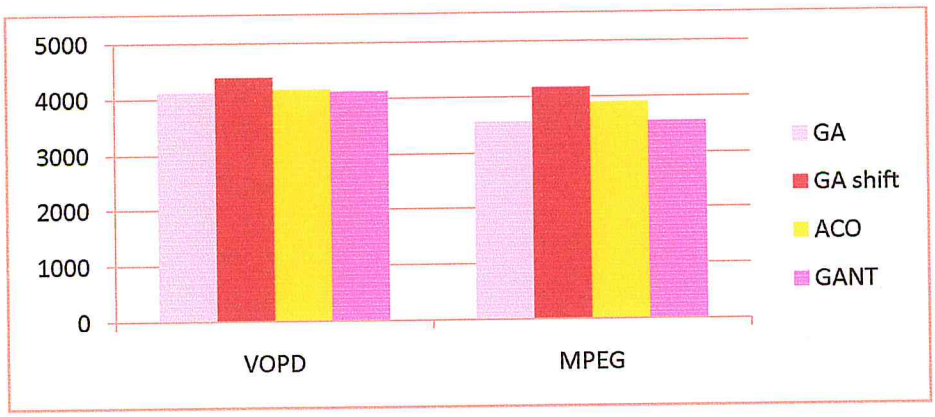


Figure 4.3: Comparaison entre les techniques

D'après l'histogramme, nous remarquons que l'introduction d'opérateurs d'évolutionnaire donne des meilleurs résultats par rapport les autres. Le choix de l'opération de sélection, le taux de croisement et taux de mutation influence aussi la diversification de la

recherche. En effet l'introduction de l'opérateur de croisement shift ne donne pas d'aussi bons résultats que l'opérateur de croisement a deux points.

II. Comparaisons Avec les techniques publiées dans la littérature :

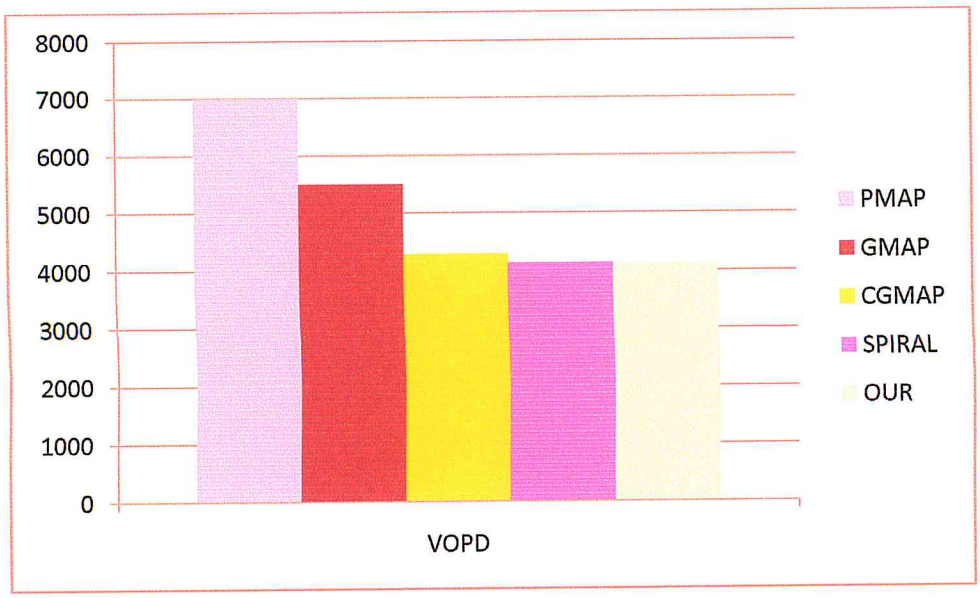


Figure 4.4 : Comparaison entre différentes techniques de mapping statique appliquées sur le VOPD

Afin l'histogramme démontrer que les choix faits apporté une amélioré sur les techniques déjà élaborées.

Conclusion

A travers les tests effectués, notre but était d'étudier les performances des techniques d'Hybridation de deux méta-heuristiques .

La première étude a démontré que GA avec son meilleur solution obtenu les algorithmes évolutifs produisait de meilleurs résultats.

Le résultat obtenu montre que les algorithmes génétiques et colonne de fourmis et l'hybridation des deux converge toujours vers la solution de littérature.

De plus, nous avons implémentés 4 méthode pour résolution avec une de l'opérateur à introduire influençait les résultats.

Nous avons utilisé dans notre technique une hybrides des deux approche qui permis de développer l'algorithme de colonne de fourmi, cette hybridation permis d'améliore les résultats de fourmis seulement.

Conclusion générale

Le but principal du travail présenté dans ce document est de proposer une technique de Mapping des IPs d'une application sur une structure de réseau sur puce en minimisant le coût de communications, après le passage une revue sur les différentes solutions proposées nous avons cherché à emprunter une nouvelle voie pour la résolution du problème.

D'après les recherches effectuées, nous avons remarqué que les méthodes hybrides (Méthodes Evolutives et Optimisation par colonies des fourmis) n'avaient jamais été utilisées pour résoudre le problème de mapping, donc nous proposons une nouvelle solution basée sur l'hybridation des deux méta-heuristiques.

En fait il n'existe pas une plateforme qui permette de tester les techniques de mapping et comparer les solutions implémentées et simuler le réseau sur puce, nous avons développé un nouvel environnement qui réalise ces fonctions,

Perspectives

Certains enrichissements peuvent être apportés à notre travail, d'abord nous implémentons 4 métas heuristiques puis nous essayons d'achève.

- Développer des nouvelles techniques d'ordonnancement.
- combinaison de méta heuristique pour l'optimisation pas seulement de cout de communication mais le temps d'exécution, l'énergie, la surface …etc.
- inclure l'utilisation de description VHDL de IPs (tache).
- Inclure un réseau sur puce hétérogène.

Il plan de chose qu'il faut ajouter avant que cette plat forme sera complet.

Bibliographie

[1] **RISO Séverine, TORRES Lionel, SASSATELLI Gilles, ROBERT Michel, MORAES Fernando:** « Réseau d'interconnexion pour les Systèmes sur puce : Le Réseau HERMES », SCD'04 : Signaux, Circuit set Systèmes, 2004. Ŕpp. 34-40.

[2] **G. E. Moore**: « Cramming more components onto integrated circuits », journal, Electronics Magazine.1965.

[3] **Kai Hwang**: « Computer Architecture and Parallel Processing», design Automation and Test in Europe Conference, March 2003.

[4] **Nicolas Wojcik** : « COHÉRENCE ET COMPLÉTUDE DES ADÈLES DE LACEMENT DE GASPARD ». Thèse de doctorat .Université des Sciences et Technologies de Lille. 2008, pp.1-10.

[5] **A. Rowson, J.**A, Sangiovanni Vincentelli : « Interface-based design », Proc. Design Automation Conference, Juin 1997, p. 178–183.

[6] **LEMAIRE Romain** : « Conception et modélisation d'un système de contrôle d'applications de télécommunication avec une architecture de réseau sur puce ».Thèse de doctorat. Institut National Polytechnique de Grenoble (CEA-LETI), 2006.

[7] **K. Lahiri, A. Raghunathan, S. Dey**: « Evaluation of the traffic-performance characteristics of system-on-chip communication architectures », in Proc. Fourteenth International Conference on VLSI Design, 3–7 Jan. 2001, pp. 29–35.

[8] **C. Zeferino, M. Kreutz, L. Carro, A. Susin:** « A study on communication issues for systems-on-chip», in Proc. 15th Symposium on Integrated Circuits and Systems Design, 2002, pp. 121–126.

[9] **W. Dally and B. Towles:** « Route packets, not wires: on-chip interconnection networks», in Proc. Design Automation Conference, 2001, pp. 684–689.

[10] **S. Evain :** « uSpider Environnement de Conception de Réseau sur Puce », Ph.D. dissertation, IETR–INSA Rennes, Lab-STICC–UBS Lorient, 2006.

[11] **C. Grecu, P. P. Pande, A. Ivanov, R. Saleh**: « Structured interconnect architecture: a solution for the non-scalability of bus-based SoCs », in GLSVLSI 04: Proceedings of the 14th ACM Great Lakes symposium on VLSI. ACM, 2004, pp. 192–195.

[12] **R. Lemaire** : « Conception et modélisation d'un système de contrôle d'applications de télécommunication avec une architecture de réseau sur puce (NoC), » Ph.D. dissertation, Institut National Polytechnique de Grenoble, 2006.

[13] **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.

[14] **MEHRAN Armin, SAEIDI Samira, KHADEMZADEH Ahmad, AFZALI KUSHAAli:** « SPIRAL: A heuristic mapping algorithm for network on chip ». IEICE Electronics Express, 2007

[15] **Romain LEMAIRE** : « Conception et modélisation d'un système de contrôle d'applications de télécommunication avec une architecture de réseau sur puce (NoC) », Thèse, INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE ,2006.

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

[17] CIDON Israel, KEIDER Idit: « Zooming in on Network on Chip Architectures ».Technical Report CCIT 565, Technion Department of Electrical Engineering, December 2005.

[18] **NICOPOULOS Chrysostomos, NARAYANAN Vijaykrishnan, DAS Chita R:** «Network-on-chip architectures: a holistic design exploration ». The Journal of Systems Architecture. Springer, 2009.

[19] **www.design-reuse.com/news/9888/arteris-products-building-networks-chip-noc.html**

[20] **www.rd.springer.com/chapter/10.1007/978-1-4020-6488-3_12**

[21] **CARVALHO Ewerson, MARCON César, CALAZANS Ney, 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.

[22] **ANDRE Françoise, PAZAT Jean-Louis** : « Le placement de tâches sur des architectures parallèles ». Revue TSI : Technique et science informatiques 1988.

[23] **BAPTISTE AUTIN: «Les méta-heuristiques en optimisation combinatoire** ». MEMOIRE, 9 mai 2006.

[24] **www.fr.wikipedia.org/wiki/Algorithme_de_colonies_de_fourmis**

[25] **Misagh Tavanpour, Ahmad Khademzadeh, Somayyeh Pourkiani, Mehdi Yaghobi** : « GBMAP: An Evolutionary Approach to Mapping Cores onto a Mesh-based NoC Architecture», Journal. Islamic Azad University, Tehran Mars 2010.

[26] **MOEIN-DARBARI Fahime, KHADEMZADE Ahmad, GHAROONI-FARD**

Golnar: « CGMAP: A new approach to network-on-chip mapping problem ». IEICE Electronics Express, 2009.

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

[28] **Wein-Tsung Shen, Chih-Hao Chao, Yu-Kuang Lien, An-Yeu (Andy) Wu**: « A NEW BINOMIAL MAPPING AND OPTIMIZATION ALGORITHM FOR REDUCED-COMPLEXITY MESH-BASED ON-CHIP NETWORK»journal, 2012.

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

[30] **www.codentest.com**

[31] **JENA R.K:** « APPLICATION MAPPING OF MESH BASED NOC USING EVOLITIONARY ALGORITHM», Journal**.** Institute of Management Technology, Nagpur, India,2012.

[32] **Armin Mehran, Ahmad Khademzadeh, Samira Saeidi**: « A Heuristic Dynamic Spiral Mapping algorithm for network on chip », journal, Islamic Azad University , 2008.

[33] **Jian WANG1, Yubai LI, Song CHAI, Qicong PENG** : « Bandwidth-Aware Application mapping for NoC-Based MPSoCs».Journal of Computational Information Systems,2011

[34] **Stewart Baird**: «Sams Teach Yourself Extreme Programmingin 24 Hours »,Sams Publishing, octobre 23, 2002.

[35] **Jian WANG1, Yubai LI, Song CHAI, Qicong PENG**: « Bandwidth-Aware Application Mapping for NoC-Based MPSoCs»,journal ,University of Electronic Science and Technology of China, 2011

[36] **DELORME Julien, HOUZET Dominique** : « Technique de mapping pour les réseaux sur puce 2D »,the Journal of Systems Architecture, Septembre 2007.

[37] « A Survey of Meta-Heuristic Solution Methods for Mapping Problem in Network-on-Chips»

[38] **Bernhard Rumpe**: «Executable Modeling with UML A Vision or a Nightmare», Munich University of Technology.

[39] **Krishnan Srinivasan , Karam S. Chatha:** « A Technique for Low Energy Mapping and Routing in Network-on-Chip architectures». Journal, National Science Foundation.

[40] www.extremeprogramming.org

[41] **Wooyoung Jang, David Z. Pan**: «Architecture-Aware Analytic Mapping for Networks-on-Chip» Department of Electrical and Computer Engineering, University of Texas at Austin, 2010.

[42] wwwunix.mcs.anl.gov/dbpp/text/node45.html#eqlamatm