F.S.D .....N° D'ordre .....

**Université Saad Dahlab de Blida**

**Faculté des Sciences**

**Département d'informatique**



**Mémoire présenter par :**

*DAHMAN Souad*
*RAZALI Hayet*

En vue d'obtenir le diplôme de Master

**Domaine :** Mathématique et Informatique

**Filière :** Informatique

**Spécialité :** Génie logiciel (**GL**)

**Thème :**

# Optimisation Multicritères pour la Conception des Réseaux sur Puce

**Soutenue le:** 29 Septembre 2013 **, devant le jury composé de :**

| | |
|---|---|
| M^me boumahdi | Président |
| M^elle Toubaline | Rapporteur |
| M^me Arkam | Examinateur |
| M^r Bougherara | Promoteur |
| M^elle Zahra | Encadreur |

**Année Universitaire**

***2012/2013***

# Remerciement

*Nous remercions tout d'abord* ***ALLAH le*** *tout puissant qui a donné la force de mener à bon terme ce modeste travail.*

*Nous tenons à remercier sincèrement tous les personnes qui ont apporté leur contribution à l'aboutissement de ce projet.*

*Nous remercions notre promoteur* ***Mr.Bougherara*** *pour son soutien et ses précieux conseils.*

*Nos vifs remerciements vont tous d'abord à* ***Melle.Zahra*** *qui nous aide à développer notre travail.*

*Nous remercions aussi tous les enseignants de département d'informatique pour les efforts consacré pour nous transmettre le savoir*

*Notre reconnaissance s'adresse à toutes les personnes qui chères et ceux qui nous ont aidée de près ou de loin.*

*En fin, on tient à remercier les membres de jury qui vont faire l'honneur d'apprécier notre travail.*

# Dédicace



*Merci Allah (mon dieu)*

*de m'avoir donné la capacité d'écrire*

*et de réfléchir, la force pour terminer mes études*

*et ce modeste travail.*

*Je dédie à celle qui m'a donné la vie, le symbole de tendresse, qui s'est sacrifiée pour mon bonheur et ma réussite, à ma mère …*

*A mon père, école de mon enfance, qui a été mon ombre durant toutes les années des études, et qui a veillé tout au long de ma vie à m'encourager, à me donner l'aide et à me protéger.*

*Que dieu les gardes et les protège.*

*A mes adorables sœurs Razika, wahida.*

*A mes frères Halim, Mohammed.*

*A mon cher binôme Hayet.*

*A mes amies et mes collègues.*

*A Tous mes enseignants.*

*A tout qui m'aiment et qui j'aime.*

SOUAD

# Dédicace



*Je dédie ce travail à :*

*A la plus ma merveilleuse mère dans le monde*

*Et à mon cher père*

*A mon très cher époux Hocine qui a souffert de mes absences et qui a assumé toutes mes responsabilités durant mes absences pour atteindre cet objectif.*

*A ma belle fille Maria (Amina).*

*À mon frère Amine et ma sœur Ibtissem.*

*A toute ma famille.*

*A mon cher binôme Souad.*

*A mes amies surtout Aicha, Hakima et Radia*

*A mes collègues.*

*A Tous mes enseignants.*

*A tout qui m'aiment et qui j'aime.*

HAYET

# Résumé

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 NoC. Elle permet de placer les éléments d'une application sur l'architecture NoC. Le but est de minimiser le coût de communication, la consommation d'énergie et l'espace.

C'est dans ce cadre que s'insère notre travail de réaliser une nouvelle méthode d'optimisation multicritère pour la conception du réseau sur puce , le but de ce sujet est de rendre le problème de mapping plus difficile ,puisque on essaye d'ajouter des critères de faisabilité sur la solution de mapping avant d'optimiser les résultats obtenues par le mapping. Ces nouvelles solutions sont basées sur l'optimisation par l'algorithme de Recuit simulé.

**Mots clés :**

Réseaux sur puce, Systèmes sur puce, communication, mapping, optimisation, l'algorithme de Recuit simulé.

# 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 parts of the intensive application on NoCs tiles. The goal is to minimize communications cost energy consumption and space.

It's in this context that fits our work. It consists of proposing new technique multi-criteria optimization method for mapping applications onto NoC architecture, the goal of this is to make the mapping problem more difficult, since we try to add feasibility criteria of the mapping solution before optimizing the results obtained by the mapping. These new solutions are based the optimization algorithm simulated annealing.

**Keywords:**

Networks on chip, systems-on-chip, communication, mapping, optimization, simulated annealing algorithm.

# Table des matières

# Liste des Figures

# Liste des Tableaux

**ASIC:** Application Specific Integrated Circuit

**CGMAP:** Chaotic-Genetic Mapping

**CMP:** Chip Multi Processor

**CPU:** Central Processing Unit

**DMA:** Direct Memory Access

**DSM:** Dynamic Spiral Mapping

**DSP:** Digital Signal Processor

**EPAM:** Energy and Performance Aware Mapping

**FLIT:** Flow Control Unit

**FPGA:** Field-Programmable Gate Array

**GBMAP:** Genetic-based Mapping

**GMAP:** Genetic Mapping

**IP:** Intellectual Property

**MPSoC:** Multiprocessor System on Chip

**MWD:** Multi Window Display

**NoC:** Network on Chip

**PAQ :** Problème d'Adéquation Quadratique

**SoC :** System on Chip

**VOPD:** Video Object Plane Decoder

# *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 ont 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 (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.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. Il devient nécessaire d'élaborer des outils et des méthodes qui automatisent ce processus.

Notre travail s'insère de réaliser une nouvelle méthode d'optimisation multicritère pour la conception du réseau sur puce , le but de ce sujet est de rendre le problème de mapping plus difficile ,puisque on essaye d'ajoute des critères de faisabilité sur la solution de mapping avant de d'optimiser les résultat obtenue par le mapping. Ces nouvelles solutions sont basées sur l'optimisation par l'algorithme de Recuit simulé tout en minimisant le coût de communications, la consommation d'énergie et l'espace.

Le document est organisé en 5 chapitres :

- Le premier chapitre introduit les notions liées aux réseaux sur puce, les composants de cette structure, ses caractéristiques ainsi que quelques exemples d'architectures sont présentés.
- Le second est consacré à la conception des réseaux sur puce. Les phases et outils de mise en œuvre sont définis tout en mettant l'accent sur la phase de mapping et ses méthodes de résolution. Quelques techniques proposées dans la littérature sont introduites.
- Le troisième chapitre est consacré à l'optimisation multi critère (multi objectifs) avec ses concepts ainsi quelques méthodes permettent de résoudre le problème de mapping.
- Le quatrième chapitre présente une proposition pour la résolution du problème de mapping il s'intéresse à l'approche de résolution Recuit simulé avec son détaille ainsi que l'adaptation au problème de mapping et ses formalismes mathématiques qui les entourent.
- Dans le dernier chapitre, des tests sont effectués afin de comparer la technique proposée avec quelques méthodes existantes. Ce mémoire s'achève par une conclusion générale et quelques perspectives de notre travail.

*Chapitre I*

# Les réseaux sur Puce (NoCs)

## Introduction

Durant les dernières décennies, on est passé de la conception de circuits composés de quelques milliers de portes à des systèmes structurés et intégrés comme un réseau sur une même puce .Les puces modernes peuvent contenir plusieurs processeurs, de la mémoire et un réseau de communication complexe. Le principe de la conception reste le même. Il s'agit de générer une réalisation physique sous forme d'une puce en partant d'une spécification système. Par contre, les outils mis en œuvre et l'organisation du travail durant le processus de conception ont beaucoup évolué. Partant d'une conception complètement manuelle où l'on dessinait les masques du circuit à réaliser sur du papier spécial, on est passé à une conception quasi automatique en partant d'une description du comportement du circuit sous forme d'un programme décrit dans un langage de haut niveau.

## 1. Les systèmes embarqués :

Les systèmes embarqués sont les systèmes informatiques d'employés dans le monde physique. Ils captent des informations de leur environnement, s'adaptent et agissent sur lui sans intervention humaine .Ils communiquent généralement à travers des réseaux qui véhiculent des flux de données toujours plus importants.

Les systèmes embarqués incluent tous les appareils intégrant des logiciels dans leur fonctionnement, que ceux-ci soient visibles à leurs utilisateurs ou non. Cette thématique réunit de nombreuses compétences (informatique, électronique, architecture de systèmes, mathématiques). Elle nécessite de prendre fondamentalement en compte l'hétérogénéité des composants et des logiciels .Il faut donc maîtriser la complexité qui en d´écoule en développant des techniques d'abstraction et de modèles hétérogènes tout en impulsant une forte coordination afin de traiter les questions de normalisation essentielles à une forte interopérabilité **[MOH 10].**

L'augmentation constante du nombre de fonctions intégrées dans un seul système a fortement augmenté les contraintes et les exigences d'architecture en termes de composants et les réseaux de communication .La **Figure1** simplifie la vue des tendances de conception adoptée.

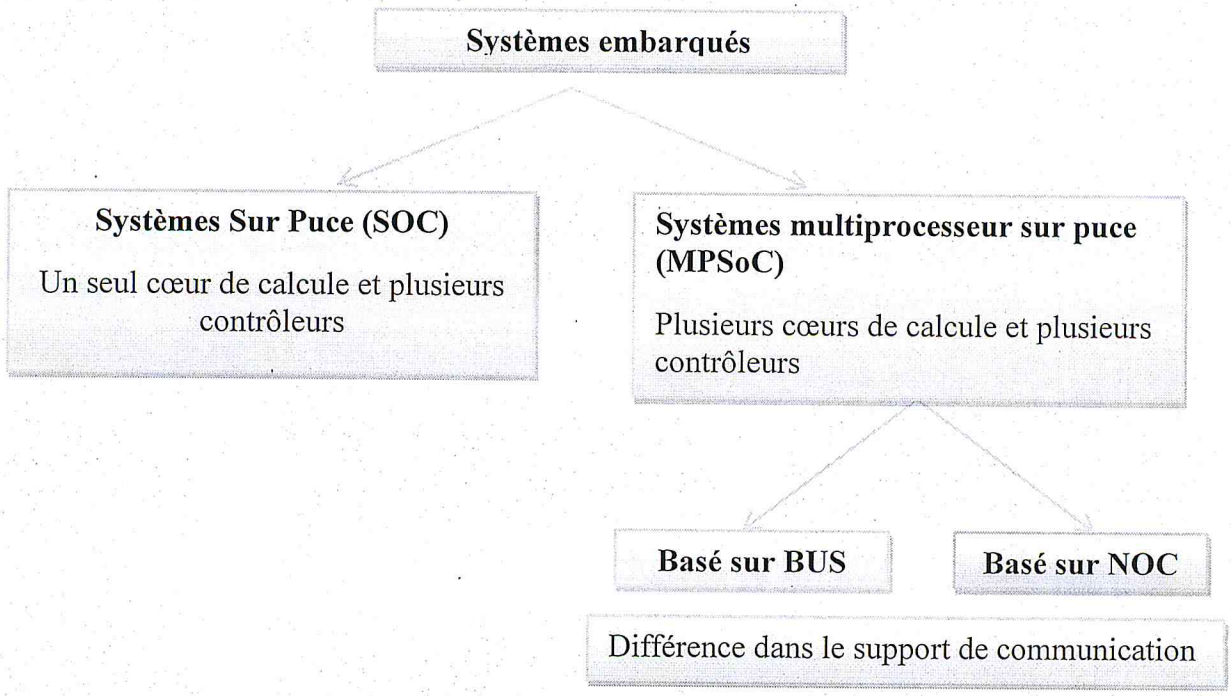


**Figure 1:** Les tendances de conception actuelle et future d'un système embarqué **[ASH 06]**.

### 1.1. Les caractéristique d'un système embarqué :

On énumère les caractéristiques principales suivantes :

- C'est un système principalement numérique.
- Il met en œuvre généralement un ou plusieurs processeurs.
- Il exécute une application logicielle dédiée pour réaliser une fonctionnalité précise et n'exécute donc pas une application scientifique ou grand public traditionnelle.
- Il n'a pas réellement de clavier standard (bouton poussoir, clavier matriciel...).

Ce n'est pas un PC en général mais des architectures similaires à basse consommation d'énergie qui sont de plus en plus utilisées pour certaines applications embarquées **[ABO 08]**.

### 1.2. Domaines d'applications des systèmes embarqués :

Les domaines dans lesquels on trouve des systèmes embarqués sont de plus en plus nombreux **[ABO 08]:**

- Transport : Automobile, Aéronautique (avionique) etc.
- Astronautique : fusée, satellite artificiel, sonde spatial, etc.
- Militaire : missiles.
- Télécommunication: Set-top box, téléphonie, routeur, pare-feu, serveur de temps, téléphone portable etc.
- Electroménager : télévision, four à micro-ondes.
- Impression : imprimante multifonctions, photocopieur etc.
- Informatique : disque dur, lecteur de disquette etc.
- Multimédia : console de jeux vidéo, assistant personnel.
- Guichet automatique bancaire (GAB).
- Equipement médical.
- Automate programmable industriel, contrôle-commande.
- jeux : consoles.
- Métrologie.

## 2. Système sur puce (SOC) :

L'importante évolution de la technologie de fabrication des systèmes électroniques lors ces dernières années a permis l'apparition d'une nouvelle génération de systèmes appeler les systèmes monopuces (Soc : System on a Chip). Ils apportent des changements importants dans les méthodologies de conception des systèmes classiques **[IAR 01]**.

Un SoC désigne l'intégration d'un système complet sur une seule puce de silicium. Mais le plus important c'est d'avoir à la fois du matériel et du logiciel **[WRO 99]**.

Dans la **Figure 2** est présenté un exemple de système mono-puce typique. Il se compose d'un cœur de processeur (CPU), d'un processeur de signal numérique (DSP), de la mémoire embarquée, et de quelques périphériques tels qu'un DMA (Direct Memory Access) et un contrôleur d'E/S.

De nos jours, nous assistons à un immense progrès de ces systèmes. Ils sont de plus en plus utilisés dans les ordinateurs, téléphones portables ou encore consoles de jeux. Cette évolution a donné naissance à des architectures de plus en plus complexes exécutant plusieurs fonctionnalités à la fois.

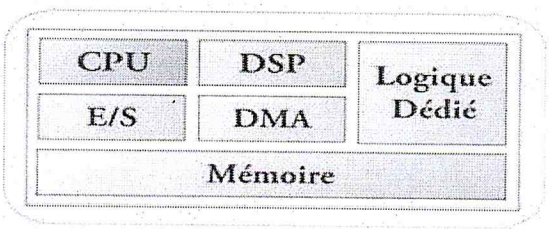


**Figure 2 :** Schéma typique d'un SOC **[LAK 98]**.

### 2.1. Caractéristiques des SOC :

- **Hétérogénéité:**

Le système mono puce n'encapsule pas des systèmes purement électroniques, il peut intégrer aussi des composants non électroniques .C'est le cas par exemple, des systèmes intégrant des éléments micromécaniques. Une première classification se fait entre parties analogiques et parties numériques. Un exemple quotidien de tels systèmes, dits hybrides, est le téléphone portable, où la partie radio (analogique) et la partie traitement de signal et gestion de l'interface utilisateur (numérique) cohabitent souvent sur la même puce.

- **Embarquement:**

Les systèmes embarqués sont par définition des sous-systèmes englobés par des systèmes plus larges, cela dans le but d'avoir des fonctions particulières. Les SOC sont aussi des systèmes embarqués car sont toujours appelés à évoluer dans

l'environnement du système qui les englobe. L'interaction entre les SOC et leurs environnements se fait via des convertisseurs analogique/numérique.

- **Spécification:**

Les systèmes mono-puce sont différents par rapport aux ordinateurs qui ont un usage général, mais un SOC est destiné à faire une tâche particulière bien définie. Cette caractéristique a fait la différence au niveau de la méthodologie de conception entre les systèmes mono-puce et les systèmes à usage général. Il faut indiquer que grâce aux méthodologies de conception des SOC il est maintenant possible de réutiliser des composants des autres systèmes déjà conçu, mais dans le cadre des spécifications du SOC actuel **[BOU 12]**.

## 3. Multiprocesseurs dans SOC (MPSoc) :

Le besoin de puissance de traitement, 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 d'où la naissance des architectures MPSoc en deux génération :

- **Multiprocesseurs de 1ère génération :**

La première tendance est l'utilisation d'architectures monoprocesseurs tout en améliorant considérablement les performances du CPU et l'utilisation de coprocesseurs. Les traitements de données, autrefois pris en charge par des accélérateurs matériels dédiés, sont maintenant effectués par un ou plusieurs processeurs spécialisés (Exemple: Digital Signal Processor).Cette solution permet d'exploiter à la fois la puissance et la flexibilité des processeurs spécialisés (**Figure 3**). Dans telles architectures, la communication est de type maître/esclave. Elle peut se faire par mémoire partagée ou par des canaux de communication point-à-point spécialisés **[ATA 07]**.

- **Multiprocesseurs de 2éme génération :**

Consiste à utiliser des architectures microprocesseurs. L'idée principale d'un tel concept matériel tient à la fois d'une stratégie diviser pour régner et du fait que les SoCs sont généralement dédiés à une classe d'applications donnée. En particulier, le concept de MPSoC se concentre sur l'amélioration des performances d'une

application logicielle.

Cette génération d'architectures est massivement parallèle et hétérogène, pour satisfaire la complexité croissante des applications embarquées, qui exige des capacités de traitement très importantes. Cette augmentation à une conséquence directe sur les architectures des systèmes mono-puce. En effets, un problème important auquel font face ces architectures concerne la communication, vu la quantité importante d'informations qui doit être échangée entre les
Différents composants de l'architecture **[MOH 10]**.

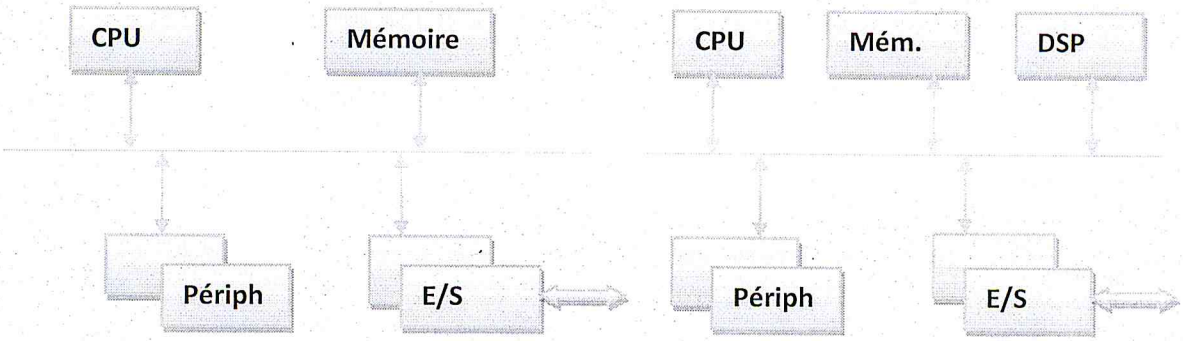


**Figure 3**: Architecture Monoprocesseur et architecture Multiprocesseur **[BOU 06].**

## 4. Les Interconnexions au sein des Socs :

Le besoin d'intégrer plusieurs fonctionnalités dans un même système a donné naissance aux architectures MPSocs. L'interconnexion entre les modules (IPs) composant ces systèmes a du suivre ce progrès et a subi une évolution structurelle et topologique **(Figure 4)**.

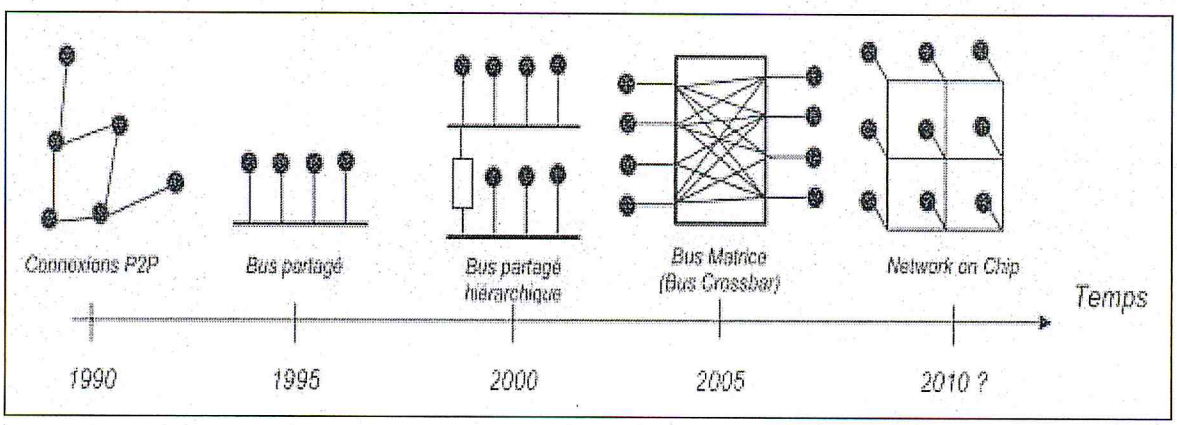


**Figure 4**: Evolution des interconnexions sur les SoCs **[TAN 08]**.

### 4.1. Connexion Point-à-point :

La connexion point à point permet de relier deux IPs directement sans avoir à définir un protocole de gestion de communication **[DEL 07]**.

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 **[DAL 01]**.

### 4.2. Connexion par Bus Partagé :

Dans une interconnexion par bus partagé, un seul cœur à la fois peut utiliser le médium (bus). Un protocole d'arbitrage gère les communications et résout les conflits d'accès (Stratégie par priorité) **[RIS 05]**.

Le bus présente plusieurs avantages tels que la possibilité de diffuser des informations, la flexibilité et la réutilisation des IPs. Cependant, cette structure n'est pas extensible et toute forme de parallélisme est exclu (Les tâches peuvent être exécutées en parallèle, mais une seule communication se fait à la fois).

### 4.3. Connexion par Bus Hiérarchique :

La structure de bus hiérarchique a été proposée afin de pallier aux problèmes rencontrés dans l'interconnexion par bus partagé. Elle connecte plusieurs segments de bus deux à deux par l'intermédiaire d'un pont (Bridge) **[DEL 07]**.

Cette approche réduit le nombre d'IPs connectés sur un même bus ce qui diminue la longueur de celui-ci et améliore les performances. De plus, il permet de paralléliser les communications se trouvant sur des lignes différentes.

La communication entre deux segments bus se fait par l'intermédiaire du pont. L'accès à ce dernier implique un coût important en termes de latence et devient un goulot d'étranglement lorsque le nombre d'éléments communicants augmente.

### 4.4. Connexion par Réseau de Commutateurs :

Un réseau sur puce (NoC pour « Network on Chip ») est composé d'un ensemble de commutateurs (routeurs) connectés par des liens de communications (fils)**[SCH 06]**.

Cette structure apporte une solution aux problèmes rencontrés dans les

interconnexions actuelles et semble s'adapter aux systèmes futurs.

Contrairement aux bus, le NoC supporte les communications concurrentes et offre une bande passante plus large car celle-ci n'est pas partagée. De plus, cette architecture est flexible, facilement extensible, et offre la possibilité de réutiliser les IPs. Néanmoins, cette solution présente des coûts de mise en œuvre importants et manque d'outils de conception et d'aide à la décision.

### 4.5. Comparaison entre les Types d'Interconnexion :

Nous avons cité les différentes architectures d'interconnexions dans les systèmes sur puce ainsi que leurs avantages et leurs limites qui ont conduit aux solutions actuelles, notamment aux réseaux sur puce.

Le tableau ci-dessous **(Tableau 1)** résume les caractéristiques de ces structures d'interconnexion.

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

**++ : Très bon** **+ : Bon** **--: Très mauvais** **-: Mauvais**

**Tableau 1:**Comparaison entre les différentes structures d'interconnexion dans les SoCs**[RIS 04].**

## 5. Réseau sur puce (NOC) :

Du succès du SoC (System-on-Chip) est né le NoC (Network-on-Chip). Le premier de ces concepts consiste à placer au sein d'une même puce plusieurs fonctions jusque-là gérées par des puces dédiées et regroupées sur une carte. Créée en début 2003, la start-up française Arteris propose la notion de réseau à l'intérieur de la puce, qu'elle appelle NoC **[BOU 12]**.

## 6. Composants d'un Réseau sur Puce (NoC) :

Un réseau sur puce est composé de routeurs, d'interfaces réseaux et des liens de communications (**Figure 5**).

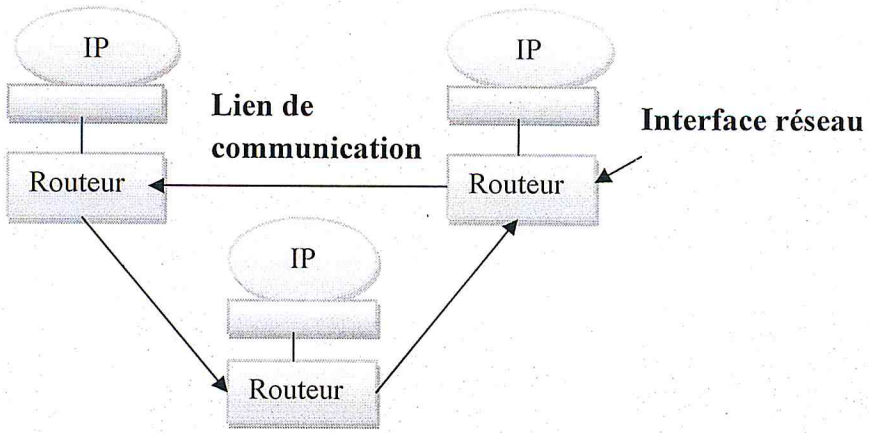


**Figure 5 :** Structure d'un réseau sur puce **[SCH 06].**

### 6.1. Routeurs :

Le rôle principal d'un nœud de routage est d'acheminer les données d'une source à une destination. Il est constitué de:

- Files d'attente pour stocker les paquets qui transitent dans le réseau.
- Un commutateur qui connecte les files d'entrées aux ports(ou files) de sortie.
- Une unité de routage et d'arbitrage qui assure la fonction d'aiguillage et gère les situations de conflits **(Figure 6)**

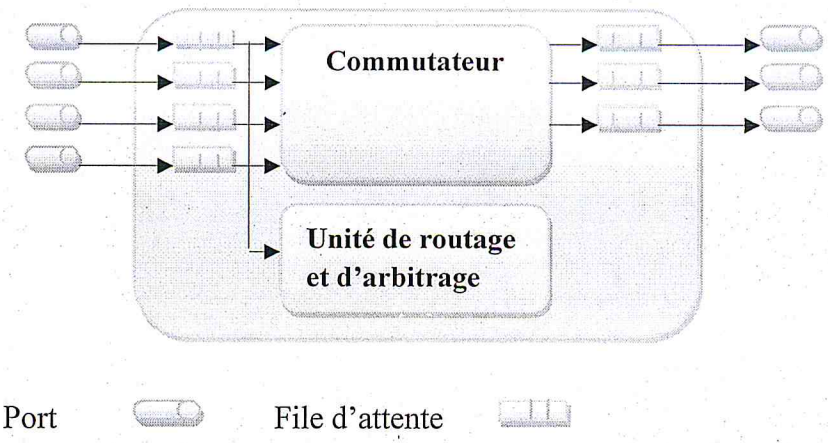


**Figure 6 :** Structure d'un routeur d'un réseau sur puce **[DEL 07].**

Chaque routeur est connecté à une ressource de l'application. Celle-ci exécute une fonction particulière IP (Processeur) et elle est logée dans une tuile **(Figure 7)**.

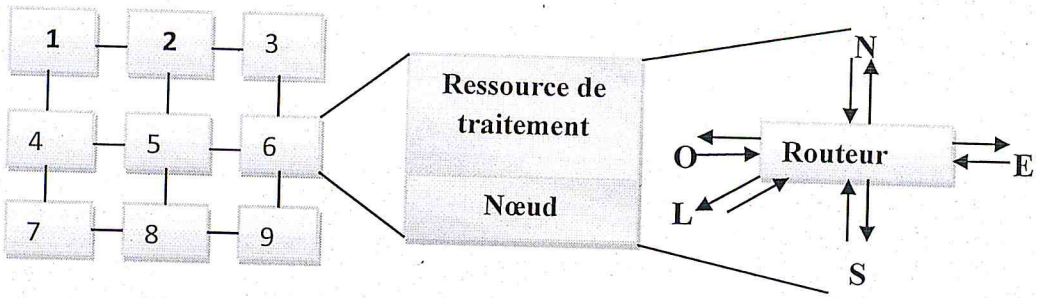


**Figure 7:**Structure d'une tuile d'un NoC **[HOU 07]**.

### 6.2. Interface Réseau :

Une interface réseau traduit les messages envoyés par les modules en requêtes compréhensibles par l'interconnexion **[LEM 06].**Ceci permet de séparer matériellement la fonction « calcul » de la fonction « communication » et de réaliser des IPs réutilisables **[TRA 08]**. Elle est composée de deux parties. Une partie back-end liée au routeur et une partie front-end connectée à la ressource de traitement. Le second bloc implémente des standards qui réalisent l'adaptation des protocoles de communication entre la ressource et le routeur **[OCP]**.

### 6.3. Liens de Communications :

Un lien de communication physique permet de connecter les nœuds de routage et de transférer des données entre eux **[MEL 05]**.

## 7. Caractéristiques des Réseaux sur Puce :

Un réseau sur puce est défini par une topologie, un mode de commutation, une stratégie de routage et une politique de contrôle de flux. Dans ce qui suit, quelques topologies des réseaux sur puce sont présentées.

### 7.1. Topologie :

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 Maillée**

La maille à 2 dimensions est la topologie la plus employée dans les réseaux sur puce. Ceci est dû à la facilité de son implémentation **(Figure 8.a)** et la simplicité des structures de ses routeurs **[HOU 07]**.

- **Topologie en Tore**

C'est une topologie dérivée de la structure 2D possédant la particularité d'un repliement des bords extérieurs sur eux-mêmes **(Figure 8.b)**. Elle offre ainsi une bande passante légèrement supérieure à celle de la 2D maillée mais elle est plus complexe à implémenter et très difficile à tester **[DEL 07]**.

- **Topologie en Anneau**

La topologie en anneau est une structure facilement intégrable sur silicium **(Figure 8.c)**. Cependant, elle n'est pas extensible car ses performances se dégradent à mesure que le nombre d'IPs connectées augmente **[DEL 07]**.

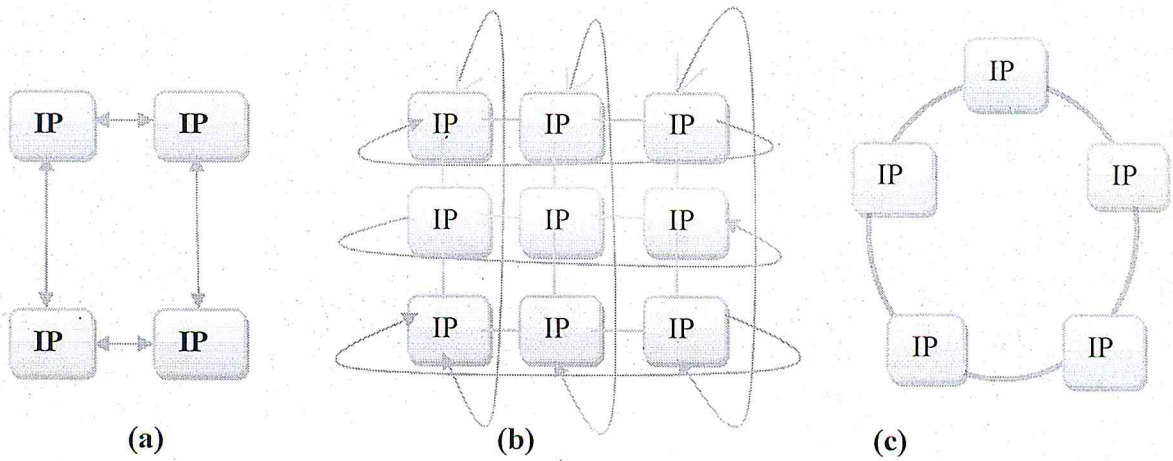


**Figure 8 :** Topologies 2Dmaillée, Tore et Anneau **[DEL 07]**.

### 7.2. Modes de Commutation

Les modes de commutation définissent la stratégie d'allocation des différentes ressources du réseau sur puce afin d'acheminer des données **[TRA 08]**. Deux modes de commutation sont utilisés dans les réseaux sur puce: la commutation de circuit et la commutation de paquets.

- **Commutation de Circuit**

Ce mode de commutation consiste à établir un circuit dédié au sein du réseau pour chaque paire émetteur/récepteur **[DEL 07]**. Ceci garantit une large bande passante entre les deux ressources et augmente les performances du système lorsque les données échangées sont de taille importante. Cependant, ce mode de commutation est très pénalisant en termes de ressources car celles-ci sont réquisitionnées tout au long du transfert.

- **Commutation de Paquets**

Ce mécanisme de commutation consiste à découper un message en plusieurs paquets avant d'être envoyé. Dans les réseaux sur puce, un paquet est décomposé en plusieurs FLITs (« Flowcontrol un IT », Adaptation sur silicium). Chaque flit est stocké dans une file d'attente puis transmis sur la voie appropriée. A la réception, le paquet est reconstitué à partir des flits reçus.

Ce mode de commutation permet un meilleur partage des éléments du réseau car les voies sont libérées dès qu'un flit est envoyé. Ceci réduit la latence et améliore les performances du système **[LEM 06]**.

Les modes de commutations de paquets les plus utilisés dans les réseaux sur puce sont:

- ✓ **Store and Forward** (Stocker et propager): avec cette stratégie, tous les flits constituant un paquet sont stockés avant d'être transmis. Le but de ce stockage est le contrôle des paquets envoyés.
- ✓ **Wormhole** (Trou de ver): ce mode de commutation réduit la latence du système car il n'exige pas que tout le paquet soit stocké avant d'être envoyé. Dès qu'une voie est libre un flit est transmis au routeur destinataire.

### 7.3. Algorithme de Routage

Un algorithme de routage définit le chemin que doit emprunter un paquet pour atteindre sa destination. Il doit éviter les situations d'inter blocage tout en optimisant l'utilisation des liens de communications (optimisation de la consommation de l'énergie) **[RIS 05]**.

Une des techniques de routage les plus utilisées dans les structures de réseau sur puce est X-Y.

- **Routage Ordonné X-Y**

L'algorithme XY est un algorithme déterministe **[MCK 93]**. 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 9** illustre le fonctionnement de cette technique.

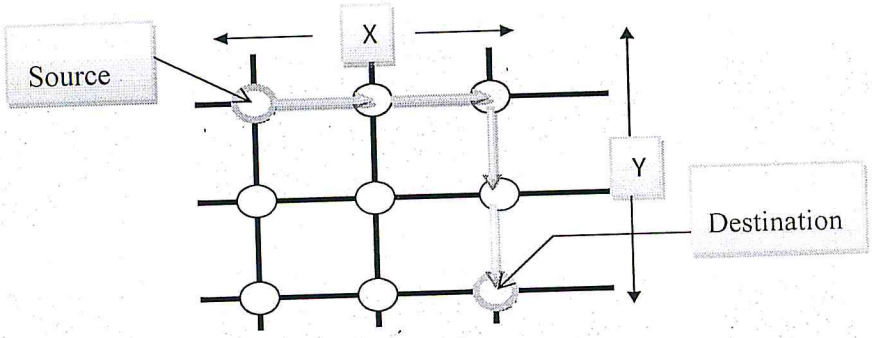


**Figure 9** : Routage Ordonné X-Y. **[MCK 93]**

### 7.4. Contrôle de Flux

Le contrôle de flux est un ensemble de mécanismes qui évitent la surcharge du réseau et régulent le trafic. Afin de réaliser ces fonctions, des signaux de requêtes et d'acquittement sont utilisées par les routeurs. Les deux stratégies de contrôle de flux présentées sont le Handshake et le Credit-Based. **[GAP]**

- **Handshake :**

Lorsqu'un routeur envoie une donnée, il est en attente d'un acquittement qui lui permet de reprendre ses transactions. Ce mécanisme est simple à mettre en place mais il nécessite au moins deux cycles d'horloges pour effectuer un transfert. Ceci augmente la latence du système et dégrade ses performances.

- **Credit-Based :**

p Avec ce mécanisme de contrôle de flux, les données sont envoyées jusqu'à ce que les files d'attente du routeur récepteur soient saturées. Lorsque celles-ci se libèrent, le routeur l'indique en envoyant le signal « Credit ». La complexité des signaux échangés avec cette stratégie implique une augmentation de la consommation de l'énergie . Néanmoins, il est simple à implémenter et il améliore les performances (en termes de bande passante et de débit) du système car les transferts de données ne nécessitent qu'un seul cycle d'horloge.

## 8. Quelques Nocs académiques :

### 8.1. Hermès :

Le réseau Hermès, développé par l'équipe de **F. Moraes [MOR 03]** est un réseau sur puce à topologie maillée 2D avec commutation de paquets (Wormhole). Un buffer sur les ports d'entrée, un arbitrage de type « Round-Robin » et un algorithme de routage gérant les conflits permettent d'éviter les situations d'inter blocage dans ce réseau.

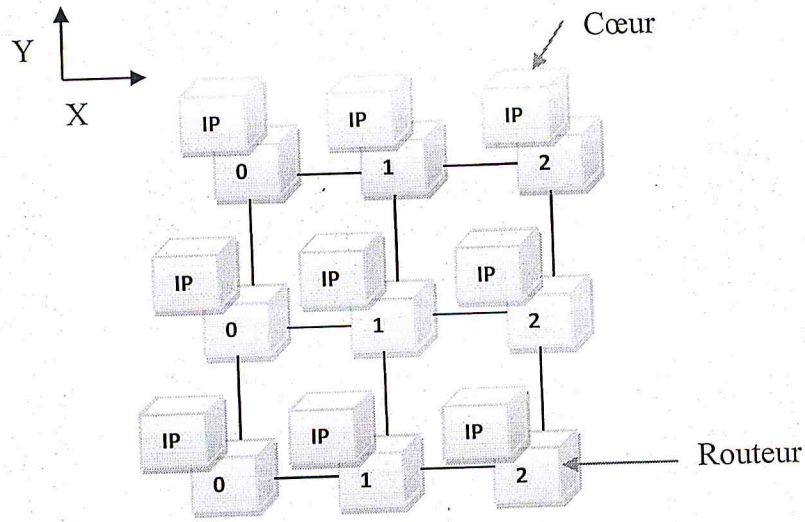


**Figure 10 :** NoC Hermès 3*3**[RIS 05]**.

### 8.2. XPIPES:

Une architecture avancée de Noc pour atteindre une meilleur performance et communication fiable pour des multiprocesseurs sur puce. Il se compose d'une bibliothèque des soft macros (commutateurs, interfaces de réseau et liens) qui sont

des composants concierges et réglables de sorte que le domaine spécifique est les architectures hétérogènes.

Les cœurs reliés entre eux par commutation de paquet (Wormhole) et effectués par un algorithme de routage statique appelé « Street Sign Routing ».L'interface réseau de Xpipes utilise le protocole de communication OCP (point-à-point).
XPIPES utilise des liens en pipeline, semblable à un registre à décalage dans des opérations réalisées en divisant les fils dans les segments du transfert actuel des flits.
XpipesCompiler est un outil qui automatiquement instancié pour adapter les besoins du client. **[ACR 09]**

## Conclusion

Le réseau sur puce est un nouveau paradigme d'interconnexion inspiré des réseaux informatiques. Il est composé de routeurs communicant entre eux via des liens physiques. Une interface réseau permet d'adapter les protocoles de communications des routeurs avec ceux des IPs sur différentes couches du réseau.

L'architecture d'un réseau sur puce peut être disposée de différentes manières. La topologie maillée à 2 dimensions est la plus répandue pour sa facilité de mise en œuvre et la simplicité de ses protocoles de communication.

Les informations transitent dans le réseau selon une commutation de circuit ou bien de paquet. La commutation de circuit garantit une latence et un débit précis. Par contre, une commutation de paquet ne garantit aucun service mais offre une meilleure utilisation des ressources du réseau.

Dans un réseau sur puce, le flux est géré selon deux stratégies. La stratégie Handshake basée sur une communication synchrone, et la stratégie Credit-Based qui permet d'envoyer des données de manière asynchrone.
Il existe plusieurs réseaux sur puce universitaires. Cependant, aucune solution n'a été Commercialisée. Ceci est dû à la complexité du processus de conception et le coût qu'il présente.

Le chapitre suivant cite les différentes phases de conception et les outils qui permettent de simplifier ce procédé et traite la phase de mapping et sa résolution.

*Chapitre II*

# Conception des Réseaux sur puce

## Introduction

Un réseau sur puce est une architecture qui permet d'interconnecter différents modules dans un système sur silicium. Il présente une réponse aux limites des solutions de communication actuelle mais sa conception est complexe et délicate. Afin d'y remédier, il est nécessaire d'automatiser les différentes étapes grâce à des méthodes et des outils.

Dans ce chapitre nous nous focalisons sur la phase de mapping et les différentes méthodes pour sa résolution et enfin quelques proposition tirées de la littérature sont citées et comparées.

## 1. Phases de conception d'un réseau sur puce :

Concevoir un réseau sur puce efficace qui satisfait les contraintes de performance de l'application est un processus complexe. Il passe par plusieurs niveaux d'abstraction partant de la modélisation haut-niveau de l'application jusqu'à l'implémentation physique du layout.


La conception d'un NoC comporte les phases suivantes **[NIC 10]** :

- Modélisation du trafic de l'application ;
- Synthèse de la topologie NoC de l'application ;
- Assignation des tâches aux nœuds de l'application ;
- Ordonnancement des tâches dans chaque nœud (se fait lorsque le nœud est un processeur) ;
- Mapping des nœuds de l'application sur la topologie ;
- Allocation des chemins de routage et réservation des ressources ;
- Vérification des performances du système ;
- Développement des modèles de synthèse et simulation.

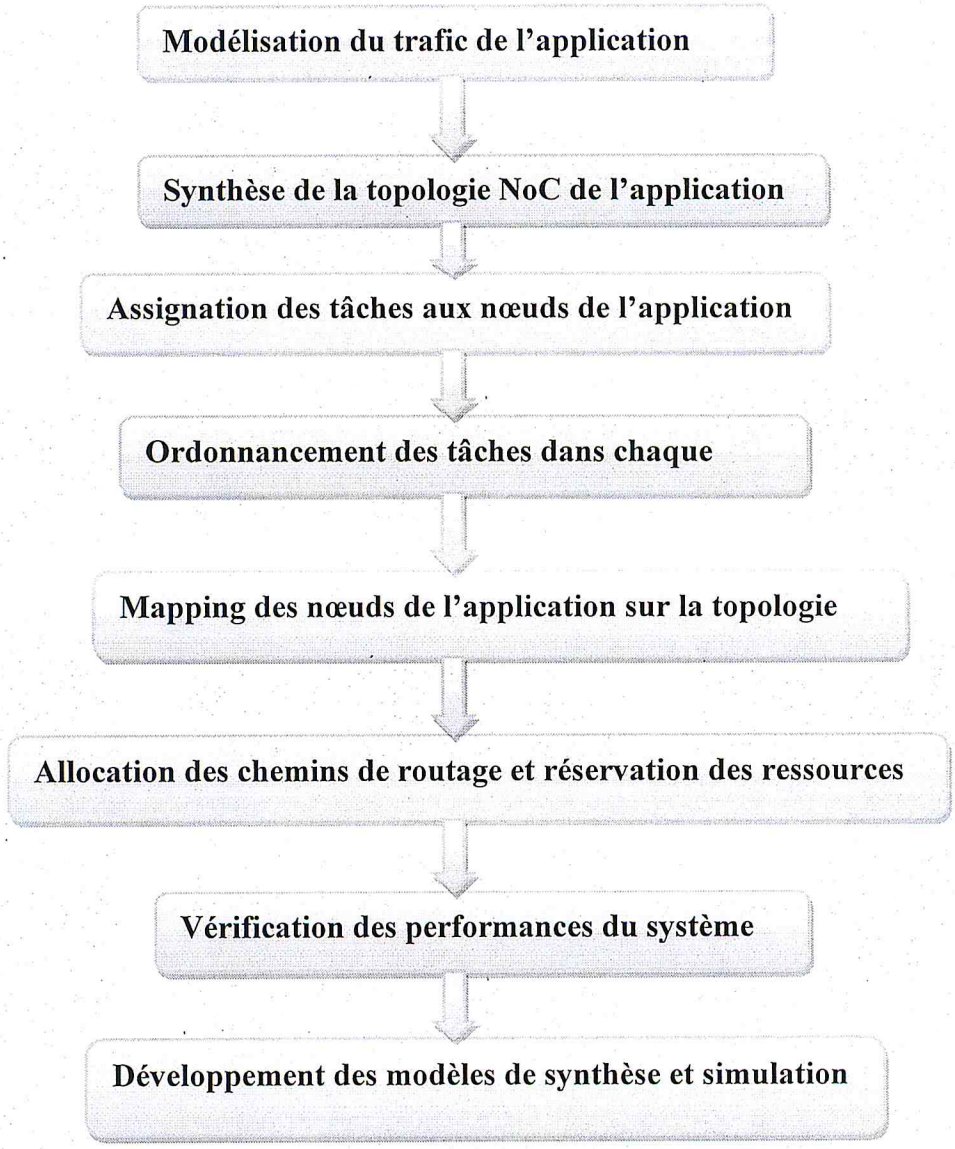


**Figure 11:**Schéma de conception d'un réseau sur puce.

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

Quelques outils qui automatisent différentes phases de conception sont présentés dans ce qui suit.

### 2.1. SUNMAP

L'outil SUNMAP **[MUR 04]** est utilisé dans le flot de conception du réseau sur puce QNoC. Il effectue la génération d'une topologie pour une application donnée afin de minimiser l'utilisation des bandes passantes et la surface de la puce. Pour ce faire, il exécute le mapping de l'application donnée sur différentes topologies et sélectionne celle qui fournit les meilleurs résultats en cout de communication.

### 2.2. NoCExplorer :

NoCExplorer **[ART]** est en environnement conçu par l'entreprise française Arteris. Il permet d'explorer l'application et de capturer ses besoins en termes de communication entre les blocs IPs (bande passante essentiellement) pour analyser les différentes topologies pouvant satisfaire ces contraintes.

### 2.3. MAIA :

MAIA **[GAP]** est un environnement open source qui permet de générer les fichiers de description d'un réseau sur puce à partir des paramètres introduit par l'utilisateur. Ces fichiers sont utilisés pour réaliser la simulation du trafic et l'évaluation des performances du réseau.

### 2.4. 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.

## 3. La phase de mapping :

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.

### 3.1. Définition de mapping

Le mapping d'une application composée de 'n' tâches communicantes sur une architecture parallèle disposant de 'p' processeurs est équivalent à la recherche de toutes les applications de T-->P où T représente l'ensemble des tâches et P l'ensemble des processeurs .il ya alors $P^n$ cas à étudier.

L'étude de tous ces cas et même leur énumération ne peut être envisagée dès que n ou p s'écarte de l'unité ; c'est-à-dire dès que le programme ou l'architecture peut être qualité de massivement parallèle.

Le problème est d'abord de trouver parmi ces $P^n$ cas, au moins un placement possible (c'est-à-dire permettant l'exécution du programme compte tenu des contraintes matérielles imposées par l'architecture). On peut alors se poser le

problème du choix d'un meilleur placement parmi tous les placements possibles **[AND88]**.

### 3.2. Problème de mapping :

Le problème de mapping est divisé en deux catégories : le mapping des calculs (tâches) et le mapping des données.

Le premier type nous permet de trouver sur quel processeur doit s'exécuter quelle tâche et à quel moment. Le deuxième type de placement doit déterminer un accès efficace et rapide aux données, avec si possible parmi les chemins d'accès, le plus court possible en prenant en considération les caractéristiques de la topologie cible telle la bande passante. Donc on ne peut faire un bon placement si on ne prend pas en considération tous les aspects des tâches. Adopter directement des solutions proposées dans les systèmes distribués classiques n'est pas conseillé pour les architectures MPSoC.

Ces dernières sont globalement hétérogènes et ont quelques éléments réguliers homogènes.

### 3.3. Types de Mapping :

Le mapping peut être effectué avant l'exécution de l'application ou durant cette phase. Ainsi, deux types de mapping sont distingués **[CAR 09]**.

❖ **Mapping statique**

Le mapping statique est adapté à des plateformes spécifiques. Toutes les IPs sont affectées aux tuiles de l'architecture avant que l'application ne soit exécutée.
Le recuit simulé est une technique de mapping statique. **[CAR 09]**

❖ **Mapping dynamique**

Le mapping dynamique est effectué durant l'exécution de l'application. Une tâche de l'application peut être ajoutée, supprimée ou replacée. **[CAR 09]**

Le mapping dynamique peut offrir de meilleures performances et apporter des avantages aux systèmes tels que la tolérance aux erreurs et la réduction de la consommation d'énergie. Néanmoins, il est complexe et difficile à tester.

### 3.4. Résolution du problème de Mapping :

Le problème de mapping est considéré comme un problème d'affectation quadratique . Placer m tâches sur n tuiles s'avère être un problème NP-difficile

**[AND88]**. Il existe $m^n$ solutions possibles et les énumérer ne peut être envisagé lorsque m ou n devient trop grand.

Pour résoudre ce problème, deux classes de méthodes sont utilisées. Il s'agit de la classe des Méthodes exactes et la classe des méthodes approchées **[DEL 03]**.

➢ **Méthodes exactes**

Les méthodes exactes garantissent de trouver l'ensemble des solutions optimales.

Cependant, leur approche par énumération accroit leur temps d'exécution de manière exponentielle en fonction de la taille du problème **[DEL 03]**.

➢ **Méthodes approchées**

Les méthodes approchées offrent l'alternative de disposer d'une solution de bonne qualité (optimale ou proche de l'optimale) en un temps réduit. Ceci est avantageux lorsque la taille du problème à traiter est importante. Ces méthodes peuvent être classées en différentes catégories **[FRE 00]**.

- Méthodes constructives telles que les algorithmes gloutons et la méthode Pilote.
- Recherche locale comme la recherche tabou et le recuit simulé.
- Méthodes évolutives telles que les algorithmes génétiques et les fourmis artificielles.
- Réseaux de neurones (Modèle de Hopfield-Tank, machine de Boltzmann, réseau auto-adaptatif, réseau élastique).
- Heuristiques Bayésiennes (optimisation globale, optimisation discrète).
- Superposition (perturbation des données, perturbation des paramètres d'une heuristique).

### 3.5. Quelques techniques proposées pour la résolution du problème de Mapping :

Afin de résoudre le problème de mapping, différentes techniques ont été élaborées.

- **Energy and Performance Aware Mapping (EPAM)**

EPAM a été proposé de trouver des correspondances juridiques avec moins totale l'énergie de communication. L'optimiseur de l'EPAM était basé sur des arbres de

recherche branch-and-bound. Les nœuds internes des arbres représentent les résultats de mapping incomplète et les nœuds feuilles représentent toutes les solutions possibles.

La **Figure 12** donne un exemple. Depuis un nœud feuille a le meilleur coût de l'énergie, 80 mJ, les autres nœuds avec des coûts supérieure à 80mJ sont jetés. L'EPAM a adopté les calculs de la haute et limite inférieure pour chaque mapping incomplète de décider s'il faut jeter des sous-arbres afin d'améliorer efficacité. Pour un nœud interne, au moins une complète solution après le mapping incomplète actuel consomme énergie inférieure ou égale à la limite supérieure, tandis que tous les résultats de le mapping complets doivent consommer plus d'énergie que la borne inférieure. En outre, EPAM a adopté files d'attentes prioritaires avec la taille limitée afin de mieux contrôler la croissance de la recherche arbres. **[PAM 05]**

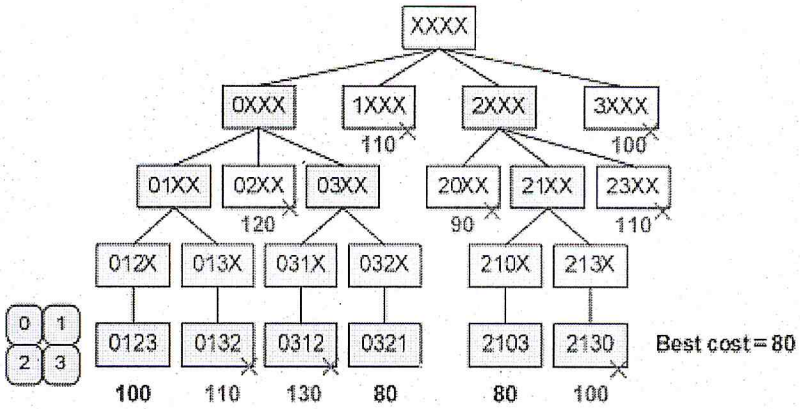


**Figure 12 :** Une illustration de l'arbre de recherche adopté en EPAM **[PAM 05]**.

- **CGMAP**

Est une nouvelle technique qui trouve le mapping des sommets d'un graphe de tâches sur les carreaux d'une architecture Noc à base de maille. Elle combine les deux idées des algorithmes génétiques et les systèmes chaotiques et profite des avantages de chacune. L'intérêt d'utiliser un algorithme génétique est de pouvoir explorer l'espace des solutions réalisables rapidement. Les séquences chaotiques quant à elles affinent la recherche. Ceci évite la convergence prématurée et fournit de meilleurs résultats que ceux obtenus en utilisant les algorithmes génétiques classiques compte tenu des indices de performance tels comme le rapport de la distance de saut, la consommation d'énergie, et la latence **[MEH 07, FAG 09]**.

- **Onyx**

Une méthode heuristique pour le mapping des cœurs sur plate-forme de maillage avec moins de complexité. Il minimise le nombre de sauts entre les cœurs IP, conduisant à l'amélioration de la consommation d'énergie et d'autres paramètres de performance.

Onyx définit les quatre mouvements pour déterminer la priorité d'une tuile pour un mapping sur un chemin sous forme losange (Lozenge_ShapePath). Après chaque mapping d'une tâche à une tuile le losange tourne au prochain mouvement. Grace aux quatre mouvements, on peut détermine l'ordre de mapping jusqu'à la fin. **[JAN 09]**

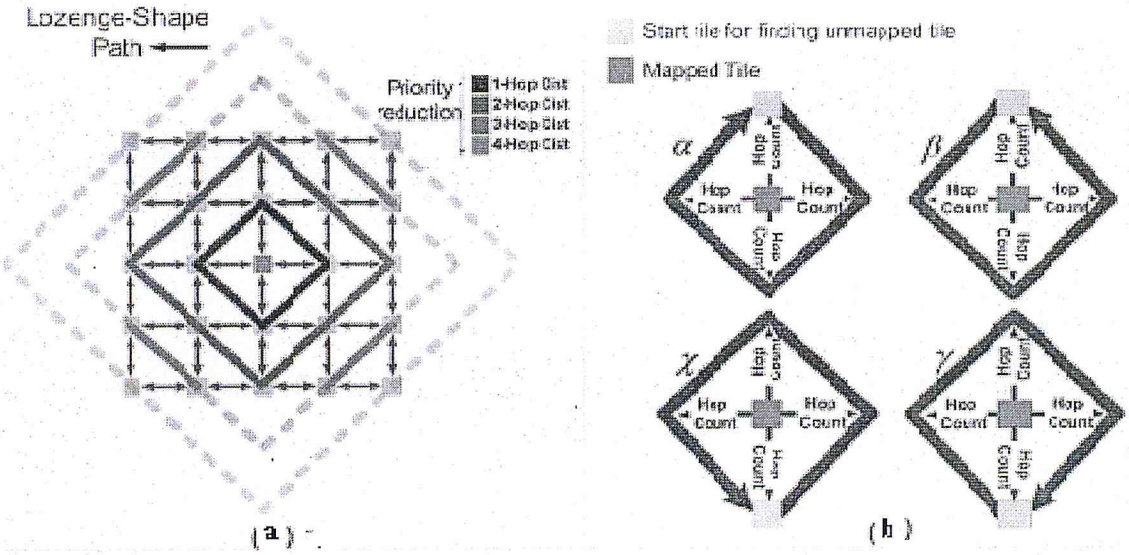


**Figure 13 :**(a) Le concept de Lozenge_ShapePath, (b) Définition des quatre mouvements **[JAN 09].**

- **SPIRAL**

La technique SPIRAL consiste à placer de façon optimale les cœurs d'une application donnée sur une architecture de réseau sur puce en topologie 2D maillée. Les métriques à optimiser sont le coût des communications, la complexité de l'algorithme et le temps écoulé pour l'exécution du système.

Cette méthode affecte les tâches de l'application aux ressources de l'architecture selon la forme d'une spirale (**Figure 14**) en démarrant du centre pour atteindre les tuiles frontières. Cette technique minimise les coûts de communication de manière considérable et son temps d'exécution est faible **[MEH 07]**.

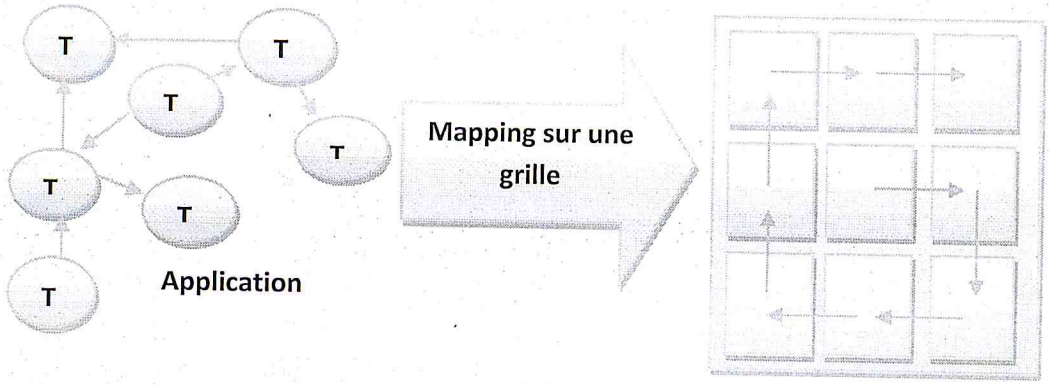


**Figure 14 :** Mapping d'une application sur une architecture avec la technique SPIRAL **[MEH 07]**.

- **GBMAP:**

Est un algorithme de mapping sur une architecture de réseau sur puce en topologie 2D maillée. GBMAP est une méthode heuristique basée sur la génétique à une meilleure performance et permet de réduire la taille de bande passante et minimise le cout de communication et la consommation d'énergie.

Dans cet algorithme, le chromosome est l'élément de base, la structure de ce dernier est comme une table bidimensionnelle de commutateurs (Switch), GBMAP utilise les opérateurs génétiques basés sur cette structure. Cette méthode basée sur des modifications aléatoire des différents échantillons sur la population et permet de choisir le meilleur entre eux. **[MAS 10]**

- **GMAP:**

Est un programme autonome qui a été conçu pour traiter les requêtes individuelles rapidement, avec pratiquement pas de temps de démarrage. Au lieu de pré chargement de totalité du fichier en mémoire on charge seulement son index d'oligomères.

GMAP regarde oligomère au besoin directement à partir du fichier. Parce que l'accès au fichier est plus lent que la mémoire, cette technique basée sur les algorithmes génétiques et utilise une stratégie d'échantillonnage conçu pour réduire le nombre de recherche d'oligomère nécessaire pour un mapping. **[LEI 03]**

Le tableau ci-dessous **(Tableau 2)** résume les techniques de mapping proposées :

| Algorithme | Méthode | Fonction objectif | Remarque | Année |
|---|---|---|---|---|
| **EPAM [PAM 05]** | Branch&bour | Optimisationc l'énergie | L'optimiseur de l'EPAM était basé sur des arbres de recherche | 2005 |
| **CGMAP [CAR 09]** | Algorithme génétique-Séquences chaotiques | Consom -mation d'énergie | Combine les deux idées des algorithmes génétiques et les systèmes chaotiques | 2007 |
| **ONYX [MAT 09]** | Heuristiques | Minimisation de nombre des sauts | Utilise l'idée de Lozenge_ShapePath pour détermine l'ordre total de mapping avec moins de complexité | 2009 |
| **SPIRAL [MEH 07]** | Algorithme génétique | Cout de commu -nication | Cette méthode affecte les tâches de l'application aux ressources de l'architecture selon la forme d'une spirale | 2007 |
| **GBMAP [MAS 10]** | Algorithme génétique | Consom -mation d'énergie | Cette méthode basée sur des modifications aléatoire des différents échantillons sur la population et permet de choisir le meilleur entre eux | 2009 |
| **GMAP [LEI 03]** | Algorithme génétique | Charge en mémoire | utilise une stratégie d'échantillon -nage conçu pour réduire le nombre de recherche d'oligomère nécessaire pour un mapping | 2003 |

**Tableau 2 :** Résumé des techniques proposées.

## Conclusion

La conception d'un réseau sur puce est une procédure très complexe. Le temps de mise sur le marché et les contraintes de performance du système exigent que celle-ci soit rapide et peu coûteuse. Afin d'y remédier, des outils ont été mis en place pour automatiser certaines phases.

Cependant, la plupart de ces outils ne sont pas gratuits et ne permettent pas de tester les techniques et les protocoles élaborés dans le cadre de conception d'un NoC (mis à part les méthodes sur lesquels ils sont basés).

Dans le chapitre suivant, on va présenter quelques algorithmes d'optimisation qui peut être utilisé pour résoudre le problème de mapping.

# *Chapitre III*

# ***Optimisation Multicritère***

## Introduction

Dans le monde industriel, généralement, les problèmes d'optimisation sont de nature multi- critères puisque plusieurs critères permettent de caractériser une solution. Pour les Noc, objet de notre étude, les critères les plus connus sont le temps de calcul, la consommation d'énergie, les dimensions de la puce, espace,…etc.

Satisfaire ces critères est notre objectif, ce qui revient à optimiser les performances du système par rapport à ces objectifs. En améliorant chacun sans pour autant détériorer les autres ; ce n'est pas évident à faire, là on est en train de faire de l'optimisation multicritère.

## 1. Problème NP-complet :

Les méthodes d'optimisation des problèmes monocritères (mono objectifs) ou multicritère (Multiobjectif) sont très variées. Certains des problèmes d'optimisation peuvent être résolus par des méthodes exactes simples, qui permettent de trouver à coup sûr la (ou les) meilleure(s) solution(s) ou solution(s) optimale(s). Mais la plupart des problèmes étudiés en optimisation appartiennent à la classe des problèmes NP-difficiles **[BAS 05]**. Cette classe rassemble des problèmes pour lesquels « on ne connaît pas d'algorithme exact rapide. Dont la résolution exacte n'est pas possible en un temps de calcul proportionnel à $N^n$, où N désigne le nombre de paramètres inconnus du problème, et n est un entier ». **[COL 02]**

## 2. Optimisation multicritère (Multiobjectif) :

La résolution d'un problème d'optimisation consiste à explorer un espace de recherche à l'aide des critères afin de maximiser (ou minimiser) une fonction donnée comme dans notre projet les critères sont : l'espace, l'énergie et la communication. En première approximation, on peut dire qu'une méthode déterministe est adaptée à un espace de recherche petit et complexe et qu'un espace de recherche grand nécessite plutôt une méthode de recherche stochastique (recuit simulé, algorithme génétique,...).

Résoudre un problème d'optimisation consiste à trouver la ou les meilleures solutions vérifiant un ensemble des critères définis par l'utilisateur. Un problème d'optimisation consiste à rechercher la valeur maximale ou minimale, appelée optimum global, d'une fonction F : S→ R. F est appelée la fonction objectif. Cette fonction est appelée fonction de coût, fonction d'adéquation ou « *fitness* ». Comme maximiser une fonction F est équivalent à minimiser la fonction –F, nous considérons dans ce manuscrit que les fonctions doivent être minimisées. L'approche qui combine les objectives sur une seule fonction est appelé **Fonction Agrégation**

$$\mathbf{F}=\sum_{i=1}^{nbobj}(w_i f_i) \quad \text{avec } \sum_{i=1}^{nbobj}(w_i)=1. \textbf{ [ROS 67]}$$

Dans un problème d'optimisation multicritère (Multiobjectif), il n'y a pas une solution optimale unique, mais un ensemble de solutions potentielles, car en général aucune solution n'est la meilleure vis-à-vis de tous les critères simultanément **[MAR 99]**. Actuellement il existe plusieurs méthodes d'optimisation. Ces méthodes varient selon leurs complexités et leurs champs d'application. En plus des méthodes exactes il existe d'autres classes pour les méthodes d'optimisation. Dans la partie suivante, nous étudierons ces classes.

## 3. Les algorithmes d'optimisation:

### 3.1. Heuristiques :

Une méthode approchée ou heuristique ***(heuristic, approximation method***), pour un problème d'optimisation, est un algorithme qui a pour but de trouver une solution réalisable, tenant compte de critères d'optimisation et des contraintes, mais sans garantie d'optimalité. On oppose les méthodes approchées aux méthodes exactes, qui trouvent toujours l'optimum, mais leur inconvénient est le temps de résolution.

Il existe un très grand nombre d'heuristiques selon les problèmes à traiter, et il est aisé d'inventer **[HAO 99]**.

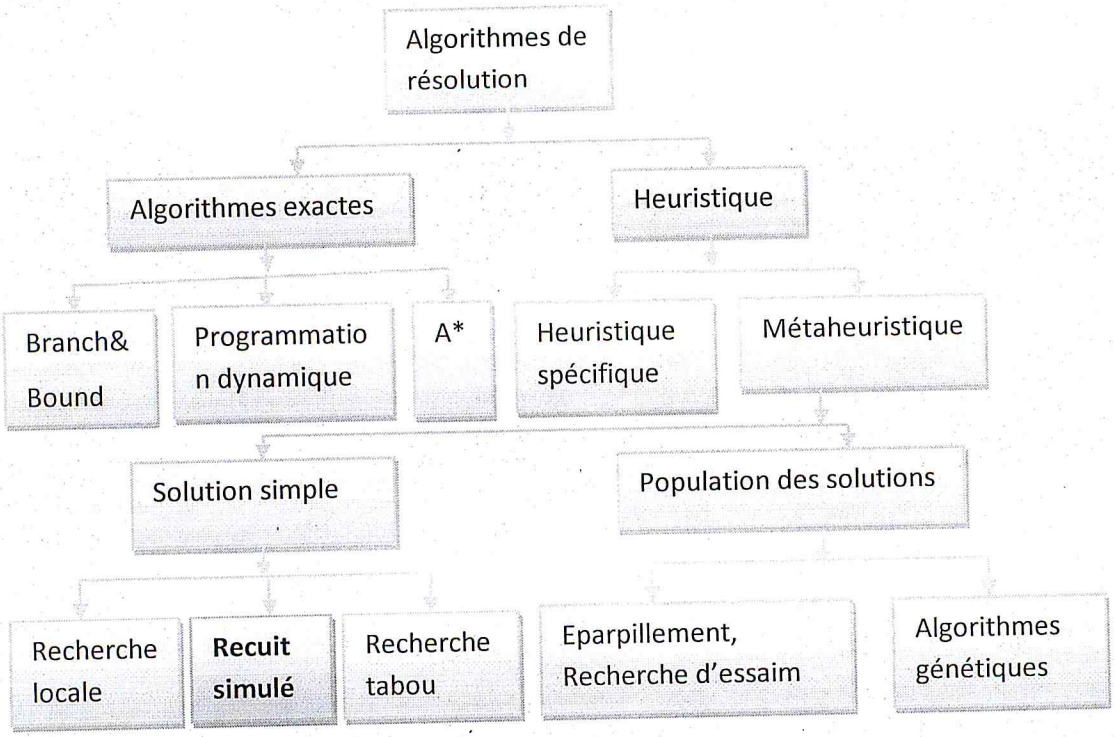


**Figure 15**: Classification des méthodes d'optimisation multicritère. **[ORA 09]**

### 3.1.1. Les métaheuristiques :

Les métaheuristiques, sont des méthodes générales de recherche dédiées aux problèmes d'optimisation difficile. Ces méthodes sont, en général, présentées sous la forme de concept **[SAD 99]**. Comme nous le verrons plus tard, elles reprennent des idées que l'on retrouve parfois dans la vie courante. Une métaheuristique est donc une méthode très générale, qui nécessite quelques transformations (mineures en général) avant de pouvoir être appliquée à la résolution d'un problème particulier. Elles sont des algorithmes d'optimisation (généralement de type stochastique) combinant plusieurs approches heuristiques.

Elles sont des techniques d'amélioration progressive d'une première solution. Elles incluent notamment le recuit simulé, les méthodes Tabou, et les algorithmes génétiques. Elles permettent de traiter des problèmes de grande taille, tout en obtenant des solutions excellentes, souvent optimales **[LAC 05]**.

#### a. Basée sur solution unique

➢ **Recuit simulé**

Inspirée du domaine de la métallurgie, cette méthode a été mise au point au début des années 1980 dans les laboratoires d'IBM par S. Kirkpatrick *et al.* **[KIR 83]**, et indépendamment par V. Cerny**[CER 85]**. Elle se base sur les procédés industriels de solidification des métaux. Il est en effet établi qu'un métal à température élevée doit être refroidi lentement (en opposition à la méthode dite de la ***trempe***) afin de laisser le temps aux atomes de s'agencer de manière optimale ce qui se traduit par un état d'énergie minimale du matériaux et donc une configuration stable le rendant plus résistant aux contraintes mécaniques. Ces propriétés thermodynamiques sont modélisées dès les années 1950 par N. **Métropolis** *et al.* **[MET 53]** qui mettent au point un algorithme basé sur une chaine de ***Markov*** utilisant une distribution de ***Boltzmann*** pour échantillonner l'espace de recherche. Cet algorithme a été repris plus tard par W.K. Hastings **[HAS 70]**, et généralisé à d'autres distributions, aboutissant à l'algorithme de ***Metropolis-Hastings***.

➢ **Méthode Tabou**

La méthode Tabou ***(tabusearch)*** a été inventée par F. Glover en 1986**[GLO 86]**. Le principe de cette méthode est à chaque itération le voisinage de la solution courante est examiné. L'algorithme enregistre la meilleure solution parmi les voisins, même si elle est moins bonne que la solution courante. L'acceptation des solutions moins performantes que la solution courante permet d'éviter de tomber dans un optimum local. Pour échapper de tourner dans un cercle entre plusieurs solutions, l'algorithme interdit le passage par des solutions récemment visitées. En pratique la méthode stocke dans une liste taboue ***T*** les attributs des dernières solutions visitées. Dans l'itération suivante, la meilleure nouvelle solution voisine enlève la solution la plus ancienne dans la liste (algorithme suivant). Dans d'autres cas, la méthode mémorise les mouvements réalisés plutôt que les solutions. Ensuite, on interdit les mouvements inverses. Cette technique est rapide et consomme peu de mémoire.

**b. Basée sur population des solutions:**

➢ **Algorithmes basés sur l'intelligence collective (colonies de fourmis) :**

Ces algorithmes fondés sur l'intelligence collective, proposés par Colonie **[FRA 12]** utilisent une analogie avec le comportement naturel d'une colonie d'insectes.

La démarche de l'optimisation s'apparente à des fourmis recherchant leur nourriture (i.e. la solution au problème proposé). Ces algorithmes s'appuient, comme les AG, sur une population d'individus. La recherche s'effectue selon deux étapes :

✓ les meilleures fourmis (i.e. celles ayant la meilleure valeur du critère à optimiser) effectuent une recherche locale autour de leur position.

✓ les autres effectuent une recherche plus globale en suivant les phéromones déposées antérieurement. Cette phase de suivi de phéromone comporte une procédure d'évaporation permettant de ne retenir, au fur et à mesure de l'exploration, que les traces les plus performantes.

➢ **Algorithmes génétiques :**

Les algorithmes génétiques **(AG)** sont également fondés sur une analogie entre un problème d'optimisation et un phénomène naturel, la loi de l'évolution énoncée par Darwin, où seuls les individus les plus forts d'une population survivent par suite de leur meilleure adaptation à leur milieu naturel. Dans un AG, après création artificielle d'une population de solutions viables pour un problème donné, la procédure fait ensuite évoluer les individus par croisement, mutation et sélection selon le critère à optimiser, jusqu'à fournir une population d'individus très bien adaptés au milieu naturel (i.e. une population de bonnes solutions pour le problème considéré).Ces algorithmes développés par Holland et Goldberg présentent un grand nombre d'avantages **[FRA 12] :**

✓ l'obtention en fin de recherche d'une population diversifiée de bonnes solutions. Si plusieurs critères sont à prendre en compte (optimisation multicritère, cadre de cette étude), cette diversité de solutions est un élément intéressant car elle

permet de disposer de possibilités plus variées (qu'une seule et unique solution optimale).

- ✓ il n'est pas nécessaire d'avoir des connaissances particulières sur les propriétés mathématiques du problème à traiter.
- ✓ ces algorithmes ont déjà montré leur robustesse et leur efficacité pour traiter des problèmes de conception optimale ou de remodelage d'ateliers de chimie fine.

## Conclusion

Les méthodes d'optimisation sont extrêmement nombreuses, elles sont basées sur des principes totalement différents, chacune explore et exploite l'espace de recherche selon des techniques qui lui sont propres.

Nous avons vue les Algorithmes génétique et Recuit simulé, Méthode Tabou, colonies de fourmis parce qu'elles sont extrêmement performantes dans de nombreux domaines. Ce sont des méthodes très efficaces lorsqu'il s'agit d'exploiter une zone de l'espace de recherche. D'autre part, elles s'adaptent assez bien au problème posé. Les concepts cités dans ce chapitre sont nécessaires à la compréhension de la démarche qu'on a apportée pour résoudre le problème de mapping dans les Nocs.

Le chapitre suivant présente le **Recuit simulé**, notre technique proposée pour résoudre le problème de mapping

# *Chapitre IV*

# ***Technique proposée***

## Introduction

Le mapping ou l'association «tâche/processeur », constitue l'étape la plus importante et la plus critique dans le flot de conception d'un réseau sur puce NoC, puisqu'elle participe d'une façon directe dans l'atteinte des critères voulus lors de la spécification des NoCs.

Le but de notre travail est de concevoir un algorithme basé sur le comportement d'heuristique-solution simple (recuit simulé) qui permet la résolution du problème de mapping d'une application sur une architecture cible, tout en essayant d'optimiser les critères. Une architecture étant un multiprocesseur hétérogène interconnecté par un réseau sur puce. Une application composée plusieurs tâches communicantes entre eux.

La majorité des travaux de résolution de ce problème présentent des méthodes d'optimisation monocritère, c'est-à-dire que ces méthodes ne permettent la vérification que d'un seul critère

En optimisation multicritère, la notion d'optimalité n'est pas aussi évidente. En effet, pour un même problème, une solution peut être bonne pour un critère et mauvaise pour un autre, pendant qu'une autre solution peut s'avérer moyenne pour chaque critère. Dans un cas comme celui-ci, il est difficile de dire objectivement laquelle est la meilleure. Ce qui cause un problème dans notre cas où on est face à trois critères qui sont : la consommation d'énergie, espace et la communication.

Donc notre approche apporte de nouveau aux anciens travaux en résolvant le problème de mapping sur architecture MPSoC avec une méthode d'optimisation multicritère.

Dans ce chapitre, on va présenter notre stratégie sous le mapping, basée sur la méthode recuit simulé avec plusieurs critères pour l'optimiser.

## 1. Problématique de mapping :

Le problème à résoudre dans sa globalité est un problème de mapping des tâches de l'application sur les processeurs de l'architecture de telle façon que les critères (objectifs) soient atteints.

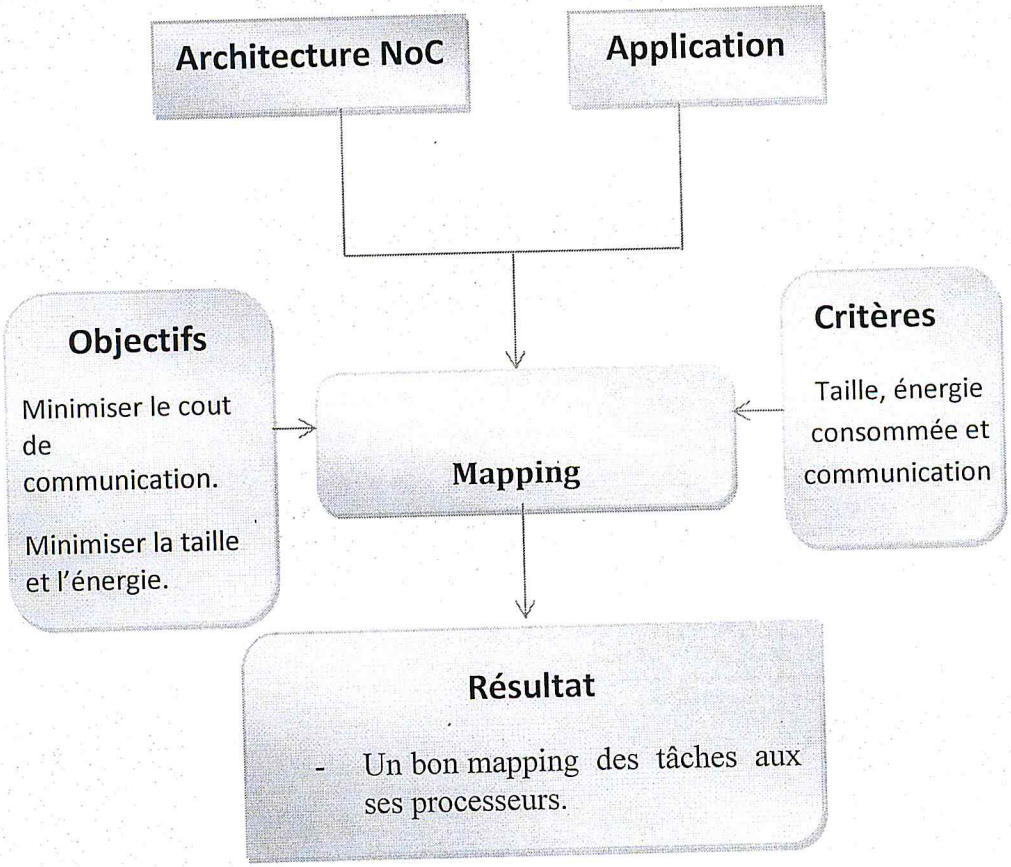


**Figure 16**: Description du problème de mapping

## 2. Définitions :

Pour présenter le problème de manière formelle, nous définissons dans ce qui suit les deux graphes utilisés. **[HOU 07]**

- **Définition1 (*graphe d'application*) :**

Le graphe d'application G (V, E) est un graphe orienté où chaque sommet (vertex) $v_i \in V$ numéroté correspond à une tâche de l'application. Les vertex sont liés entre eux par des liens orientés $e_{ij} \in E$ qui désigne la communication entre les vertex $v_i$ et $v_j$. Chaque lien est affecté d'un poids $w(e_{ij})$ qui représente le nombre de paquets

envoyés par $v_i$ à $v_j$. Chaque tâche a des propriétés qui sont les critères à optimiser (la communication, l'espace et l'énergie consommée).

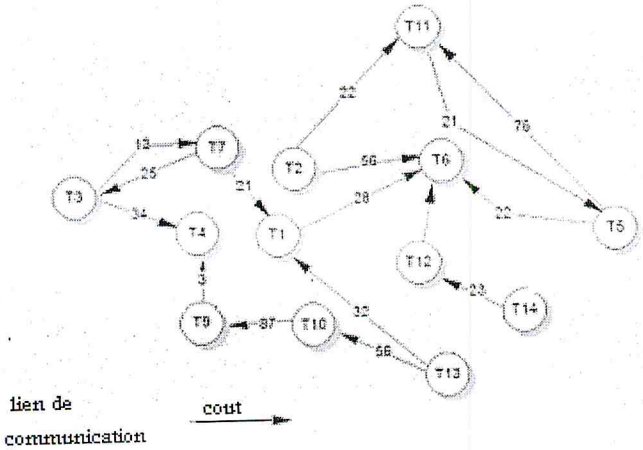


**Figure 17**: Modèle d'application.

- **Définition 2 (*graphe d'architecture*) :**

L'architecture du NoC A(T, L) est un graphe orienté où chaque sommet $t_i$ numéroté désigne une tuile de l'architecture du NoC, et chaque lien $l_{ij}$ définit un lien de communication physique entre $t_i$ et $t_j$. (Chaque processeur a les propres valeurs de la taille et d'énergie, on parle de l'hétérogénéité et n'accepte qu'une seule tâche à chaque mapping) (**Figure 18**).

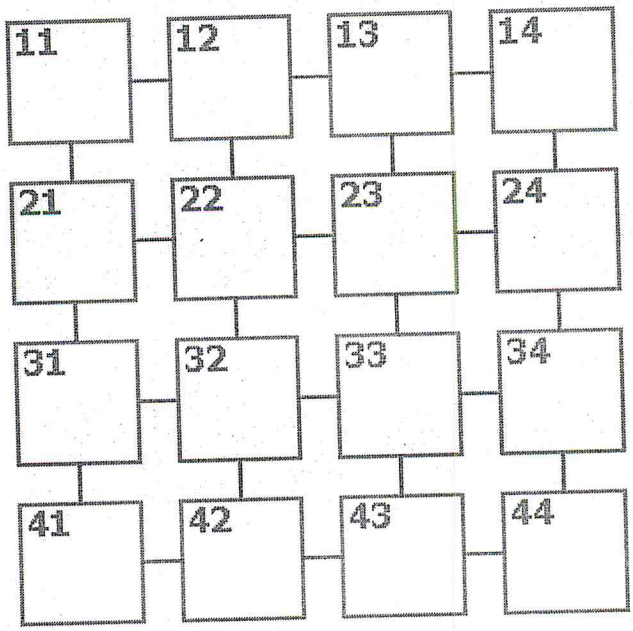


**Figure 18**: Modèle d'architecture.

- **Définition 3 (*Mapping*):**

Le mapping du graphe d'application G(V, E) sur le graphe d'architecture A(T, L) est illustré sur la **Figure 19** ci-dessous :

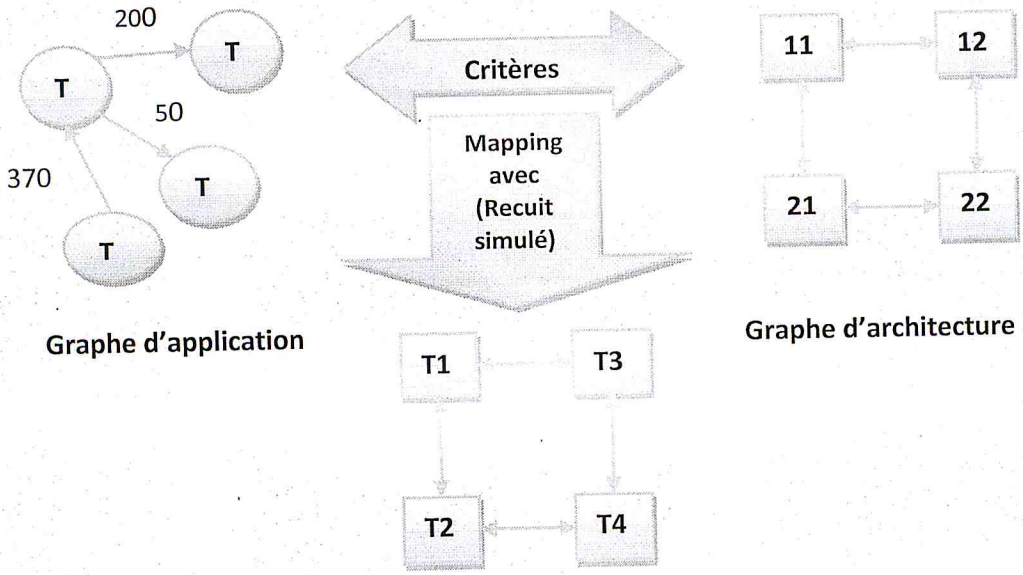


**Figure 19**: Mapping d'une application sur une structure de réseau sur puce 2D.

## 3. Les critères à optimiser :

### a) Cout de communication

Ce coût a un impact direct sur les performances du NoC. C'est le coût cumulé par la totalité communications effectuées sur le NoC afin d'acheminer tous les paquets entre les différentes IPs de l'application.

### b) Consommation d'énergie

On sait bien que l'énergie ou plutôt la consommation d'énergie est une notion très importante dans système embarqué portable, chose qui demande des efforts de conception plus élevés puisqu'on sait bien que le fonctionnement de notre système dépend de la durée de vie de la batterie. L'énergie consommée par une tâche doit être inferieur ou égale à celle disponible dans le processeur auquel elle a été affectée par le mapping.

### c) L'espace:

L'espace requis par une tâche doit être inferieur ou égale à celle disponible dans le processeur auquel elle a été affectée par le mapping.

## 4. Les formulations mathématiques :

Pour un G chaque sommet représente une tâche avec ses caractéristiques ou propriétés. Soit $T = \{t_1, t_2, \ldots\ldots, t_n\}$ l'ensemble des tâches représentées par l'ensemble des sommets $T_i$ dans G. $P = \{p_1, p_2, \ldots\ldots, p_n\}$ l'ensemble des processeurs représentés par l'ensembles des nœuds $P_i$ dans A.

- **La fonction map() :**

La fonction **map ()** qui consiste à affecter chaque tâche de l'application à un et un seul processeur de l'architecture NoC, est définie comme suit **[FAG 09]**

$$map\ (t_i) = p_j\ tels\ que\ \forall\ t_i\ \in T\ \ \exists\ p_j \in P$$

- **Les conditions de faisabilité :**

Il est nécessaire que le nombre de processeur composant l'architecture du réseau sur puce soit supérieur ou égale aux nombres des tâches à placer.

En d'autre terme, il faut que :

$$|P| >= |T|$$

$\mathbf{E_p}$ : désigne l'énergie disponible dans le processeur.

$\mathbf{E_t}$ : désigne l'énergie consommée par la tâche.

$$\mathbf{E_p >= E_t} \ldots\ldots\ldots\ldots\mathbf{(1)}$$

$\mathbf{M_p}$ : désigne l'espace disponible dans le processeur.

$\mathbf{M_t}$ : désigne l'espace requis par la tâche.

$$\mathbf{M_p >= M_t} \ldots\ldots\ldots\ldots\mathbf{(2)}$$

- **La fonction objective :**

Notre fonction objective est une agrégation des sous fonctions objectives (les critères) sous la forme suivante :

$$F = aF_1 + bF_2 + dF_3$$

Où :

$a$, $b$ et $d$ : désigne les facteurs d'agrégation et dans notre cas sont égaux à 1;

$F_1$ : désigne la fonction du cout de communication ;

$F_2$ : désigne la fonction du taille en mémoire ;

$F_3$ : désigne la fonction d'énergie consommée ;

**a. La fonction du cout :**

Lorsque le mapping est effectué, le but est de minimiser le cout de communication de l'application. De ce fait, les tâches qui communiquent le plus entre elles doivent être placées sur des processeurs proches.

La fonction de cout est ainsi donnée par l'équation suivante :

$$CC = \sum_{k=1}^{|E|} vl\left(d^k\right) \dots\dots\dots (3)$$

Où $d^k$ désigne le nombre minimale de saut entre les processeurs a et b dans l'architecture NoC.

Dans le graphe d'application, chaque communication est traitée comme un flux à chemin unique, représenté par $d^k$. La valeur $vl(d^k)$définit le cout total de la communication tel que :

$$d^k = e_{ij}, k = 1, 2, \dots, |E|, \forall\, e_{ij}\, E$$
$$map\,(v_i) = p_{ii}\; et\; map\,(v_j) = p_{jj}$$

Où :

$d^k$ : désigne la communication entre deux tâches $v_i, v_j$ ;

$vl(d^k)$ : désigne la communication entre deux tâches $v_i, v_j$ mappée dans les processeurs $p_{ii}, p_{jj}$, est calculé en de ***distance de Manhattan*** et $d^k$ :

$$\left(\left|X_{p_{ii}} - X_{p_{jj}}\right| + \left|Y_{p_{ii}} - Y_{p_{jj}}\right|\right) * d^k$$

**b. La fonction de l'espace :**

Avant d'affecter chaque tâche à un processeur, on cherche dans le NoC cel qui a une espace supérieur plus proche que celle de tâche, pour minimiser la fonction d'espace.

$m_i$ : désigne l'espace de la tâche i ;

$M$ : désigne l'espace totale, elle est calculée de la manière suivante :

$$M = \sum_{i=1}^{nbTache} (m_i) \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots (4) ;$$

**c. La fonction d'énergie :**

Avant d'affecter chaque tâche à un processeur, on cherche dans le NoC cel qui a une énergie supérieur plus proche que celle qui consommée par une tâche pour minimiser la fonction d'énergie.

$ec_i$ : désigne l'énergie consommée par la tâche i ;

$EC$ : désigne la consommation d'énergie totale, en prenant en considération la consommation des processeurs, elle est calculée de la manière suivante :

$$EC = \sum_{i=1}^{nbTache} (ec_i) \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots (5)$$

Après la définition des trois fonctions, la nouvelle forme de notre fonction objective est comme suit :

$$F = C + CC + M \dots\dots\dots\dots\dots\dots\dots (6)$$

## 5. Présentation de l'approche recuit simulé :

On construit tout d'abord une solution initiale dont la valeur de cout est l'***énergie*** $E = E_0$. On fixe une valeur de ***température*** $T = T_0$ suffisamment « élevée ». On sélectionne ensuite une solution candidate dans le voisinage de cette solution initiale, dont on détermine l'énergie $\mathbf{E_n}$. Si la variation $\Delta E = E_n - E$ est négative, on

accepte la solution candidate comme solution courante. Si elle est positive ou nulle, elle est acceptée avec une probabilité $e^{-\Delta E/T}$. Cette loi appelée ***règle de métropolis*** (issue des travaux de modélisation thermodynamique mentionnées ci-avant) emploie la variable de température : on constate que plus une solution médiocre (comparativement à la solution courante) aura de chances d'être néanmoins acceptée. C'est ce facteur qui permet à la méthode de recuit simulé de maintenir l'exploration de l'espace de recherche, même si un optimum local est atteint. **[ZHE 13]**

Une illustration de la méthode est présentée sur la (**Figure 18**).

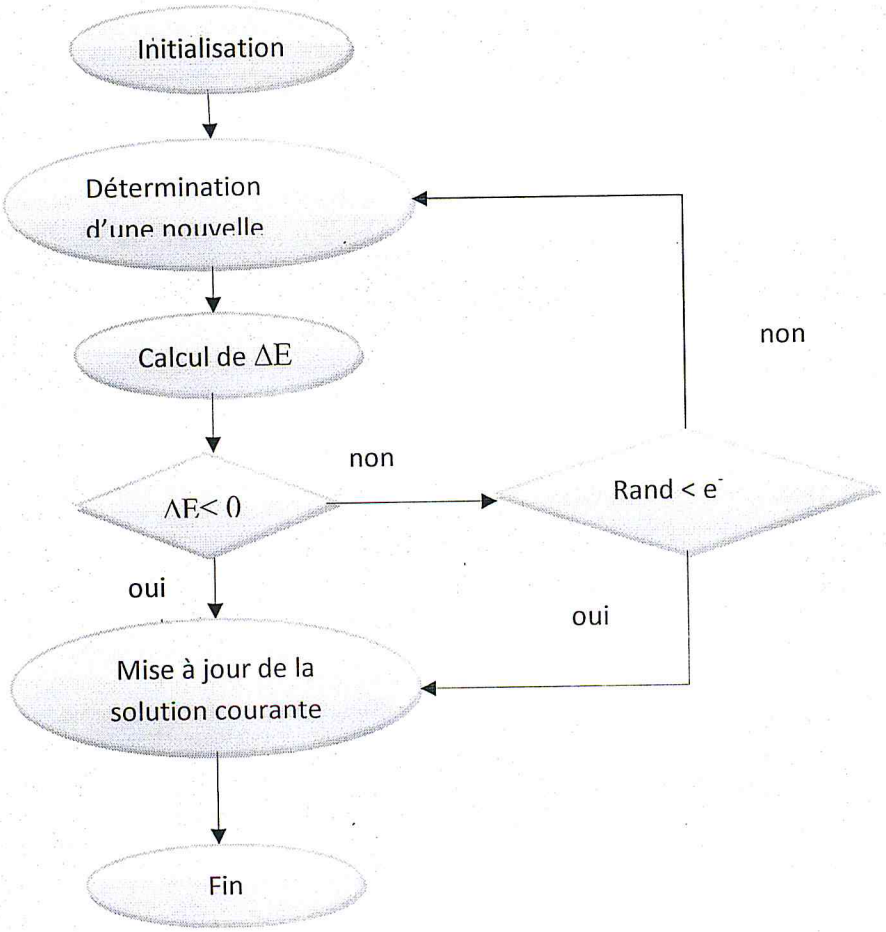


**Figure 20**: Présentation de l'approche recuit simulé.

## 6. Adaptation de l'approche au problème :

### 6.1. Les paramètres de l'algorithme :

**T :** est un paramètre fictif désigne la température du système;

**k :** désigne le nombre d'itération de l'algorithme ;

**rand :** désigne la valeur aléatoire qui va comparée avec la règle de métropolis ;

### 6.2. Le pseudo code de l'algorithme : [PRO]

$S := S_0$

$E := E(S_0)$

$K := kmax$

**Tant que** $k > 0$

$S_n :=$ Voisin(S)

$E_n := E(S_n)$

**Si** ($E_n < E$ **ou** $Rand < e^{-\Delta E/T}$) **alors**

$S := S_n$; $E := E_n$

$K := K-1$

**Retourne S**

### 6.3. Etat initiale de l'algorithme :

L'état initial est exécuté au début de l'algorithme, la solution initiale peut être prise au hasard dans l'espace des solutions possibles. Elle est sous forme d'une matrice où les cases représentent les processeurs et les contenues représentent les tâches.

- **La structure d'une solution :**

Les tâches de l'application : $T = \{t_1, t_2, \ldots\ldots, t_n\}$ ;

Les processeurs de NoC : $P = \{p_1, p_2, \ldots\ldots, p_m\}$ ;

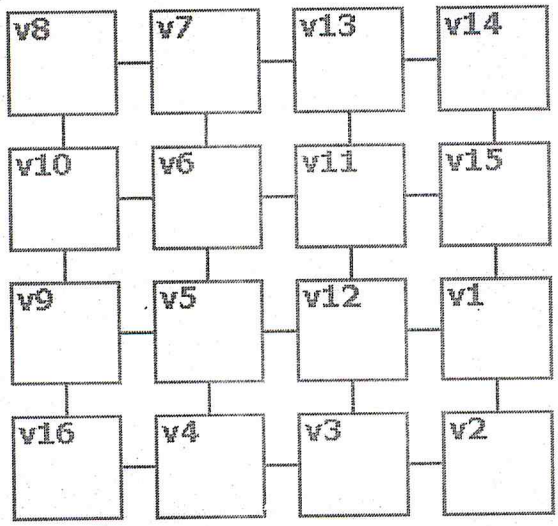


**Figure 21**: Exemple d'un mapping d'une application VOPD sur architecture NoC (Solution). **[MAS 10]**

Avant de calculer l'énergie correspondante à cette solution initiale, on teste d'abord si cette solution vérifie les conditions de faisabilité **(1)** et **(2)**, si oui donc cette solution est réalisable on rentre dans l'algorithme, sinon on itère une nouvelle solution initiale.

- **Le calcul d'énergie de solution initiale :**

A cette solution correspond une énergie initiale $E = E(S_0)$. Cette énergie est calculée par l'équation **(6)**.

### 6.4. Détermination d'une nouvelle solution :

En partant de la solution initiale, en la modifiant, on vérifie les conditions de faisabilité comme la solution initiale, on obtient une seconde. La modification pour obtenir l'état voisin aléatoire de cet état se fait par une inversion des éléments d'une paire choisie aléatoirement comme l'illustre la **Figure 22**. Deux processeurs choisis aléatoirement dans la matrice de la solution initiale sont inter changées et ainsi, une nouvelle solution est générée. L'inversion est exécutée à chaque appel de cette étape.

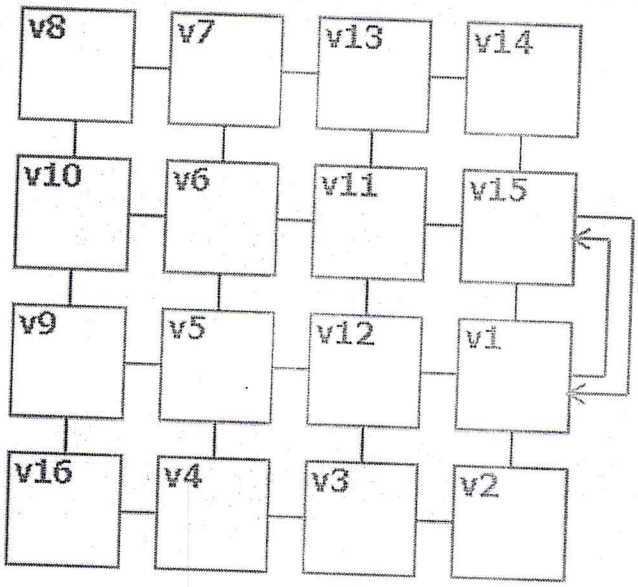


**Figure 22**: Inversion utilisée.

### 6.5. Calcul de $\Delta E$ :

Avant de calculer $\Delta E$ on calcule d'abord l'énergie du la nouvelle solution par l'équation(**6**), on a :

$E(S_n)$ :l'énergie de la nouvelle solution ;

$$\Delta E = E(S_n) - E(S) \ldots\ldots\ldots\ldots\ldots\ldots (7)$$

Si cette variation est négative (c'est-à-dire qu'elle fait baisser l'énergie du système), elle est appliquée à la solution courante. Sinon, elle est acceptée avec une probabilité :

$$e^{-\Delta E/T} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (8)$$

Ce choix de l'exponentielle pour la probabilité s'appelle ***règle de Metropolis***. Ensuite, on itère une valeur aléatoire dans l'intervalle [0, 1] (***Rand***), si $Rand<e^{-\Delta E/T}$ on accepte cette solution comme solution courante. Sinon, on revient à l'étape **6.4**. En continuant jusqu'à un maximum de nombre d'itération ou jusqu'à ce qu'un état ayant pour énergie le maximum ou moins soit trouvé.

### 6.6. Itération de l'algorithme :

A chaque itération de l'algorithme une modification élémentaire de la solution est effectuée. Cette modification entraine une variation de l'énergie du système ($\Delta E$).

### 6.7. Le schéma détaillé de recuit simulé :

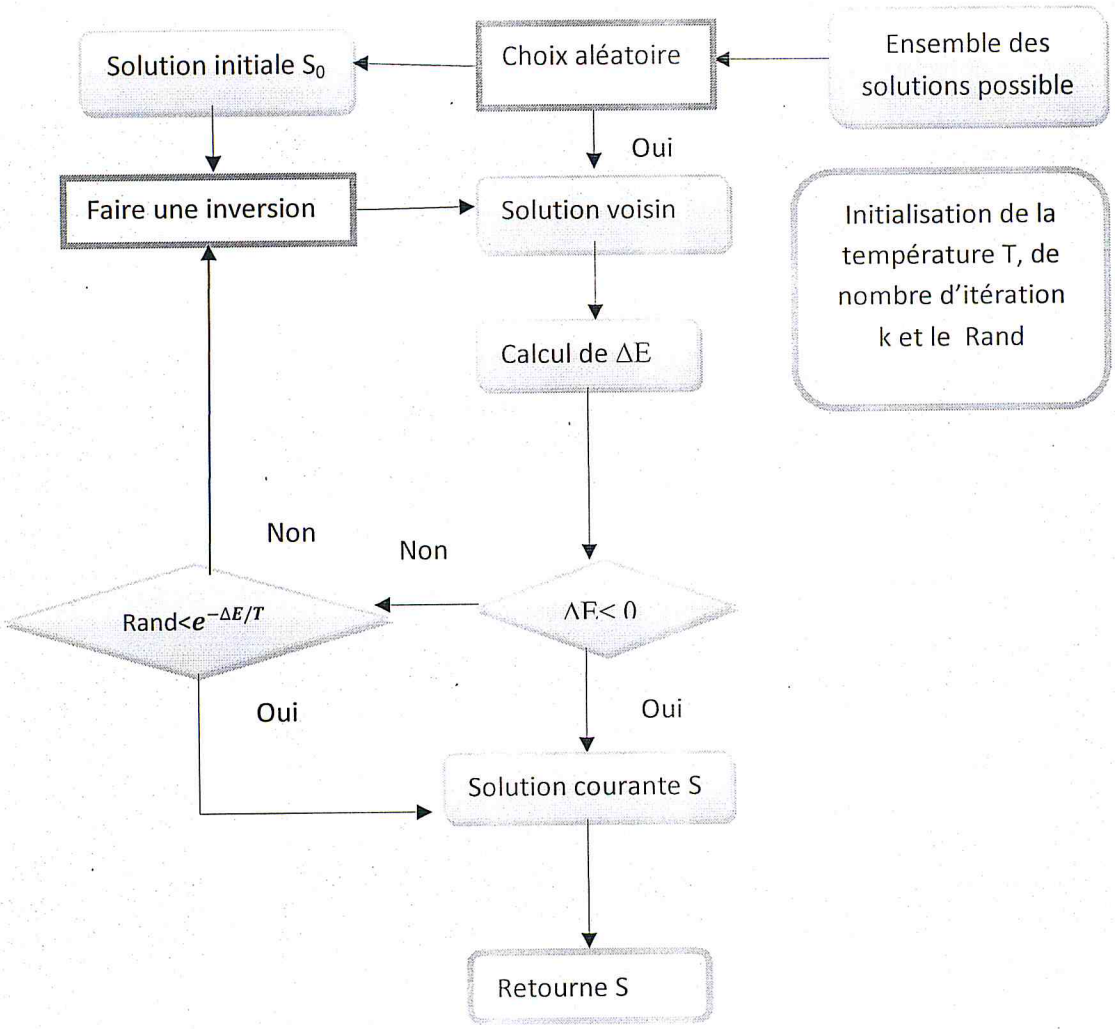


**Figure 23**: Organigramme de recuit simulé

## Conclusion

Après avoir exposé les concepts qui servent de base à l'algorithme recuit simulé, nous avons réalisé une adaptation de cet algorithme dans une démarche de recherche des bonnes solutions à notre problème de mapping d'application intensive sur une architecture NoC multiprocesseurs.

Dans le prochain chapitre, nous proposerons une implémentation de l'algorithme recuit simulé appliqués à notre problème.

# *Chapitre V*

# ***Test et Résultats***

## Introduction

Après avoir présenté notre technique proposée dans le chapitre précède, 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 après la réalisation des tests sur ces benchmarks.

Les tests ont été effectués sur une machine Intel Core i3 2.40 GHz avec 2Go de Ram sous l'environnement Windows (7 Entreprise).

## 1. Présentation des Benchmarks

Les Benchmarks sont utilisés pour tester les différentes techniques de mapping implémentées.

### 1.1. Présentation du Benchmark Video Object Plane Decoder (VOPD)

L'application VOPD est composée de 16 tâches 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 24** décrit cette application sous forme de graphe.

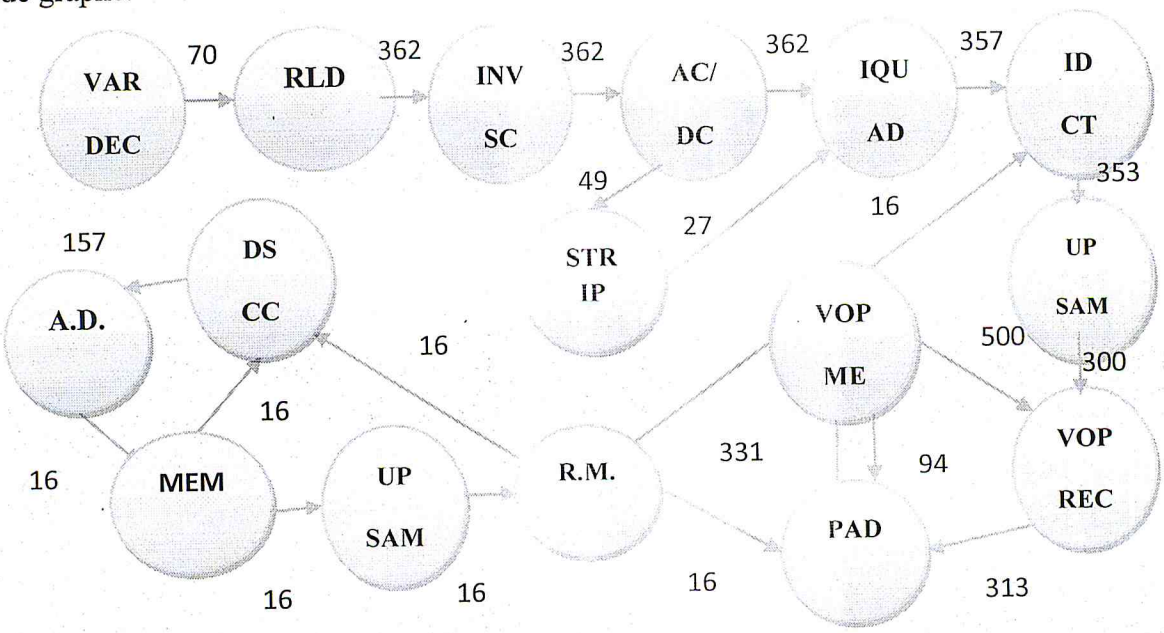


**Figure 24**: Graphe d'application du Benchmark VOPD. **[CAR 09]**

L'application VOPD est caractérisée comme suit:

| Video Object Plane Decoder | |
|---|---|
| **Type de réseau sur puce** | Hermès |
| **Dimension de la maille** | 4*4 |
| **Stratégie de contrôle de flux** | Credit-Based |
| **Algorithme de routage** | XY |

**Tableau 3**: Caractéristiqués de l'architecture du benchmark VOPD.

### 1.2. Présentation du Benchmark MPEG 4 Decoder in Decoder :

L'application MPEG 4 est composée de 12 tâches qui communiquent entre elles à travers liens. Le nombre total de paquets envoyé est de **3466** paquets. L'architecture NoC choisie est une maille de taille 4x4. La **Figure 25** décrit cette application sous forme de graphe. Ainsi, ses caractéristiques sont présentées sous forme **Tableau 4**.

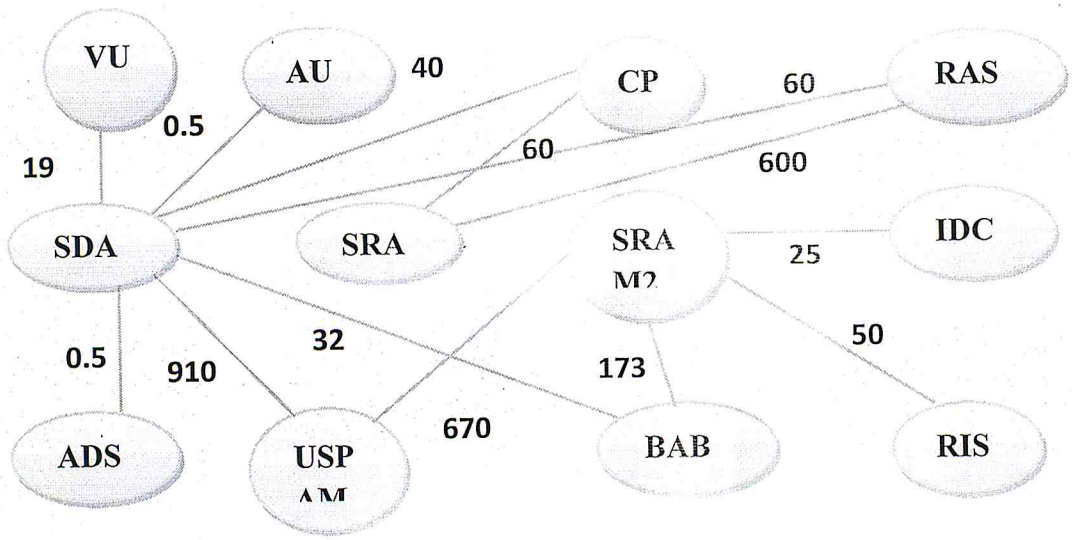


**Figure 25**: Graphe d'application du Benchmark MPEG 4. **[JAN 09]**

| MPEG 4 Decoder in Decoder | |
|---|---|
| Type de réseau sur puce | Hermès |
| Dimension de la maille | 4*4 |
| Stratégie de contrôle de flux | Credit-Based |
| Algorithme de routage | XY |

**Tableau 4**: Caractéristiques de l'architecture du benchmark MPEG 4.

### 1.3. Présentation du benchmark Multi-Window Display (MWD)

L'application MWD **[BER 05]** est composée de 12 IPs communicant à travers 13 liens. Le nombre total de paquets échangés est de 1120 paquets. Le graphe est présenté par la **Figure 26**.

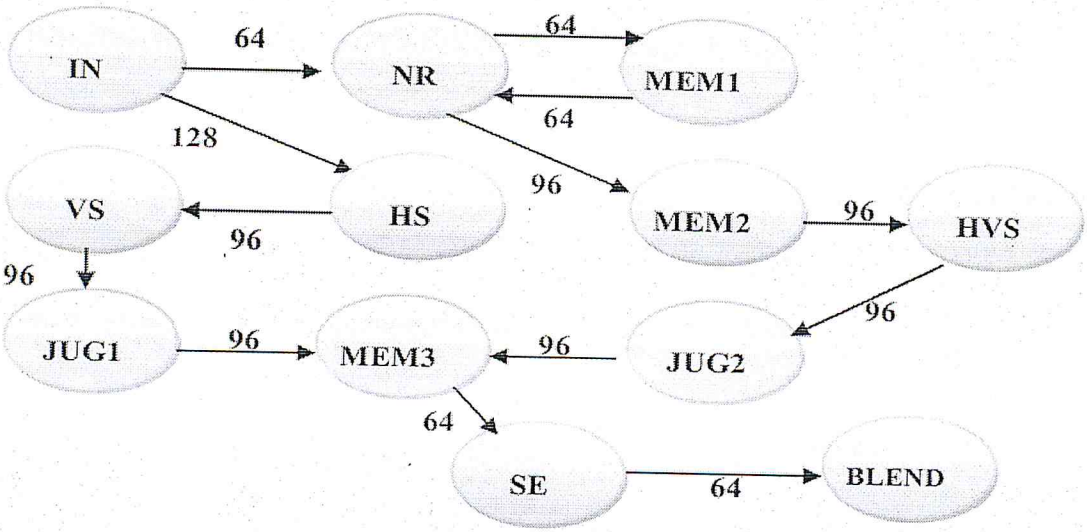


**Figure 26**: Le graphe d'application du benchmark MWD. **[CAR 09]**

Par conséquent, l'architecture choisie présente les caractéristiques suivantes :

| **Multi Window Display** | |
|---|---|
| Type de réseau sur puce | Hermès |
| Dimension de la maille | 4*4 |
| Stratégie de contrôle de flux | Credit-Based |
| Algorithme de routage | XY |

**Tableau 5**: Caractéristiques de l'architecture du benchmark MWD.

## 2. 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 proposé par rapport à la littérature. La métrique prise en considération dans ce cas est le coût des communications, l'énergie consommée et espace pour cela nous utilisons les benchmarks le plus utilisé dans ce domaine

VOPD, MPEG4 et MWD. On considère que notre architecture NoC est hétérogène et chaque processeur ne peut exécuter qu'une seul tâche.

### 2.1. Etude paramétrique pour Recuit Simulé

Cette étude est effectuée en réalisant les tests sur les trois benchmarks VOPD, MPEG4 et MWD. On commence par l'étude d'influence de nos paramètres de l'algorithme.

- **Influence du nombre d'itérations :**

Le nombre d'itérations désigne le nombre des générations produites. Il est clair que plus le nombre d'itérations augmente, meilleurs sont les résultats. On a fixé La température =20 et Rand =0.01 et on varie le nombre d'itération pour chaque critère. Les tableaux suivants résument les résultats obtenus :

- **Pour la communication**

| Benchmark | Nombre d'itération | Température | Rand | Cout de communication |
|---|---|---|---|---|
| VOPD | 10000 | 20 | 0.01 | 4135 |
| | 5000 | 20 | 0.01 | 4187 |
| | 1000 | 20 | 0.01 | 4200 |
| | 500 | 20 | 0.01 | 4311 |
| MPEG | 10000 | 20 | 0.01 | 3800 |
| | 5000 | 20 | 0.01 | 3931 |
| | 1000 | 20 | 0.01 | 4091 |
| | 500 | 20 | 0.01 | 4258 |
| MWD | 10000 | 20 | 0.01 | 1280 |
| | 5000 | 20 | 0.01 | 1408 |
| | 1000 | 20 | 0.01 | 1472 |
| | 500 | 20 | 0.01 | 1536 |

**Tableau 6**: Tableau Comparatif de variation de nombre d'itération pour la communication.

- Pour l'énergie consommée

| Benchmark | Nombre d'itération | Température | Rand | Cout d'énergie |
|---|---|---|---|---|
| VOPD | 10000 | 20 | 0.01 | 94 |
| | 5000 | 20 | 0.01 | 80 |
| | 1000 | 20 | 0.01 | 83 |
| | 500 | 20 | 0.01 | 95 |
| MPEG | 10000 | 20 | 0.01 | 74 |
| | 5000 | 20 | 0.01 | 72 |
| | 1000 | 20 | 0.01 | 60 |
| | 500 | 20 | 0.01 | 78 |
| MWD | 10000 | 20 | 0.01 | 76 |
| | 5000 | 20 | 0.01 | 80 |
| | 1000 | 20 | 0.01 | 70 |
| | 500 | 20 | 0.01 | 68 |

**Tableau 7**: Tableau Comparatif de variation de nombre d'itération pour l'énergie consommée.

- Pour l'espace

| Benchmark | Nombre d'itération | Température | Rand | L'espace |
|---|---|---|---|---|
| VOPD | 10000 | 20 | 0.01 | 8082 |
| | 5000 | 20 | 0.01 | 10572 |
| | 1000 | 20 | 0.01 | 10352 |
| | 500 | 20 | 0.01 | 10275 |
| MPEG | 10000 | 20 | 0.01 | 7175 |
| | 5000 | 20 | 0.01 | 7768 |
| | 1000 | 20 | 0.01 | 7819 |

| | | | | |
|---|---|---|---|---|
| | 500 | 20 | 0.01 | 7955 |
| MWD | 10000 | 20 | 0.01 | 7792 |
| | 5000 | 20 | 0.01 | 7682 |
| | 1000 | 20 | 0.01 | 7796 |
| | 500 | 20 | 0.01 | 7935 |

**Tableau 8**: Tableau Comparatif de variation de nombre d'itération pour l'espace.

- **Le cout total**

| Benchmark | Nombre d'itération | Température | Rand | Cout total |
|---|---|---|---|---|
| VOPD | 10000 | 20 | 0.01 | 12311 |
| | 5000 | 20 | 0.01 | 14839 |
| | 1000 | 20 | 0.01 | 14635 |
| | 500 | 20 | 0.01 | 14681 |
| MPEG | 10000 | 20 | 0.01 | 11072 |
| | 5000 | 20 | 0.01 | 11771 |
| | 1000 | 20 | 0.01 | 11970 |
| | 500 | 20 | 0.01 | 12291 |
| MWD | 10000 | 20 | 0.01 | 9084 |
| | 5000 | 20 | 0.01 | 9170 |
| | 1000 | 20 | 0.01 | 9338 |
| | 500 | 20 | 0.01 | 9539 |

**Tableau 9**: Tableau Comparatif de variation de nombre d'itération pour le cout total.

Nous remarquons que les valeurs de nombre d'itération qui fournissent des bons résultats ne sont pas toujours les mêmes pour les trois critères et que la solution meilleure (proche de l'optimale) pour le cout de communication reste meilleure pour le cout total.

### Influence de la température :

Désigne la température de système. Une fois qu'elle est basse, l'algorithme tombe dans un résultat minimal (meilleur).On a fixé Le nombre d'itération =10000 et Rand =0.01 et on varie la température pour chaque critère. Les tableaux ci-dessous résument les résultats obtenus.

- **Pour la communication**

| Benchmark | Nombre d'itération | Température | Rand | Cout de communication |
|---|---|---|---|---|
| **VOPD** | **10000** | **20** | **0.01** | **4156** |
| | **10000** | **50** | **0.01** | **4166** |
| | **10000** | **100** | **0.01** | **4269** |
| | **10000** | **200** | **0.01** | **4355** |
| **MPEG** | **10000** | **20** | **0.01** | **3823** |
| | **10000** | **50** | **0.01** | **3995** |
| | **10000** | **100** | **0.01** | **4097** |
| | **10000** | **200** | **0.01** | **4181** |
| **MWD** | **10000** | **20** | **0.01** | **1280** |
| | **10000** | **50** | **0.01** | **1472** |
| | **10000** | **100** | **0.01** | **1504** |
| | **10000** | **200** | **0.01** | **1600** |

**Tableau 10**: Tableau Comparatif de variation de la température pour la communication.

- **Pour l'énergie consommée**

| Benchmark | Nombre d'itération | Température | Rand | Consommation d'énergie |
|---|---|---|---|---|
| **VOPD** | **10000** | **20** | **0.01** | **85** |
| | **10000** | **50** | **0.01** | **92** |

| | | | | |
|---|---|---|---|---|
| | 10000 | 100 | 0.01 | 85 |
| | 10000 | 200 | 0.01 | 88 |
| MPEG | 10000 | 20 | 0.01 | 74 |
| | 10000 | 50 | 0.01 | 58 |
| | 10000 | 100 | 0.01 | 52 |
| | 10000 | 200 | 0.01 | 59 |
| MWD | 10000 | 20 | 0.01 | 76 |
| | 10000 | 50 | 0.01 | 80 |
| | 10000 | 100 | 0.01 | 64 |
| | 10000 | 200 | 0.01 | 78 |

**Tableau 11**: Tableau Comparatif de variation de la température pour la consommation d'énergie.

- **Pour l'espace**

| Benchmark | Nombre d'itération | Température | Rand | L'espace |
|---|---|---|---|---|
| VOPD | 10000 | 20 | 0.01 | 10467 |
| | 10000 | 50 | 0.01 | 10387 |
| | 10000 | 100 | 0.01 | 10467 |
| | 10000 | 200 | 0.01 | 10297 |
| MPEG | 10000 | 20 | 0.01 | 7190 |
| | 10000 | 50 | 0.01 | 7793 |
| | 10000 | 100 | 0.01 | 7733 |
| | 10000 | 200 | 0.01 | 7742 |
| MWD | 10000 | 20 | 0.01 | 7650 |
| | 10000 | 50 | 0.01 | 7802 |
| | 10000 | 100 | 0.01 | 7812 |
| | 10000 | 200 | 0.01 | 7190 |

**Tableau 12**: Tableau Comparatif de variation de la température pour l'espace.

- **Le cout total**

| Benchmark | Nombre d'itération | Température | Rand | Cout total |
|---|---|---|---|---|
| **VOPD** | **10000** | **20** | **0.01** | **14708** |
| | **10000** | **50** | **0.01** | **14645** |
| | **10000** | **100** | **0.01** | **14821** |
| | **10000** | **200** | **0.01** | **14740** |
| **MPEG** | **10000** | **20** | **0.01** | **11092** |
| | **10000** | **50** | **0.01** | **11818** |
| | **10000** | **100** | **0.01** | **11962** |
| | **10000** | **200** | **0.01** | **12000** |
| **MWD** | **10000** | **20** | **0.01** | **9148** |
| | **10000** | **50** | **0.01** | **9334** |
| | **10000** | **100** | **0.01** | **9378** |
| | **10000** | **200** | **0.01** | **11092** |

**Tableau 13**: Tableau Comparatif de variation de la température pour le cout total.

Nous remarquons que les valeurs de température qui fournissent des bons résultats ne sont pas toujours les mêmes pour les trois critères et que la solution meilleure pour le cout de communication ne reste pas meilleure pour le cout total mais les résultats restent bons.

- **Influence de Rand :**

Désigne la valeur comparée avec la règle de métropolis. Plus le Rand proche de 0, les résultats sont les meilleurs. On a fixe Le nombre d'itération =10000 et la température =20 et on varie le Rand pour chaque critère. Les tableaux suivants résument les résultats obtenus.

- **Pour la communication**

| Benchmark | Nombre d'itération | Température | Rand | Cout de communication |
|---|---|---|---|---|
| **VOPD** | **10000** | **20** | **0.01** | **4141** |
| | **10000** | **20** | **0.03** | **4226** |
| | **10000** | **20** | **0.06** | **4323** |
| | **10000** | **20** | **0.09** | **4413** |
| **MPEG** | **10000** | **20** | **0.01** | **3771** |
| | **10000** | **20** | **0.03** | **4181** |
| | **10000** | **20** | **0.06** | **4020** |
| | **10000** | **20** | **0.09** | **4053** |
| **MWD** | **10000** | **20** | **0.01** | **1226** |
| | **10000** | **20** | **0.03** | **1344** |
| | **10000** | **20** | **0.06** | **1472** |
| | **10000** | **20** | **0.09** | **1522** |

**Tableau 14**: Tableau Comparatif de variation de Rand pour la communication.

- **Pour l'énergie**

| Benchmark | Nombre d'itération | Température | Rand | Consommation d'énergie |
|---|---|---|---|---|
| **VOPD** | **10000** | **20** | **0.01** | **94** |
| | **10000** | **20** | **0.03** | **79** |
| | **10000** | **20** | **0.06** | **78** |
| | **10000** | **20** | **0.09** | **91** |
| **MPEG** | **10000** | **20** | **0.01** | **76** |
| | **10000** | **20** | **0.03** | **81** |
| | **10000** | **20** | **0.06** | **92** |
| | **10000** | **20** | **0.09** | **100** |
| **MWD** | **10000** | **20** | **0.01** | **60** |
| | **10000** | **20** | **0.03** | **78** |

| | | | | |
|---|---|---|---|---|
| | 10000 | 20 | 0.06 | 81 |
| | 10000 | 20 | 0.09 | 72 |

**Tableau 15**: Tableau Comparatif de variation de Rand pour la consommation d'énergie.

- **Pour l'espace**

| Benchmark | Nombre d'itération | Température | Rand | L'espace |
|---|---|---|---|---|
| **VOPD** | **10000** | **20** | **0.01** | **8172** |
| | **10000** | **20** | **0.03** | **10572** |
| | **10000** | **20** | **0.06** | **10352** |
| | **10000** | **20** | **0.09** | **10275** |
| **MPEG** | **10000** | **20** | **0.01** | **7190** |
| | **10000** | **20** | **0.03** | **7793** |
| | **10000** | **20** | **0.06** | **7733** |
| | **10000** | **20** | **0.09** | **7742** |
| **MWD** | **10000** | **20** | **0.01** | **8120** |
| | **10000** | **20** | **0.03** | **7802** |
| | **10000** | **20** | **0.06** | **7812** |
| | **10000** | **20** | **0.09** | **7731** |

**Tableau 16**: Tableau Comparatif de variation de Rand pour l'espace.

- **Pour le cout total**

| Benchmark | Nombre d'itération | Température | Rand | Cout total |
|---|---|---|---|---|
| **VOPD** | **10000** | **20** | **0.01** | **12407** |
| | **10000** | **20** | **0.03** | **14877** |
| | **10000** | **20** | **0.06** | **14653** |
| | **10000** | **20** | **0.09** | **14681** |

| | | | | |
|---|---|---|---|---|
| MPEG | 10000 | 20 | 0.01 | 11262 |
| | 10000 | 20 | 0.03 | 12055 |
| | 10000 | 20 | 0.06 | 12025 |
| | 10000 | 20 | 0.09 | 12195 |
| MWD | 10000 | 20 | 0.01 | 9406 |
| | 10000 | 20 | 0.03 | 9224 |
| | 10000 | 20 | 0.06 | 9365 |
| | 10000 | 20 | 0.09 | 9293 |

**Tableau 17**: Tableau Comparatif de variation de Rand pour le cout total.

Nous remarquons que les valeurs de Rand qui fournissent des bons résultats ne sont pas toujours les mêmes pour les trois critères et que la solution meilleure pour le cout de communication ne reste pas meilleure pour le cout total mais les résultats restent toujours bons.

Après cette étude, on dit que le nombre d'itération a une forte influence par rapport aux autres paramètres. Le Recuit Simulé améliore considérablement les résultats à chaque augmentation du nombre d'itération par rapport au cout total (notre fonction objective).

### 2.2. Etude comparative avec d'autres techniques de mapping :

Nous n'avons pas implémenté ces techniques mais nous nous sommes basés sur les résultats publiés. Les tests ont été réalisés sur les benchmark VOPD et MPEG-4. Les techniques comparées entre eux sont RS (Recuit Simulé), GMAP basée sur les algorithmes génétiques, CGMAP, utilisant une hybridation (Algorithme génétique-Séquences chaotiques), SPIRAL et ONYX.

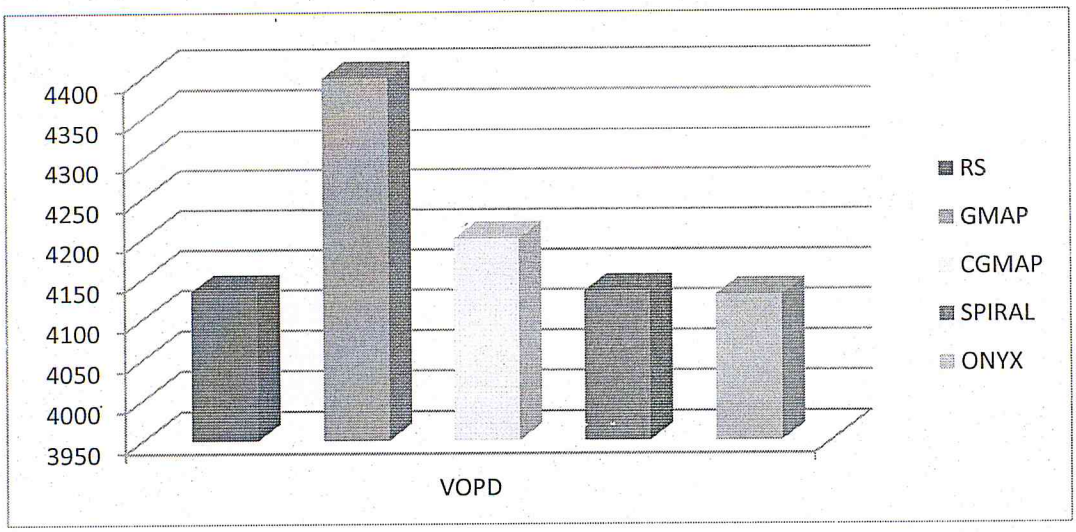


**Figure 27**: comparaison du cout de communication des différents algorithmes de mapping dans une application VOPD. **[MAS 10, JAN 11]**

Nous remarquons que la technique proposée (RS) produit des bons résultats par rapport à la solution optimale (4135).

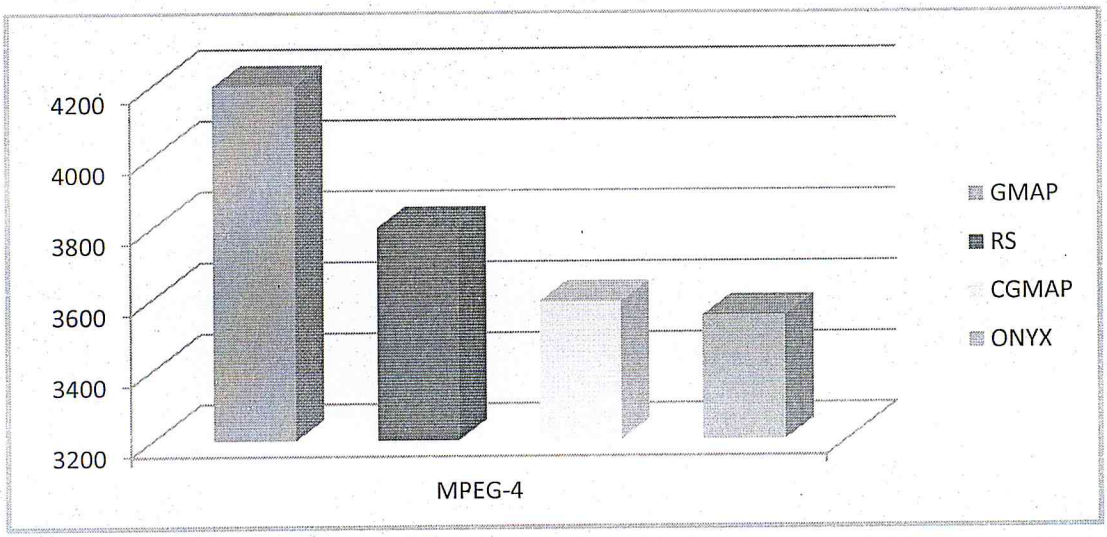


**Figure 28**: comparaison du cout de communication des différents algorithmes de mapping dans une application MPEG-4. **[MAS 10, JAN 11]**

La comparaison effectuée est basée sur les résultats publiés suite aux simulations des différentes techniques **[MEH 07]** .

Notre techniques arrive à trouver la solution proche de l'optimale (3600 selon la littérature).

## Conclusion

A travers les tests effectués, notre but était d'étudier les performances de la technique de mapping proposée.

La première étude permet de démontrer l'effet des paramètres sur l'efficacité de technique et sur les résultats obtenus, on prise en compte nos critères à optimiser.

Ensuite, une étude comparative de la technique proposée avec quelques techniques publiées dans la littérature est réalisée. Dans cette étude, la métrique prise en compte est le coût des communications.

# *Conclusion générale*

Les systèmes sur puce (SoCs) ont été très utilisés dans les dizaines d'années passées pour équiper des systèmes digitaux complexes. Cependant, avec la taille et la complexité toujours grandissantes des nouvelles applications, les structures de communication utilisées dans les SoCs, comme le bus partagé, trouvent leurs limites. Dans ce contexte, le concept de réseau sur puce a été proposé comme une solution prometteuse pour pallier aux inconvénients des bus partagés. En effet, il est plus performant en termes de flexibilité, fiabilité et extensibilité.

Comme tout nouveau domaine, le concept de réseau sur puce a connu beaucoup de travaux de recherche touchant tous les aspects et les notions qui lui sont liés. La phase de mapping, constitue une étape cruciale dans le processus de conception des réseaux sur puce. En effet, un mauvais mapping des composantes logicielles d'une application peut considérablement dégrader les performances globales du système final. C'est pourquoi, il est très intéressant de développer des méthodes et des outils pour automatiser cette étape.

Le but principal du travail présenté dans ce document est de proposer une technique de mapping d'une application sur une structure de réseau sur puce en est de rendre le problème de mapping plus difficile ,puisque on essaye d'ajoute des critère de faisabilité : la communication, énergie consommée et l'espace (surface) sur la solution de mapping avant d'optimiser les résultat obtenue.

Pour cela et d'après les recherches effectuées, on a utilisé une méthode basée sur l'algorithme d'optimisation **Recuit simulé** qui est un algorithme heuristique, Cet algorithmes cherche la bonne solution dans un espace de solution réalisable en respectant nos critères.

En fait il n'existe pas une plateforme qui permet 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 avec le langage de programmation (Java).

## Perspectives

Certains enrichissements peuvent être apportés à notre travail, d'abord nous implémentons une méthode d'optimisation multi objective (multicritère) avec agrégation.

- Développer cette technique avec Pareto.
- Faire d'autres méthodes d'optimisation.

# *Bibliographie*

[ABO 08] Abou El Hassan **BENYAMINA «Ordonnancement hiérarchique multi-objectif d'applications embarquées intensives»**. Thèse de doctorat. Université d'Oran.2008.

[ACR 09] Ankur Agarwal, Cyril Ishander, Ravi Shankar « **Survey of Network on Chip (NoC) Architectures and Contributions** ». Engineering, Computing and Architecture ISSN 1934-7197 Volume 3, Issue 1, 2009.

[AND 88] ANDRE Françoise et PAZAT Jean-Louis « **Le placement de tâches sur des architectures parallèles** ». Revue TSI : Technique et science informatiques 1988, vol. 7, n°4, pp. 385-401.

[ASH 06] Ashish Menna **«Allocation, Assignation et Ordonnancement pour les Systèmes sur multiprocesseurs»**. Thèse de doctorat, Université des sciences et technologie de Lille, Décembre 2006.

[ATA 07] ATAT Youssef **«Conception de haut niveau des MPSoCs à partir d'une spécification Simulink : Passerelle entre la conception au niveau Système et la génération d'architecture »**. Thèse de doctorat. Institut National Polytechnique de Grenoble, 2007.

[BAS 05] M. Basseur **« conception d'algorithmes coopératifs pour l'optimisation Multiobjectif: application aux problèmes d'ordonnancement de type flow-shop »**. Thèse de doctorat. Université des Sciences et Technologies de Lille(USTL) ,2005 .

[BER 05] BERTOZZI Davide, JALABERT Antoine, MURALI Srinivasan and Student Member **«NoC Synthesis Flow for Customized Domain Specific Multiprocessor Systems-on-Chip»**. IEEE

Transactions on parallel and distributed systems, Vol 16, N° 2, 2005.

[BOU 06] BOUCHHIMA Aimen **«Modélisation du logiciel embarque à différents niveau d'abstraction en vue de la validation et la synthèse des systèmes monopuces »**. Thèse de doctorat. Institut National Polytechnique de Grenoble(INPG), 2006.

[BOU 12] BOUMAAZA Farid **«Mapping multi-objectifs d'applications intensives sur architectures MPSoC»**.Thèse de magistère. Université d'Oran.2012.

[CAR 09] CARVALHO Ewerson, MARCON César, CALAZANS Ney and MORAES Fernando **« Evaluation of Static and Dynamic Task Mapping Algorithms in NoC-Based MOEPCs »**.Proceedings of the 11th international conference on System-on-chip, 2009.

[CER 85] V.Cerny. **« Thermodynamical Approach to the Traveling Salesman Problem: an Efficient Simulation Algorithm»**. Journal of Optimization Theory and Applications, Vol. 45, pp. 41--51, 1985.

[COL 02] Y. Collette et P. Siarry **« Optimisation multiobjectif»**. Eyrolles, 2002.

[DAL 01] DALLY William J. and TOWLES Brian « Route Packets, Not Wires: On-Chip Interconnection Networks ». In Proc. Design Automation Conf., 2001.

[DEL 03] DELORME Xavier **«Optimisation combinatoire et problèmes de capacité d'infrastructure ferroviaire»**. Thèse de doctorat, Université de Valenciennes et du Hainaut Cambrésis,

## Bibliographie


Valenciennes, France, Décembre2003.

[DEL 07] 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.

[FAG 09] Fahime Moein-darbari, Ahmad Khademzade et Golnar Gharooni-fard, **«CGMAP: a new approach to Network-on-Chip mapping problem »**.IEICE Electron. Express, Vol. 6, No. 1, pp.27-34, (2009).

[FRA 12] François LUCAS **« Des métaheuristique pour le guidage d'un solveur de contraintes dédié a la planification automatisé d'itinéraires de véhicules »**.Thèse de doctorat, l'Ecole national supérieure des mines de Paris

[FRE 00] FREVILLE Arnaud **« Méthodes de recherche locale »**. Journée AFPLC - École des Mines de Nantes, 2000.

[GLO 86] F. Glover. **«Future Parth for Integer Programming and Links to Artificial Intelligence»**. Computers and Operations Research, Vol. 13, No. 5, pp. 533-549, 1986.

[HAO 99] J.-K. Hao, P. Galinier, M. Habib **« Métaheuristique pour l'optimisation combinatoire et l'affectation sous contraintes »**. Revue d'intelligence artificielle Vol:No.1999. http://www.info.univ-angers.fr/pub/hao/papers/RIA.pdf

[HAS 70] W. Hastings. **« Monte Carlo Sampling Methods Using Markov Chains and Their Applications »**. Biométrika, vol.57, no. 1, pp.

97—109, 1970.

[HOU 07] DELORME Julien et HOUZET **Dominique «Technique de mapping pour les réseaux sur puce 2D»**. PATMOS 2007, Suède Göteborg, Septembre 2007.

[IAR 01] I. Bolsens, A. Jerraya, R. Bergamaschi **«Are single-chip multiprocessors in reach? »** .IEEE Design and TestofComputers, Vol.18, p.82–89, 2001.

[JAL 04] JALABERT Antoine, MURALI Srinivasan, BENINI luca and DE MICHELI Giovanni **«XpipesCompiler : A tool for instantiating application specific Networks on Chip»**

[JAN 09] M. Janidarmian, A. Khademzadeh et M. Tavanpour, **«Onyx: A new heuristic bandwidth-constrained mapping of cores onto tile based Network on Chip»**, IEICE Electron. Express, Vol. 6, No. 1, pp.1-7, January 2009.

[JAN 11] Majid Janidarmian, Atena Roshan Fekr **«A Survey of Meta-Heuristic Solution Methods for Mapping Problem in Network-on-Chips»**, Novembre 2011, www.scientific.net/AMR.403-408.3994 .

[KIR 83] S. Kirkpatrik, C. Gelatt, and M. Vecchi. **«Optimization by simulated Annealing»**. Science, vol. 220, No. 4598, pp. 671-680, may 1983.

[LAC 05] P. Lacomme, **« méthodes exactes et approchées pour l'optimisation des systèmes à moyens de transport »**. Mémoire pour l'obtention de l'Habilitation à Diriger des Recherches de l'Université Blaise Pascal (Clermont –Ferrand II).06/07/2005 :

# Bibliographie

NO :194 ;


[LAK 98] Lance Hammond, Kunle Olukotun « **Considerations in the design of Hydra: A multiprocessor-on-a-chip micro architecture**». Technical report,Stanford, CA, USA, 1998.

[LEI 03] LEI Tang and KUMAR Shashi « **A Two-Step Genetic Algorithm for Mapping Task Graphs to a Network on Chip Architecture** ».Digital System Design. Proceedings. Euromicro Symposium on, 2003.

[LEM 06] 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.

[MAP 12] B.FARID, AH.BENYAMINA « **Mapping multi Objectifs d'application intensive Sur architecture MPSOC ».** Novembre 28, 2012. http://ceur-ws.org/Vol-942.

[MAR 99] N. Marco, J.-A. Désidéri, S. Lanteri«**Multi-Objective Optimization in CFD by Genetic Algorithms**». Rapport de recherche n° 3686 INRIA (SOPHIA ANTIPOLIS),Avril 1999.

[MAS 10] Misagh Tavanpour, Ahmed Khademzadeh, Soumeyya Pourkiani and Mehdi Yaghobi« **GBMAP : A new Evolutionary Approach to Mapping Core onto Mesh-based NoC Architecture** ».Journal of Communication and Computer, ISSN 1548-7709, USA ,Mar. 2010, Volume 7, No.3 (Serial No.64).

# Bibliographie


[MCK 93] MCKINLEY Philip and NI Lionel « ***A Survey of Wormhole Routing Techniques in Direct Networks*** ». Computer, 1993.

[MEH 07] MEHRAN Armin, SAEIDI Samira, KHADEMZADEH Ahmad and AFZALI-KUSHA Ali « **SPIRAL: A heuristic mapping algorithm for Network on chip** ». IEICE Electronics Express, 2007.

[MEL 05] MELLO Aline, TADESCO Leonel, CALAZANSNey LaertVilarand MORAES Fernando Gehm« **Virtual Channels in Networks on Chip: Implementation and Evaluation on Hermes NoC** ». 18th SBCCI, pp. 178-183, 2005.

[MET 53] N.Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, E. Teller. **«Equation of State Calculation by Fast Computing Machines»**. Journal of Chemical Phisics, vol.21, no. 6, pp. 1087—1092, 1953.

[MOH 10] Mohamed Akli Redjedal« **Optimisation multicritère pour le placement d'applications intensives sur système-sur-puce (SoC)** ». Thèse de Master, l'Université de Lille, 2010.

[MOR 03] MORAES Fernando, MELLO Aline, MÔLLER Leandro, OST Luciano, and CALAZANS Ney « ***A Low Area Overhead Packet-switched Network on Chip: Architecture and Prototyping*** ». IFIP Very Large Scale Integration (VLSI-SOC), 2003, pp. 318-323.

[MUR 04] MURALI Srinivasan and DE MICHELI Giovanni « **SUNMAP: a tool for automatic topology selection and generation for NoC**». Design Automation Conference, 2004.

# Bibliographie


[NIC 10] NICOPOULOS Chrysostomos, NARAYANAN Vijaykrishnan, and DAS Chita R « **Network-on-chip architectures: a holistic design exploration**». Springer, 2009.

[ORA 09] Oulhaci Abdalhak, Rajdal mohammed akli. **"Ordonnancement hiérarchique d'application temps réel sur une architecture NOC ".** Mémoire de fin d'étude. Université d'Oran. Juin 2009.

[PAM 05] Jingcao Hu, R. Marculescu« **Energy- and Performance-Aware Mapping for Regular NoC Architectures»,** IEEE Trans,On Computer-Aided Design of Integratedvol. 24, no. 4, pp. 551-562, April. 2005.

[RIS 05] RISO Séverine « **Evaluation des paramètres des Réseaux sur puce** ». Thèse de doctorat, Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier(LIRMM), 2005.

[ROS 67] R.S.Rosenberg « **Simulation of genetic populations with biochemical properties**». PhD thesis, Universit of Michigan, Ann Harbor, Michigan, 1967.

[SAD 99] Sadiq M. Sait and Habib Youssef. **"Iterative Computer Algorithms with Applications in Engineering: Solving Combinatorial Optimization Problems"**. IEEE Computer Society Press, Los Alamitos, CA, USA, 1999.

[SCH 06] SCHERRER Antoine « **Analyses statistiques des communications sur puce**». Thèse de doctorat, Ecole Normale Supérieure de Lyon, 2006.

# Bibliographie


[TAN 08] TANOUGAST Camel, JOVANOVIC Slavisa, MONTEIRO Fabrice, DIOU Camille, DANCDACHE Abbas « **Initiation à la modélisation et Co-simulation comportementale C-VHDL d'un réseau de communication sur Puce (Network on Chip)**». 10$^{es}$Journées Pédagogiques du CNFM, Saint-Malo, 2008.

[TRA 08] TRAN Xuan-Tu **«Méthode de Test et Conception en Vue du Test pour les Réseaux sur Puce Asynchrones : Application au Réseau ANOC»**.Thèse de doctorat, Institut National Polytechnique de Grenoble (CEA- LETI), 2008.

[WRO 99] W. Ron **«Is soc really different?** ». November8, 1999. http://www.eetimes. Com/story/OEG19991108S0009.

[ZHE 13] Zhenlong Song, Yong Dou, Mingling Zheng, and Weixia Xu « **A Quick Method for Mapping Cores Onto 2D-Mesh Based Networks on Chip** », School of Computer, National University of Defense Technology, Changsha China : NCCET 2012, CCIS 337, pp. 173–184, 2013


## Webographie


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

[GAP] http://www.inf.pucrs.br/~gaph/AtlasHtml/AtlasIndex_us.html

[OCP] http://ocpip.org

[PRO] http://www.proba.jussieu.fr/cours/dea/telehtml/node37.html