République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université SAAD DAHLEB BLIDA-1
Faculté des sciences
Département d'informatique

Mémoire de fin d'étude pour l'obtention du diplôme MASTER
Option : Systèmes Informatiques et Réseaux

Thème :

Optimisation multiobjectif des performances des réseaux sur puce

---

Réalisé par :
ABBAD Salim
TAILEB Selma

Soutenu devant un jury constitué de :

- ❖ Mme Zahra F.Z. — Présidente
- ❖ Mr Kameche A. — Examinateur
- ❖ Mme Toubaline N. — Promotrice

Année universitaire: 2019/2020

Résumé

La conception moderne des systèmes sur puce (SoC) montre une nette tendance vers l'intégration de plusieurs composants au sein d'une même puce. Pour la communication de ces derniers, différents types d'interconnexion ont été proposés (point à point, bus partagé et bus hiérarchique). Cependant ces interconnexions rencontrent des limites dans les systèmes caractérisés par de fortes communications. Le réseau sur puce (NoC) est apparu comme une structure de communication évolutive qui offre une amélioration des performances.

La conception d'un NoC nécessite plusieurs phases et paramétrages. On s'intéresse dans notre projet à la conception de topologies 3D personnalisées dans le cadre des applications spécifiques (ASIC) d'une part, et à la phase de mapping qui consiste au placement des composants communiquants sur la topologie du réseau d'autre part.

Dans cette perspective, et afin de proposer des solutions qui maximisent les performances des NoCs (minimisent le coût de communication et surface), nous nous sommes orientés vers les méthodes d'optimisation multiobjectif.

Mots clés : système sur puce, réseau sur puce, application spécifique, mapping, topologie personnalisée 3D, optimisation multiobjectif.

Abstract

Modern System-on-Chip (SoC) design shows a clear trend towards the integration of several components on a single chip. For the communication of the latter, different types of interconnection have been proposed (point-to-point, shared bus and hierarchical bus). However, these interconnections encounter limitations in systems characterized by strong communications. The Network on Chip (NoC) has emerged as a scalable communication structure that offers improved performance.

The design of a NoC requires several phases and parameterizations. In our project, we are interested in the design of customized 3D topologies for specific applications (ASICs) on the one hand, and in the mapping phase which consists in the placement of the components communicating on the network topology on the other hand.

In this perspective, and in order to propose solutions that maximize NoCs performances (minimize communication and surface cost), we have oriented ourselves towards multiobjective optimization methods.

Keywords : system-on-chip, network-on-chip, specific application, mapping, 3D custom topology, multiobjective optimization.

الملخص

التصميم الحديث للأنظمة على شريحة يظهر (soc) إتجاهًا واضحًا نحو دمج مكونات متعددة في شريحة واحدة. ولتحقيق التواصل بين هذه المكونات تم إقتراح أنواع مختلفة من الترابطات (من نقطة إلى نقطة ، ناقل مشترك وحافلة هرمية)، إلا أن هذه الأخيرة مازالت تواجه حدودًا خاصة بالنسبة للأنظمة ذات اتصالات قوية .لهذا ظهرت الشبكة على شريحة (NoC) كهيكل اتصال قابل للتطوير اذ انه يوفر أداءً أفضل.

يتطلب تصميم NoC عدة مراحل وإعدادات. بالنسبة لمشروعنا صببنا اهتمامنا نحو تصميم طبولوجيا ثلاثية الأبعاد مخصصة في سياق تطبيقات محددة (ASIC) من ناحية، و على مرحلة رسم الخرائط الذي يتوقف على تموضع المكونات التي تتواصل على طوبولوجيا الشبكة من ناحية أخرى.

من هذا المنظور، ومن أجل تقديم حلول تزيد من أداء NoCs (تقليل تكلفة الاتصال ومساحة السطح)، لجأنا إلى طرق تحسين متعددة الأهداف.

الكلمات الرئيسية: نظام على شريحة، شبكة على شريحة، تطبيق محدد، رسم خرائط، طوبولوجيا ثلاثية الأبعاد مخصصة، تحسين متعدد الأهداف.

Remerciements

En premier lieu et avant tout nous remercions DIEU « ALLAH » le tout puissant de nous avoir donné le courage, la patience et la force de réaliser ce projet de fin d'étude.

Nous tenons à remercier très chaleureusement notre promotrice Mme. TOUBALINE Nesrine (MCB à l'université SAAD DAHLAB Blida), qui nous a guidé tout au long de ce travail et nous a aidé avec ses conseils, et nous la remercions surtout pour la confiance qu'elle a mise en nous pour que nous puissions donner le meilleur de nous-même.

Nous présentons nos respects et nos sincères remerciements aux membres du jury qui nous ont fait l'honneur d'avoir accepté de faire partie du jury et de nous avoir consacré de leur temps précieux.

Nous remercions tous nos collègues et amis, pour les conseils, les services et plus particulièrement pour l'amitié qu'ils nous ont témoignées.

En terminant, nous souhaitons démontrer notre grande gratitude à toutes les personnes ayant participé de près ou de loin et plus particulièrement à nos familles à la réalisation de ce projet.

Dédicaces

***J**e dédie ce modeste travail accompagné d'un immense amour :*

***À** celle qui m'a donné la vie, qui m'a arrosé de tendresse et d'espoir à la source d'amour infini*
Ma très chère mère

***À** mon support dans la vie celui qui a été toujours à mes côtés et qui m'a soutenu dès mes débuts*
Mon très cher père

***À** celle qui m'a couvert par son affection et bienveillance et qui m'a toujours aimée et protégée*
Ma très chère grand-mère

***À** celle qui procure la joie et le bonheur pour toute la famille.*
Mon adorable soeur Amina .

***À** la source d'énergie incessible mon frère Hichem.*

***À** ceux avec qui j'ai passé les meilleurs moments de mon cursus : Amina, Chouaib, Imen, Selma, Yousra.*

***À** mon binôme Salim.*

Selma

Dédicaces

***J**e dédie le fruit de ce modeste travail comme un geste de gratitude*

À mes chers parents

***P**our leur sacrifice, leur amour, leur soutien et encouragements tout au long de mes études*

À mes très chères sœurs et frères

À mes amis

À mon binôme Selma

À tous les professeurs que ce soit du primaire, du moyen, du secondaire ou de l'enseignement supérieur

Salim

Table des matières

Table des figures

Liste des tableaux

LISTE DES ACRONYMES

A :

ACO : Les colonies de fourmis

ACP : Application Communication Pattern

AG : Algorithmes genetiques

ASIC : Un circuit integre specique a une application

B :

B&B : Branch-and-bound

C :

CAO : Conception assistée par ordinateur

CMAP : Constructive Mapping Algorithm,

D :

DSP : Traitement numerique du signal

DOL : Distributed Operation Layer

E :

EP : Programmation evolutive

ES : Strategies d'evolution

F :

FIFO : First in first out

FLIT : FLow control unIT

FPGA : Field Programmable Gate Arrays

G :

GP : Goal Programming

H :

HGLA : Hajela and Lin's Genetic Algorithm

I :

IC : Circuit Intégré

ILP : La programmation linéaire entière

IP : Intellectual Property

M :

Micro-GA : Micro genetic algorithm

MMT3D : 3D Modied Mesh Topology

MOGA : Multi-Objective Genetic Algorithms

MPSOC : Multiprocesseur systeme sur puce

N :

NBI : Normal Boundary Interaction

NI : Interface reseau

NoC : Réseau sur puce

NSGA : Non-dominated Sorting Genetic Algorithm

NSGAII : Non dominated sorting genetic algorithm II

O :

OCF : Optimisation par colonie de fourmis

P :

PSO : Particle Swarm Optimization

R :

ReNoC : Recongurable NoC

S :

SOC : Systeme sur puce

SNN : The Smart Nearest Neighbor

SP : Somme ponderee

T :

TSV : Through Silicon Via

V :

VCT : Virtual-cut-through

VDD : Valeurs de tensions d'alimentation

VEGA : Vector Evaluator Genetic Algorithm

Introduction générale

Depuis l'invention du transistor en 1947 puis le développement du premier circuit intégré (IC) environ dix ans plus tard, une évolution technologique des semi-conducteurs a permis d'intégrer plus d'un milliard de transistors dans le même substrat de silicium.

Le système sur puce (SoC) est un moyen moderne pour combiner de nombreuses fonctionnalités hétérogènes sur un même circuit intégré (IC). Le nombre de blocs fonctionnels de natures hétérogènes augmente, cependant les liens de communication n'évoluent pas à la même vitesse et deviennent un goulot d'étranglement.

Les interconnexions classiques comme point à point, bus partagé, bus hiérarchique trouvent leur limites face à la scalabilité, le parallélisme, la consommation d'énergie. Le réseau sur puce (NoC) est apparu comme une technologie évolutive majeure pour les SoC. Il est inspiré des réseaux informatiques classiques.

Problématique :

Lors de la conception des réseaux sur puce il faut suivre un processus en phases ou étapes, cependant chaque étape a ses défis. Citons le placement des composants communicants (mapping), la sélection de la topologie du réseau et le choix de la technologie d'intégration. Trouver une combinaison optimale de ces paramètres afin de maximiser les performances d'un NoC est un problème d'optimisation combinatoire.

Les outils de CAO (conception assistée par ordinateur) doivent optimiser les circuits au niveau de la performance. Les performances des réseaux sur puce se mesure communément selon plusieurs métriques, citons : le délai, la consommation d'énergie ou encore la surface. Toutefois, il n'existe pas de modèle d'évaluation des performances qui permette de décrire les compromis optimaux entre plusieurs mesures de performance du réseau.

L'objectif :

Le but de ce travail est de proposer un outil qui maximise les performances des réseaux sur puce dans un cadre multiobjectif.

Le plan du document :

Ce document est composé de 5 chapitres :

Le premier chapitre décrit le contexte et l'apparition des réseaux sur puce dans les sytèmes sur puce. Nous présentons les composants des réseaux sur puce, ces caractéristiques et ces phases de conception.

Le deuxième chapitre introduit les notions liés à l'optimisation multiobjectifs. Nous résumons les différentes approches et méthodes de résolution multiobjectifs et on conclut par l'analyse de performance des solutions obtenues dans un cadre multiobjectifs.

Le troisième chapitre se compose de deux sections : la première est consacré aux travaux qui s'intéressent à la topologie, la classification, la technologie 3D et les topologies personnalisées. La deuxième section porte sur la phase mapping, les types de mapping et les travaux de la littérature. Ce chapitre est achevé par une synthèse et critiques.

Le quatrième chapitre détaille nos contributions algorithmiques à travers les méthodes d'optimisation multiobjectif proposées. La méthode exacte, l'heuristique et la méta-heuristique adoptée aux problèmes du choix de topologie et mapping sont expliquées.

Dans le dernier chapitre, des tests sont effectués afin d'évaluer les solutions proposées. L'analyse des résultats obtenus permet de mettre en évidence les avantages des solutions élaborées.

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

Chapitre 1

Généralités sur les réseaux sur puce

1 Introduction

L'apparition des systèmes sur puce a permis aux systèmes en temps réel d'être plus performants en matière de puissance et de rapidité. Pour la communication des processeurs dans la même puce différents types d'interconnexion ont vu le jour.

Cependant, ces interconnexions classiques présentent des limites. Elles ne supportent pas les débits élevés, manquent de flexibilité et ne peuvent pas être adaptées pour les systèmes futurs implémentant plusieurs centaines d'unités de traitement.

Afin de surmonter ces problèmes, un nouveau paradigme de communication a été proposé par différents groupes de recherche dans les années 2000. C'est le réseau sur puce (NoC : Network on Chip). Ce type d'architecture présente de réels atouts au niveau de conception, performance, et implémentation.

Ce chapitre introduisait les concepts des réseaux sur puce. Nous allons décrire les différents types d'interconnexions utilisés dans les systèmes sur puce. Ensuite, nous représentons les composants et caractéristiques des réseaux sur puce. Enfin, nous exposerons ces phases de conception et quelques outils existant dans la littérature.

2 Généralités sur les Systèmes Sur Puce

Afin de pouvoir comprendre le système sur puce (SoC), il faut tout d'abord connaître certaines terminologies et connaissances qui ont une relation avec les SoCs :

2.1 Les circuits intégrés

À volume égal, le circuit intégré contient un nombre grandissant de transistors, autorisant un développement plus puissant dans un espace toujours moindre car plus un circuit peut contenir des transistors, plus nombreuses sont les fonctions qu'il est capable d'accomplir[1].

Les CIs peuvent être classifiés selon :

- **Leurs modes de programmation**

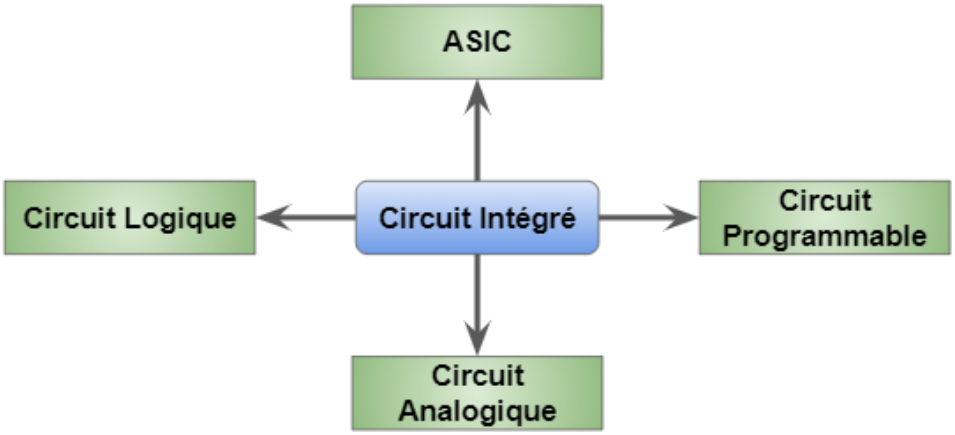


FIG. 1.1 : La classification des circuits intégrés[2]

1. ASIC : ce sont des circuits intégrés "fabriqués à la demande". Ils peuvent intégrer des structures analogiques et logiques mais sont d'un coût élevé à petite échelle.
2. Circuits analogiques : ce sont des circuits intégrés qui mettent en forme des informations analogiques.
3. Circuits programmés : ils nécessitent des informations virtuelles (un programme) régissant leur fonctionnement. Ce sont généralement des circuits logiques (Processeur, EPLD...).
4. Circuits logiques : regroupent les structures logiques intégrées non programmées.

- **Leurs niveaux d'intégration**

L'échelle d'intégration définit le nombre de portes par boîtier ou le nombre de fonctions intégrées sur un seul morceau, ils sont classés dans le tableau ci-dessous :

TAB. 1.1 : Échelle d'intégration[2]

Terme	Nombre de porte logique par boitier
SSI :Small Scale Integration	<12
MSI :Medium Scale Integration	12 à 99
LSI :Large Scale Integration	100 à 9 999
VLSI : Very Large Scale Integration	10 000 à 99 999
ULSI : Ultra Large Scale Integration	100 000 et plus

2.2 Système sur puce (SoC)

Un système sur puce ou SoC (System-on-Chip) est un circuit intégré où il contient un ensemble de composants matériels et de logiciels conçus et intégrés dans une seule puce électronique pour réaliser les fonctionnalités demandées[3, 4].

2.3 Multiprocesseur système sur puce (MPSoC)

Un MPSoC typique est un système multiprocesseur hétérogène. En effet, il a différents types d'élément de traitements, le réseau d'interconnexion entre les éléments de traitement et la mémoire distribuée autour de la machine peut également être hétérogène.

Les MPSoCs nécessitent souvent de grandes quantités de mémoire. Le périphérique peut disposer d'une mémoire embarquée sur puce, mais il peut aussi s'appuyer sur une mémoire standard hors puce[4, 5].

2.4 Un circuit intégré spécifique à une application (ASIC)

ASIC est un circuit intégré conçu pour une application ou une utilisation particulière, comme dans un lecteur de disque compact ou un système de télécommunications. Les ASIC contrastent fortement avec les produits CI standard tel que les mémoires ou les microprocesseurs qui sont généralement conçus pour une utilisation dans un large éventail d'application.

Les ASIC définissent également un style ou une méthode de conception qui est basée sur l'utilisation extensive d'outils et de systèmes de conception assistée par ordinateur (CAO). Les ASIC sont généralement classés dans l'une des trois catégories suivantes ; entièrement personnalisé, semi-personnalisé et structuré.

Les ASIC entièrement personnalisés sont entièrement conçus sur mesure pour une application particulière dès le départ. Étant donné que la conception et les fonctionnalités ultimes sont pré-spécifiées par l'utilisateur, il ne peut pas être modifié pour s'adapter à différentes applications, et il est généralement produit en tant que produit unique et spécifique pour une application particulière uniquement.

Les ASIC semi-personnalisés, d'autre part, peuvent être en partie personnalisés pour remplir différentes fonctions dans leur domaine d'application général. Ils sont conçus pour permettre un certain degré de modification pendant le processus de fabrication.

Les ASIC structurés ou de plate-forme, une classification relativement nouvelle, sont ceux qui ont été conçus et produits à partir d'un ensemble étroitement défini de méthodologies, de conception, de propriétés intellectuelles et de silicium bien caractérisé, visant à raccourcir le cycle de conception et à minimiser les coûts de développement.

Les avantages des ASICs sont qu'ils permettent une capacité entièrement personnalisée pour le concepteur du système car l'appareil est fabriqué selon des spécifications de conception personnalisées. De plus, pour les conceptions à très grand volume, une implémentation ASIC aura un coût par unité nettement inférieure par rapport à les FPGAs[5, 6, 7].[1]

[1]FPGA sont des dispositifs semi-conducteurs qui peuvent être programmés et reprogrammés selon les exigences d'application

3 Les différents types d'interconnexion dans les systèmes sur puce

La réduction continuelle des dimensions des transistors a ouvert la porte à la conception de circuits complexes multicœurs (Soc). Alors les interconnexions ont subi une évolution structurelle et topologique pour répondre aux nouvelles exigences des systèmes sur puce, parmi ces interconnexions on trouve :

3.1 Point à point

Il s'agit de la connexion la plus simple qui soit pour connecter deux IP (Intellectual Property). Les blocs fonctionnels sont donc reliés entre eux directement sans aucun protocole de gestion de communications [8](Figure 2). Ce type d'interconnexion offre une bande passante très élevé car chaque deux IPs ont un lien dédié. Le problème majeur de cette architecture c'est le nombre important de liens physiques intégrés qui ont un faible taux d'utilisation (environ 10% selon une recherche de Dally et al [9]) et un impact négatif sur le coût de la surface.

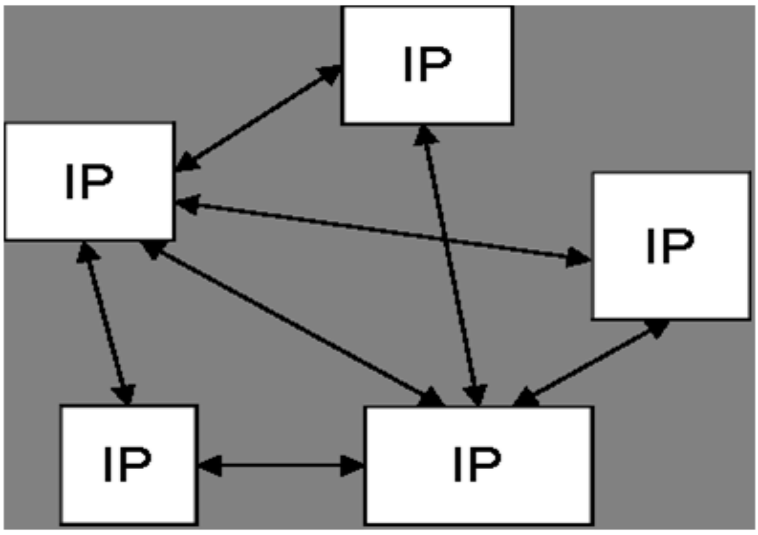


FIG. 1.2 : Point à Point[8]

3.2 Bus

Un bus est une structure de communication reliant deux ou plusieurs modules. Une caractéristique clé d'un bus est qu'il s'agit d'un support de transmission partagé. Plusieurs périphériques se connectent au bus, et un signal transmis par un seul périphérique est disponible à la réception pour tous les autres périphériques connectés au bus. Si deux appareils émettant au cours de la même période, leurs signaux se chevaucheront et deviendront brouillés. Ainsi, un seul appareil à la fois peut transmettre avec succès.

Les structures d'interconnexion les plus courantes reposent sur l'utilisation d'un ou plusieurs bus [10]. Cette architecture présente des avantages comme la simplicité de la topologie, l'extensibilité et le faible coût de surface. Toutefois, il ne permet aucune forme de parallélisme et plus le

nombre d'IP connectées augmente plus la longueur du bus augmente et cela a un impact négatif sur le délai de communication, la consommation d'énergie et la bande passante (Figure 3).

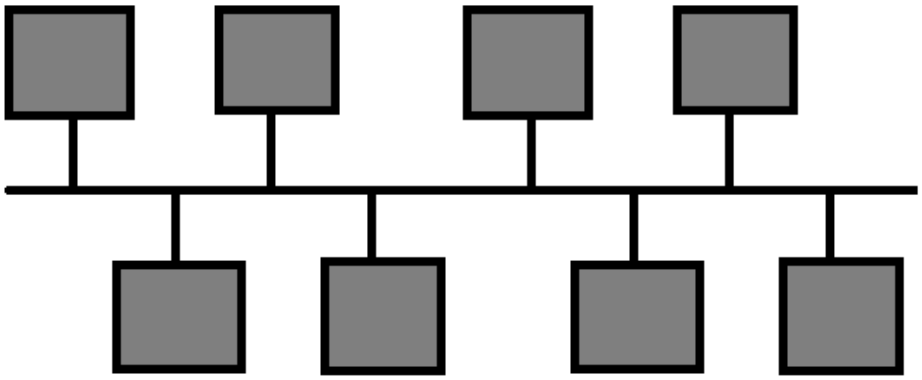


FIG. 1.3 : Bus Partagée

3.3 Bus Hiérarchique

Afin de surmonter le problème de la surcharge d'un bus unique, on a dû avoir recours aux architectures multi-bus pour pouvoir bénéficier de la bande passante nécessaire aux systèmes sur puce complexe [11]. Dans cette architecture le bus est segmenté en plusieurs segments où chaque deux segments sont connectés à l'aide d'une passerelle (voir Figure 4).

Cela a pour conséquence la diminution de la longueur du bus, la consommation d'énergie et la latence. Cette approche permet aussi une certaine forme de parallélisme donc une augmentation des performances du système.

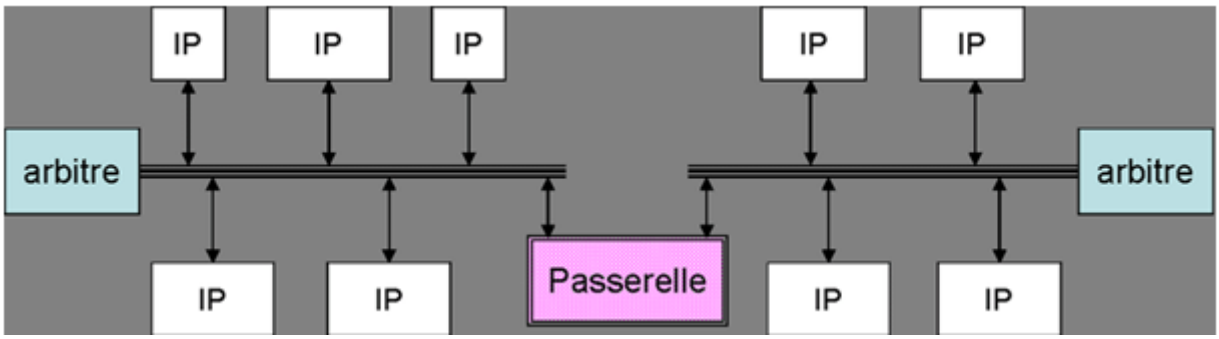


FIG. 1.4 : Bus Hiérarchique[11]

3.4 Réseaux sur Puce

Le réseau sur puce (NoC) est proposé comme une structure de communication évolutive pour les plates-formes à plusieurs cœurs. En raison de ses nombreuses propriétés souhaitables, le NoC s'est rapidement émergé comme un nouveau domaine de recherche dans l'architecture des ordinateurs [12]. Les IPs au sein d'un réseau sur puce communiquent via des routeurs, chaque cœur est attaché à un routeur (switch) via un module d'interface réseau (NI) (Voir Figure 5). Le réseau utilise une communication sur puce à commutation de paquets entre les routeurs. Les réseaux sur puce présentent plusieurs avantages tels que le parallélisme, une faible consommation d'énergie ainsi qu'une faible latence comparée aux autres types d'interconnexions.

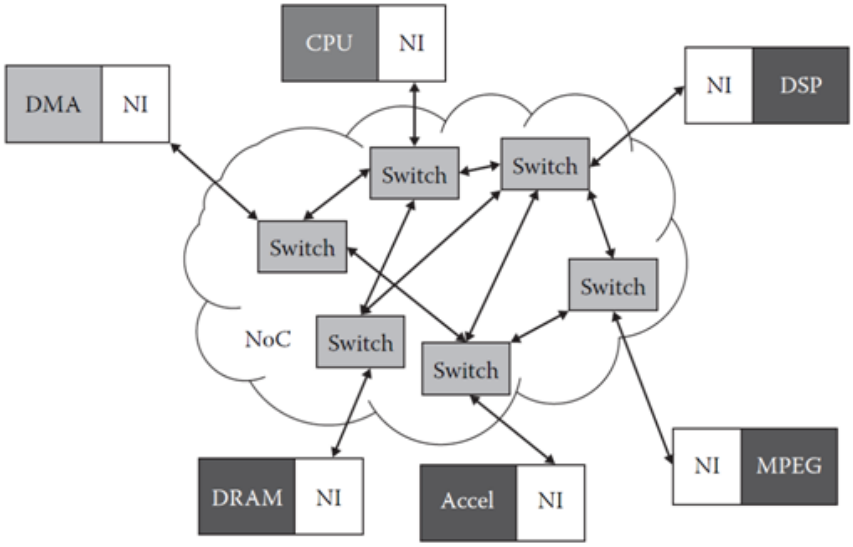


FIG. 1.5 : Exemple d'un Réseaux sur Puce[12]

Une comparaison entre les différents supports de communication qui peuvent être utilisées dans un système sur puce est donnée par le tableau 1.2.

TAB. 1.2 : Comparaison entre les supports des interconnexions[11]

	Point à Point	Bus Partagé	Bus Hiérarchique	NoC
Parallélisme	++	- -	+	++
Réutilisation	-	++	++	++
Consommation	+	-	-	++
Scalabilité	- -	-	-	++

+ + : très bon + : bon - - : très mauvais - : mauvais

En terme de parallélisme le réseau sur puce et les connexions point à point sont les meilleurs, cependant les liens point à point sont dédiés et ne peuvent être réutilisés. Pour la consommation d'énergie il est clair que le NoC est le plus performant vu que les liens utilisés dans les réseaux sont plus courts. De plus, l'avantage principal des systèmes utilisant le NoC est le passage à l'échelle :

on peut ajouter des IPs sans altérer les performances globales du système. Enfin, le réseau sur puce est l'interconnexion la plus adaptée pour les SoCs car en général, les performances globales d'un SoC dépendent fortement du type d'interconnexion utilisé[13].

4 Les composants d'un réseau sur puce

Les NoCs sont composés de routeurs, d'interfaces réseau et des liens physique. (Voir Figure 6)

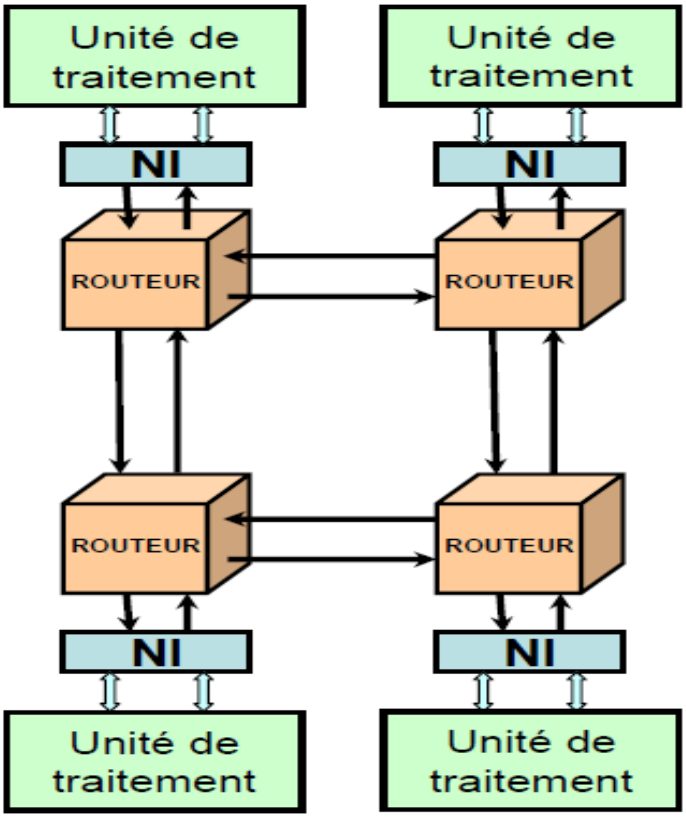


FIG. 1.6 : Topologie d'un NoC[11]

4.1 Routeurs

Les routeurs acheminent les paquets du nœud source vers le nœud de destination en fonction de la topologie réseau sous-jacente et de la stratégie de routage [14]. La microarchitecture du routeur pour un certain réseau sur puce est unique, selon la méthode de commutation mise en œuvre, la qualité de service mise en œuvre et l'algorithme de routage utilisé [15]. En général, un routeur est composé de cinq unités principales :

1. Mémoire tampon FIFO : Elle est utilisée pour garder en mémoire les paquets entrants et sortants dans le routeur.

2. Unité de routage : Elle calcule la décision de routage des paquets entrants. En général, il existe deux implémentations différentes du circuit du moteur de routage. Le premier est Routing State Machine, et l'autre est Table-based Routing[15].

3. Unité d'arbitrage : Elle est utilisée pour sélectionner un paquet à partir d'un certain port entrant pour accéder à son port sortant demandé. Elle joue un rôle d'arbitre pour contrôler les conflits entre certains paquets nécessitant le même port de sortie.

4. Unité de multiplexage cross-bar : Elle définit un réseau de connexions à l'intérieur de routeur en effet chaque port d'entrée doit accéder à tout l'ensemble des ports des sorties, le cross-bar est une unité de multiplexage permettant d'acheminer le paquet du port d'entrée au port de sortie et dont l'entrée de sélection est définie à partir de l'unité de routage [11].

5. Contrôleur de liaison : Il est utilisé pour contrôler la transmission des données entre les ports d'entrée et de sortie des routeurs adjacents. Le contrôle des données est utilisé pour éviter les débordements de données et les réplications de données incorrectes.

4.2 Interface réseau

Une interface réseau traduit les messages envoyés par les modules en requêtes compréhensibles par l'interconnexion [16]. La fonction de l'interface est d'isoler le calcul de la communication [14]. Elle est composée de deux parties, la première partie s'occupe de l'adaptation du protocole de communication utilisé et la deuxième partie est l'interface physique qui lie le bloc IP au routeur (Voir Figure 7).

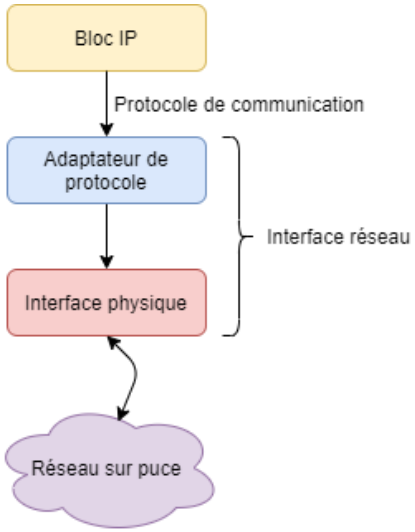


FIG. 1.7 : La connexion d'un bloc IP avec le réseau via l'interface réseau

4.3 Liens physiques

Un lien de communication physique permet de connecter les nœuds de routage et de transférer des données entre eux. Il peut être multiplexé en plusieurs canaux virtuels afin de réduire le taux de congestion lors des communications. Une stratégie d'ordonnancement est donc nécessaire pour affecter la bande passante du lien physique aux différents canaux [17].

5 Topologies standards

Il existe plusieurs types de topologie dans la littérature pour la représentation du réseau sur puce, ces derniers spécifient l'organisation physique du réseau et définissent la disposition structurelle de ses éléments. Parmi les topologies les plus standards on cite[4, 16, 17] :

5.1 Topologie 2D maillé (MESH)

La topologie 2D maillée est la plus utilisée car son arrangement physique et elle offre une grande scalabilité (capacité d'accroître facilement la structure matérielle pour répondre à une exigence de performances). Elle permet d'utiliser des stratégies de routage simple et donc peu coûteuses.(Voir Figure 8[a])

5.2 Topologie tore (torus)

La topologie du tore en 2D (Voir Figure 8[b]) est une topologie très semblable à celle de la topologie 2D maillé. La particularité de celle-ci réside dans les routeurs se trouvant sur les bordures du SoC. Ainsi, tous les routeurs du réseau sont reliés à quatre routeurs, y compris ceux des bordures. Cependant, contrairement à la majorité des canaux du réseau, ceux qui relient les routeurs des bordures étant trop longs peuvent engendrer des latences prohibitives.

5.3 Topologie anneau

Cette topologie offre une latence, une scalabilité et une implantation sur silicium plus faible par rapport à la topologie 2D maillée mais elle n'est pas extensible car ses performances se dégradent à mesure que le nombre d'IPs connectées augmente (Voir Figure 8[c]).

5.4 Topologie arbre

Le réseau arbre est un réseau indirect où les nœuds sont des routeurs et les feuilles sont des IPs connectés au réseau. En effet chaque routeur a des multiples ancêtres qui signifient qu'il y a beaucoup de chemins alternatifs entre eux (Voir Figure 8[d]).

5.5 Topologie 3D maillé (MESH)

La topologie 3D maillée offre des interconnexions globales plus courtes avec une grande bande passante par rapport à 2D, cependant le 3D a comme inconvénient l'implantation sur silicium, routage complexe et le coût élevé dû à l'utilisation des TSVs. (Voir Figure 8[e])

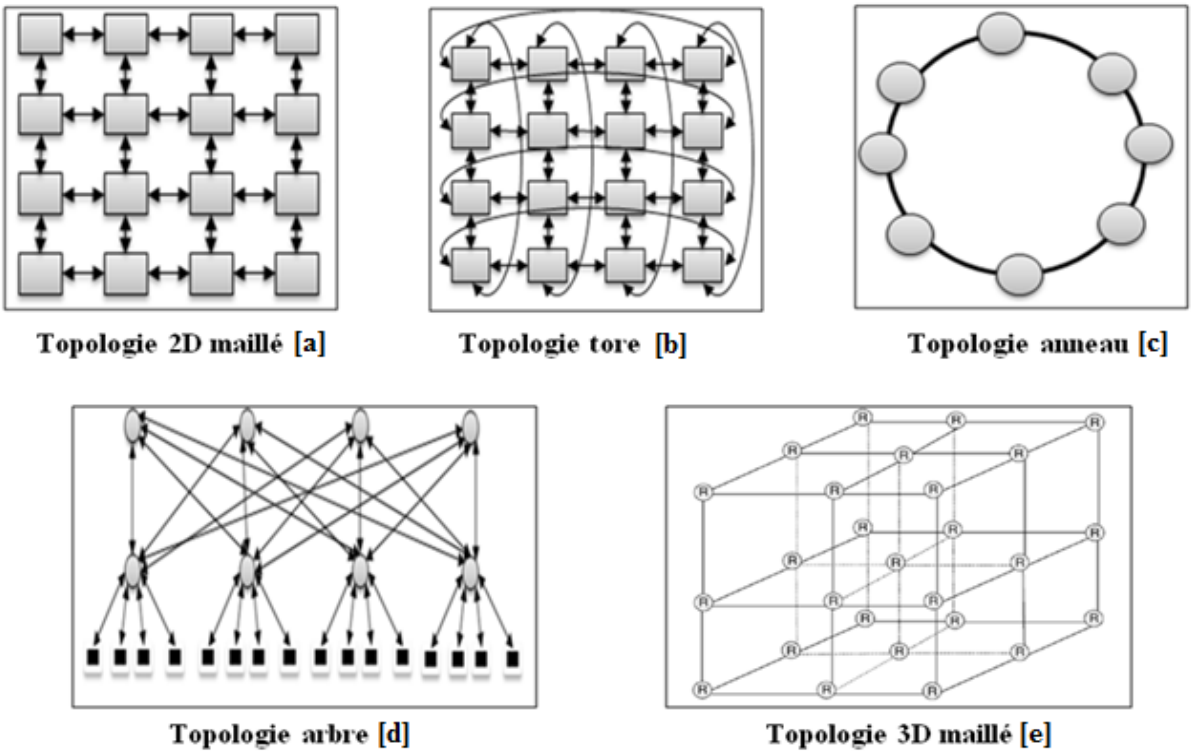


FIG. 1.8 : Exemples de topologies standards[4, 16, 17]

6 Les caractéristiques des NoCs

6.1 Les métriques de performance

La métrique de performance est une quantité mesurable qui saisit précisément ce qu'on veut mesurer, elle peut prendre plusieurs formes. Il n'existe pas de définition générale de la métrique de performance, car cette dernière dépend du système d'où la nécessité de la bonne compréhension de ce dernier et de ses utilisateurs, la connaissance des conditions expérimentales sont également importante[18].

Les métriques de performance sont principalement utilisées afin d'obtenir des informations pertinentes sur la simulation permettant de guider les itérations lors du cycle de développement de l'architecture selon les requis et spécifications initiales, et de valider les compromis de vitesse de simulation versus précision du niveau d'abstraction ciblée.

Lorsque l'on souhaite évaluer les caractéristiques d'une architecture de réseau sur puce, de nombreux critères peuvent être pris en compte. Dans l'idéal, on désire une architecture proposant des débits de transferts élevés, avec une faible latence, en minimisant la consommation d'énergie, le tout occupant une faible surface de silicium[16].

6.1.1 Le débit de données

Chaque élément du réseau (liens, routeurs, etc.) peut être caractérisé par sa bande passante. Cette valeur est exprimée en bit par seconde (bps) n'est pas représentative du fonctionnement réel du réseau. À cause de congestions et de conflits de routage, les éléments du réseau ne peuvent jamais être utilisés au maximum de leur capacité. C'est pourquoi, il est plus pertinent de s'intéresser au débit de données effectif qui est défini comme étant le nombre prévu d'arrivées par seconde. Dans un réseau où il n'y a pas de perte de paquet, il est égal au débit en émission[12, 14, 16].

$$Throughput = \frac{(Total\ packets\ completed * Packet\ length)}{(Number\ of\ IPblocks * Total\ time)}$$

Où :

Total packets completed : est nombre de paquets qui arrivent avec succès à leur cœur IP de destination.

Packet length : se mesure en termes de flits (un message est décomposé en paquets qui sont décomposés en flits).

Number of IP blocks : est nombre de blocs IP impliqué dans la communication.

Total time : indique le temps de simulation (en cycles d'horloge).

6.1.2 La latence de transfert

Elle peut se définir à plusieurs niveaux [16] :

1. Flit : est le temps écoulé entre l'introduction du flit dans le réseau et sa réception par la ressource destinatrice.
2. paquet : est un ensemble de flits, la latence correspond au temps entre l'envoi du premier flit et la réception du dernier.
3. Message : est un ensemble de paquets, c'est le temps entre l'envoi du premier paquet et la réception du dernier.

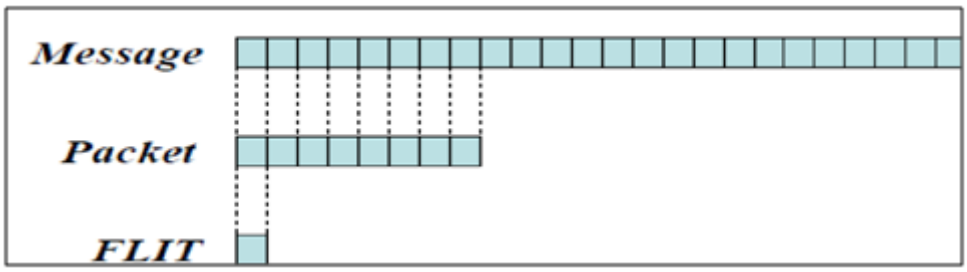


FIG. 1.9 : Format des éléments de base caractérisant les communications dans les NoCs[16]

La latence dépend d'abord de critères statiques. Il s'agit principalement de la position relative de l'émetteur par rapport au récepteur dans le réseau. Plus le nombre de liens et de nœuds à parcourir sera important, plus la latence va augmenter, celle-ci résultant principalement du temps

de routage dans les nœuds. La latence peut également varier de manière dynamique. Cela résulte du partage du réseau entre plusieurs flots de paquets simultanément.

6.1.3 La consommation d'énergie :

La consommation d'énergie est une métrique très importante dans l'étude de performance d'un réseau sur puce et pour déterminer cette dernière, il faut tout d'abord calculer celle de chaque routeur, celle des liaisons et des IPs.

1. Lien : La consommation d'énergie d'un fil est donnée comme suit [5] : $E = \frac{CV^2}{2}$ Où : C : est la capacité ; V : est le voltage.
2. Routeur : On peut déterminer la consommation d'énergie de chaque routeur à l'aide de Synopsys Prime Power.
3. IP : La consommation d'énergie d'un IP dépend de sa nature.

6.1.4 La surface

Du fait du besoin croissant d'augmenter la densité d'intégration dans les SoC, il est intéressant d'évaluer la superficie dédiée aux interconnexions. Le coût en surface d'un réseau sur puce est déterminé par les routeurs ou bien les émetteurs-récepteurs, les points de croisement ainsi que les liens physiques de communication. Généralement, la surface du réseau sur puce en silicium est dominée par celle des routeurs ou des émetteurs-récepteurs, selon le type du NoC.

6.2 Mode de commutation

Afin de gérer les communications au sein d'un réseau sur puce, différents mécanismes de base existent, parmi ces modes de commutation, on peut en retenir deux principaux qui sont : le circuit switching (commutation de circuit) et le packet switching (commutation de paquets)[8].

6.2.1 Commutation de circuit

Ce mode de commutation consiste à établir un circuit dédié au sein du réseau pour chaque paire émetteur-récepteur. 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 et d'énergie car celles-ci sont réquisitionnées tout au long du transfert. Lorsque la communication est terminée, le chemin est dissous[8, 17].

6.2.2 Commutation de Paquets

Il existe plusieurs mécanismes de commutation basés sur le principe de commutation des paquets dont les plus connus sont store and forward, le wormhole et Virtual-cut-through (VCT), ces trois modes de commutation partagent le principe de diviser le message à transmettre en un ensemble de paquets où chaque paquet lui-même est formé par un ensemble de FLIT.

Chaque flit est stocké dans une file d'attente puis transmis sur la voie appropriée. À 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 mais nécessite un contrôle de flux et de congestion[8, 17].

La commutation de circuit est sûre, alors que la commutation de paquets est flexible.

6.3 Algorithme de Routage

L'algorithme de routage est responsable de décider quel chemin le message doit prendre afin d'être acheminé efficacement de sa source à sa destination où il doit éviter les situations d'interblocage tout en optimisant l'utilisation des liens de communications afin d'augmenter la performance du réseau (la latence du réseau et le débit.). Les techniques de routage les plus utilisées dans les structures de réseau sur puce sont X-Y, West First et Negative First.

6.3.1 Routage Ordonné X-Y

L'algorithme XY est un algorithme déterministe et garantit toute situation de blocage (deadlock) . 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.

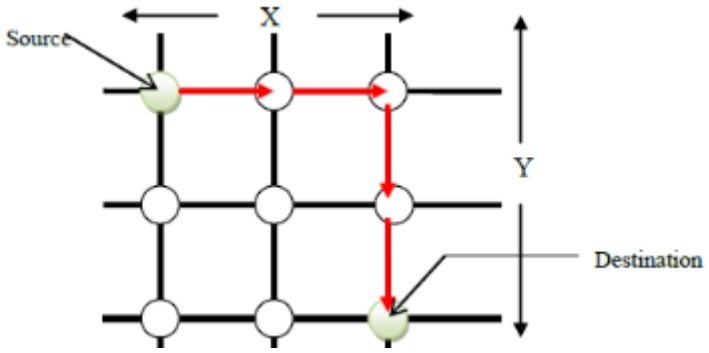


FIG. 1.10 : Routage Ordonné X-Y[8]

6.4 Protocole de communication (contrôle de flux)

Le protocole de communication définit l'ensemble de mécanismes qui contrôlent la communication entre les différentes entités au sein de réseau, il permet de valider et de synchroniser la circulation de données entre ces derniers, d'éviter la surcharge du réseau et régulent le trafic.

Il y a plusieurs techniques de contrôle de flux. Parmi ces techniques :

6.4.1 Contrôle de flux par handshaking

La communication est synchrone où l'émetteur et le récepteur doivent être synchronisés par le même signal d'horloge, l'envoi des données à partir de l'émetteur n'a lieu qu'après la réception d'un acquittement indiquant la liberté du récepteur. Dans le cas de la non-disponibilité du récepteur

(réception d'un acquittement de la non-disponibilité ou absence d'acquittement), l'émetteur reste en attente pour un certain délai ou annule la transmission, selon l'algorithme de routage adopté par ce dernier. Ceci augmente la latence du système et dégrade ses performances (Figure 1.10.a)[17, 19].

6.4.2 Contrôle de flux par credit-based

La communication dans ce cas de protocole est asynchrone, l'envoi des données à partir de l'émetteur a lieu instantanément sans devoir attendre un acquittement de la part du récepteur. 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"(Figure 1.10.b) [17, 20].

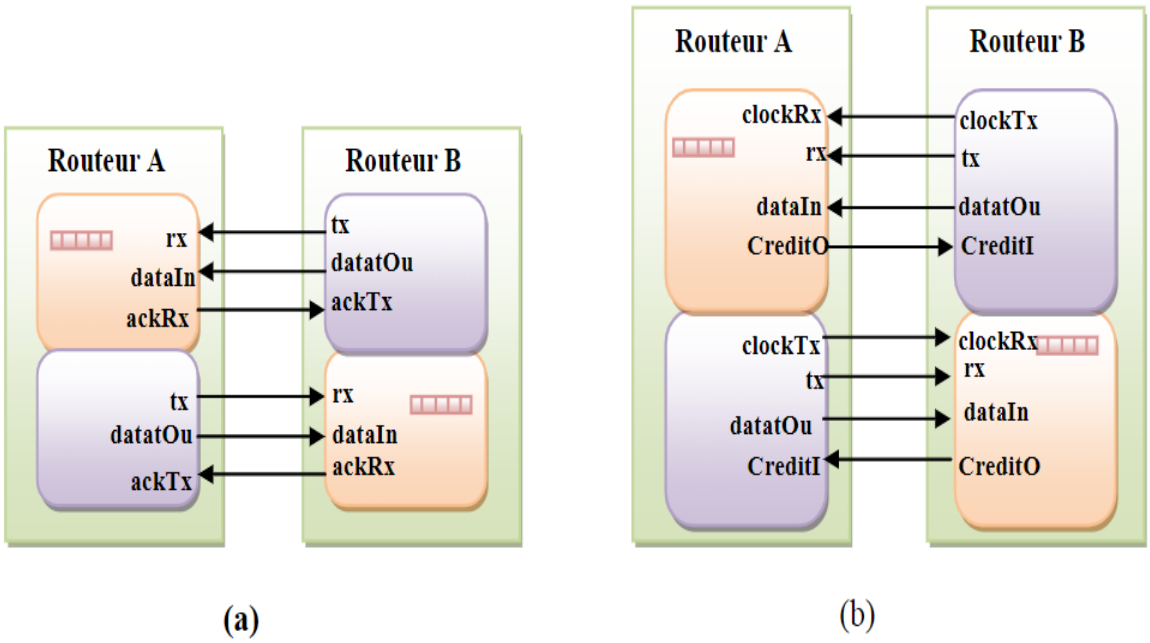


FIG. 1.11 : Stratégies de contrôle de flux[17]

7 Phases de conception

Les performances d'un SoC dépendent fortement du système d'interconnexion entre les unités de calcul. Donc il est critique de concevoir un NoC efficace pour exploiter pleinement le nombre et la puissance de traitement des IPs dans un même circuit. C'est un processus complexe qui passe par plusieurs niveaux d'abstraction.

La conception d'un NoC passe généralement par les phases suivantes :

- Modélisation du trafic généré par l'application ;
- Choix de la topologie adaptée ;

- Assignation des tâches aux IPs ;
- Mapping des IPs sur la topologie ;
- Allocation des chemins de routage et réservation des ressources ;
- Test de performance ;
- Implémentation.

8 Outils d'Aide à la Conception des Réseaux sur Puce

La conception d'un réseau sur puce est une tâche complexe, il est donc nécessaire d'avoir un outil d'aide à la conception pour les NoC. Cet outil peut être automatique ou semi-automatique afin de permettre au concepteur d'avoir la possibilité de forcer des contraintes dans le flot de conception [8]. Dans ce qui suit on va illustrer quelque outil d'aide à la conception d'un NoC.

8.1 Noxim

Cet outil a été proposé par la Computer Architecture team de l'université de Catane. Il est développé en Langage SystemC. Il permet à l'utilisateur de définir une architecture NoC 2D Mesh avec différents paramètres, notamment [21] :

- Taille du réseau ;
- Taille des tampons ;
- Taille du paquet ;
- Algorithme de Routage ;
- Taux d'injection des paquets.

Il permet aussi une évaluation des NoC en termes de débit, de latence et de consommation d'énergie.

8.2 NS-2

NS-2 a d'abord été développé pour le prototypage et la simulation de réseaux informatiques ordinaires. Cependant, comme les NoC partagent de nombreuses caractéristiques avec les réseaux classiques, NS-2 a été largement utilisé par de nombreux chercheurs pour simuler les NoC [22]. Il a été utilisé dans de nombreuses études sur les NoC comme outil de simulation, ce qui en fait une référence fiable notamment pour comparer les performances de deux architectures différentes [23]. Enfin, NS-2 est un simulateur open source piloté par événements discrets et développé en C ++ et OTcl. Cette modularité et cette disponibilité ont facilité sa diffusion entre les chercheurs.

8.3 SunFloor

SunFloor est un outil d'aide la conception des NoCs. Il peut être utilisé dès les premières phases de conception pour synthétiser les topologies avec ces contraintes en entrée (modèle, énergie et Espace, Objectifs de conception). À partir de ces données, SunFloor génère une spécification système prête à être traduite en architecture complète, généralement en langage SystemC et par l'intervention d'un deuxième outil qui est xpipesCompiler[24].

8.4 SunFloor 3D

SunFloor 3D est une extension de la version précédente. La principale fonctionnalité ajoutée est la génération de spécifications pour les NoCs 3D [25]. Les deux versions ont été développées par l'équipe du Prof. Giovanni De Micheli,[26].

8.5 ORION

Il a été proposée par une équipe de l'Université de Princeton en 2003 [27]. Il s'agit d'un simulateur dédié principalement à l'estimation de la puissance et de l'espace pour les architectures NoC. En 2009 ORION 2.0 a été présenté, parmi les améliorations par rapport à la première version, on trouve le support de la nouvelle technologie des semi-conducteurs[28].

8.6 DARSIM

DARSIM est un simulateur NoC développé au Massachusetts Institute of Technology (MIT). Cet outil permet la simulation d'architectures Mesh de 2 et 3 dimensions. Il propose une multitude de configurations de simulation pour NoC avec différents paramètres[29].

TAB. 1.3 : Tableau récapitulatif des outils d'aide à la conception des NoC [21]

Outil	Année	Paramètres					Disponibilité
		TR	TT	TP	AR	TIP	
Noxim	2010	+	+	+	+	+	+
NS-2	1995	+	+	+	+	+	+
SunFloor - 3D	2006 2009	+	-	-	-	-	-
ORION 1 et 2	2003 2009	-	+	-	-	-	-
DARSIM	2009	+	+	+	+	+	-

TR : Taille du réseau. TT : Taille des tampons. TP : Taille des paquets
AR : Algorithme de Routage. TIP : Taux d'injection des paquets.
+ : OUI - : NON

Tous les outils de cette liste ont été présentées par la communauté scientifique à travers des équipes de recherche. Néanmoins, ils ne sont pas tous accessibles, on trouve que Noxim qui est toujours maintenu par ses développeurs et NS-2 avec ça nouvelle version NS-3. Toutefois, il y a d'autres propositions de l'industrie, comme FlexNoC présenté par ARTERIS, The CHAIN works tool suite conçut par Silistix ou encore iNoC. Ces outils sont disponibles mais ils sont payants.

9 Conclusion

Dans ce chapitre sur les réseaux sur puce, on a abordé les raisons pour lesquelles les bus et les connexion point à point trouvent leurs limites dans les systèmes sur puce actuels. Ces limitations étant liées au faible parallélisme des communications mais également à une limitation en terme de consommation énergétique. C'est pourquoi les NoC apparaissent naturellement pour proposer des solutions à ces critères limitatifs des topologies bus.

En effet, les NoC offrent de grandes perspectives de part leur structure mais également par la possibilité de proposer une qualité de service pour les communications de l'application. Ces réseaux peuvent avoir des topologies différentes mais également des gestions de flux de données différentes influant directement sur la complexité matérielle du NoC mais également sur la latence du transport des données. Ces NoC sont donc extrêmement paramétrables. Ainsi, de nombreux paramètres sont à prendre en compte lors de leur conception, car chacun d'entre eux peut influencer directement ou indirectement les performances globales de l'application.

Tous ces paramètres engendrent nécessairement une complexité de mise en oeuvre accrue en comparaison avec les SoC en topologie bus. Afin de proposer aux concepteurs des réseaux sur puce un outil d'aide de conception qui maximise plusieurs performances des NoCs, et qui sont parfois contradictoires, nous traiterons dans ce projet certaines phases de conception dans un cadre multiobjectif. Pour cette raison, nous étudions les techniques d'optimisation multiobjectif et les méthodes dédiées à leurs résolutions dans le chapitre suivant.

Chapitre 2

Optimisation multiobjectif : état de l'art

1 Introduction

Dans le cas de l'optimisation mono-objectif, la solution optimale est facilement définie. Cependant, il existe des problèmes avec plusieurs objectifs, donc il consiste à trouver l'optimum qui est un assortiment de solutions et qui correspond le mieux aux préférences du décideur parmi les solutions de bon compromis.

L'optimisation multiobjectif occupe une place très importante en recherche opérationnelle, en mathématiques discrètes et en informatique, à cause de la nature multiobjectif de la plupart des problèmes réels. En effet ce domaine aide à la décision multicritère. Et d'après la littérature, et spécialement dans le domaine industriel la plupart des problèmes nécessitent une optimisation multiobjectif. Pour les NoCs, trouver une combinaison optimale des paramètres afin de maximiser ces performances est un problème d'optimisation multiobjectif.

De ce fait, ce chapitre sera consacré à l'étude des méthodes d'optimisation multiobjective.

2 Optimisation multiobjectif

2.1 Concepts et Définitions

Dans le point de vue informatique (Computational Problem) veut dire comment faire relier un ensemble de données par un ensemble de résultats, où les données vont subir un ensemble d'opérations dont on les appelle le traitement. La complexité (temporelle) d'un problème est le nombre d'opérations élémentaires (affectations, comparaisons, opérations arithmétiques) pour trouver une solution. Ce nombre s'exprime en fonction de la taille n des données[30].

2.1.1 Un problème combinatoire

Un problème combinatoire est toute situation dont on cherche à avoir une solution tout en respectant la présence d'un ensemble de contraintes. La solution c'est une combinaison de ces

contraintes d'une manière qu'on maximise quelques uns et on minimise les autres, ces contraintes ont une caractéristique primordiale, c'est que chaque contrainte influe sur les autres soit quand on minimise sa valeur ou on la maximise, dans un autre terme on dit que les contraintes sont conflictuelles.

2.1.2 Un problème d'optimisation

Les problèmes d'optimisation ont été définis par un espace d'état, une ou plusieurs fonction(s) objective(s) et un ensemble de contraintes[31].

1. L'espace d'état est défini par l'ensemble de domaines de définition des variables du problème.
2. Les variables du problème peuvent être de nature diverse (réelle, entière, booléenne, etc.).
3. Une fonction objective représente le but à atteindre pour le décideur (minimisation de coût, de durée, d'erreur, ...). Elle définit un espace de solutions potentielles au problème.
4. L'ensemble de contraintes définit des conditions sur l'espace d'état que les variables doivent satisfaire. Ces contraintes sont souvent des contraintes d'inégalité ou d'égalité et permettent en général de limiter l'espace de recherche.
5. Une méthode d'optimisation cherche le point ou un ensemble de points de l'espace d'état possible qui satisfait au mieux un ou plusieurs critères. Le résultat est appelé valeur optimale ou optimum [31].
6. Optimum / valeur optimale est le résultat obtenu à la fin du processus (minimum ou bien le maximum).

2.2 Le problème d'optimisation multiobjectif

Les premiers travaux menés sur les problèmes multiobjectifs furent réalisés au 19éme siècle sur des études en économie par Edgeworth[32] et généralisés par Pareto[33]. Il peut être défini comme un problème dont on cherche l'action qui satisfait un ensemble de contraintes et optimise un vecteur de fonctions objectives.

En effet, dans la plupart des problèmes de conception plusieurs objectifs doivent être optimisés simultanément pour aboutir à une solution satisfaisante[34]. Le caractère multiobjectif du problème est donné par l'existence d'objectifs contradictoires signifiants que l'amélioration d'un objectif implique la détérioration d'un autre. Ce conflit entre les objectifs est souvent rencontré lorsque l'on cherche à obtenir les meilleures performances possibles pour un faible coût. Plus une structure sera complexe et performante plus son coût sera élevé[35]. L'optimisation multiobjectif consiste alors à rechercher l'ensemble des solutions qui correspondent aux meilleurs compromis entre les objectifs à maximiser et de coût à minimiser [36, 37].

2.3 La dominance

Lorsque l'on cherche à obtenir une solution optimale à un problème d'optimisation multiobjectif donné, on s'attend souvent à trouver une solution et une seule. En effet, on rencontre rarement ce cas de figure. La plupart de temps, on trouve une multitude de solutions, car certaines des objectifs sont contradictoires. Donc, quand on résoudra un problème d'optimisation multiobjectifs, on obtiendra une grande quantité de solutions. Ces solutions comme on peut s'en douter, ne seront pas optimales.

Un concept intéressant, qui nous permettra de définir les solutions obtenues, est le compromis. En effet, les solutions que l'on obtient lorsqu'on a résolu le problème sont des solutions de compromis. Elles minimisent un certain nombre d'objectifs tout en dégradant les performances sur d'autres objectifs. Il est vital pour identifier ses meilleurs compromis de définir une relation d'ordre entre ces éléments. Dans le cas des problèmes d'optimisation multiobjectifs, ces relations d'ordre sont appelées relations de dominance. Plusieurs relations de dominance ont déjà été présentées : la A-dominance [38], la dominance au sens de Geoffrion [39], la cône-dominance [30], . . . Mais la plus célèbre et la plus utilisée est la dominance au sens de Pareto[33].

2.3.1 La dominance au sens de Pareto

On dit qu'une solution faisable X domine une autre solution faisable Y au sens de Pareto si :

- X est au moins aussi bon que Y dans tous les objectifs, et,
- X est strictement meilleur que Y dans au moins un objectif.

Les solutions qui dominent les autres mais ne se dominent pas entre eux sont appelées solutions optimales au sens de Pareto (ou solutions non dominées).

2.3.2 Front de Pareto

Dans l'optimisation multiobjectif, on a besoin du concept de dominance pour dire quand une solution est meilleure qu'une autre (ou si aucune ne l'est). La solution optimale d'un problème d'optimisation multiobjectif est connue sous le nom de front de Pareto, qui est un ensemble de solutions, et non une solution unique comme dans l'optimisation mono-objectif.

3 Approches de résolution multiobjectif

Dans différentes publications et recherches il existe deux classifications différentes des approches de résolution de problèmes multiobjectifs. Le premier classement adopte un point de vue décideur et l'autre adopte un point de vue concepteur.

3.1 Classification"Point de vue de Décideur"[40]

Les approches sont classées en fonction de l'usage que l'on désire en faire. On distingue trois schémas possibles [40, 41] :

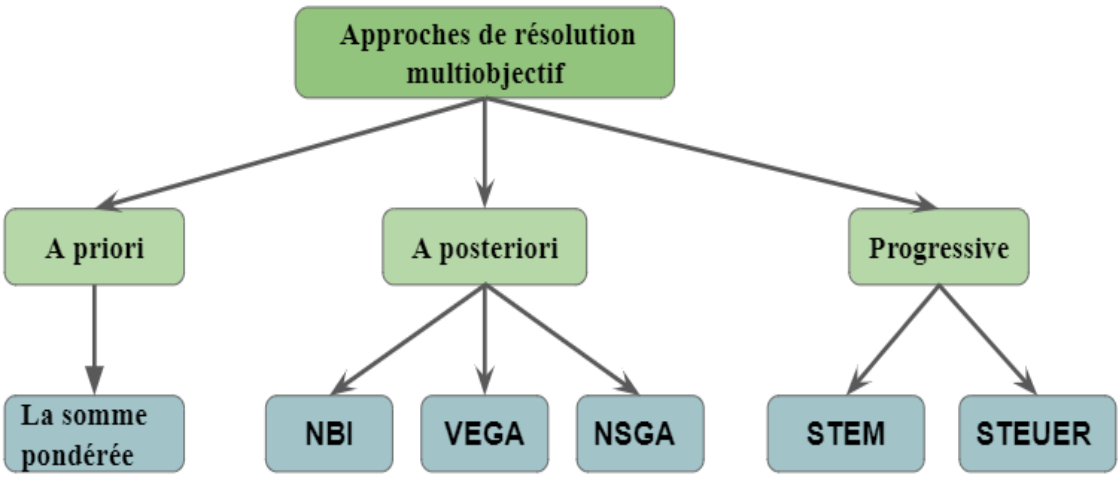


FIG. 2.1 : Les approches de resolution du point de vue de décideur [40]

3.1.1 Approches à priori

Elle caractérise les méthodes a priori utilisée à cause de leur simplicité de mise en oeuvre, en effet les objectifs du problème d'optimisation sont transformés en une seule fonction objective (mono-objectif).

Parmi les méthodes utilisées on a la somme pondérée "SP" c'est une méthode simple à mettre en oeuvre son principe est d'affecter aux fonctions objectifs des poids relatifs à l'importance de chacun des objectifs. Une difficulté de cette approche réside dans le choix des poids en fonction des préférences des experts.

3.1.2 Approches à posteriori :

Dans cette approche l'expression des préférences et la pondération des objectifs se font a posteriori. C'est-à-dire que le processus d'optimisation détermine un ensemble de solutions candidates et la sélection de la meilleure solution ne se fait qu'à la fin du processus.

Parmi les approches a posteriori NBI : détermine les solutions optimales par rapport à chaque objectif ; l'ensemble des solutions optimales est défini par l'espace délimité par les segments de droites reliant les solutions optimales. Les approches a posteriori les plus utilisées sont celles basées sur les algorithmes génétiques, appelés aussi algorithmes évolutionnaires. On trouve différentes formulations de ces approches, comme VEGA, HLGA, NSGA.

3.1.3 Approche progressive :

Le principe de l'approche progressive est de guider l'exploration tout au long du processus d'optimisation, est d'alterner entre le processus de recherche et le processus de décision défini par un classement de solutions, ou bien un choix des poids de pondération des objectifs. À chaque étape, l'ensemble des solutions est analysé afin d'orienter les futures itérations vers les zones les plus intéressantes de l'espace de recherche. C'est pour cela qu'elles sont également appelées méthodes interactives. Les méthodes les plus connues de cette classe sont la méthode STEM, aussi appelée STEP et celle de Steuer.

3.2 Classification"Point de vue de Concepteur"[36]

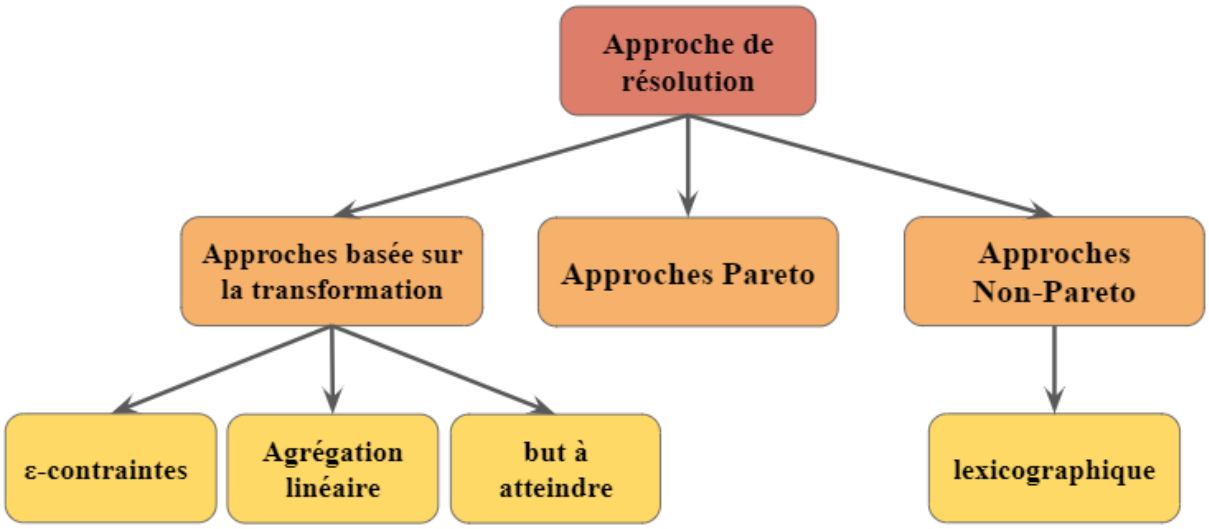


FIG. 2.2 : Les approches de resolution [42]

3.2.1 Les approches basée sur la transformation

Elle consiste à transformer un problème multiobjectif en un problème mono-objectif. Parmi ces méthodes :

3.2.1.1 L'agrégation linéaire

Cette méthode populaire transforme le problème multiobjectif en un problème mono-objectif en combinant linéairement les différents objectifs. Elle est intéressante pour des problèmes ayant de nombreux objectifs et/ou un grand nombre de solutions supportées bien réparties. Dans ce contexte, il peut être suffisant de générer les solutions supportées[44].

3.2.1.2 Approches ε-contraintes

Cette méthode permet de transformer le problème d'optimisation multiobjectif en un problème mono-objectif. La méthode consiste à convertir (m-1) des m objectifs du problème en contraintes et d'optimiser séparément l'objectif restant.

La démarche est la suivante :

- Choisir un critère à optimiser prioritairement.
- Garder l'objectif prioritaire et transformer les autres objectifs en des contraintes d'inégalité.

3.2.1.3 La méthode du but à atteindre "GP"

Une solution dite utopique S* était fixée au départ par le décideur. Le problème d'optimisation détermine la meilleure solution celle qui se rapproche le plus de S*.

Afin de comparer différentes solutions, plusieurs variantes de cette approche existent. L'une, appelée archimédienne, utilise la somme pondérée afin d'avoir une seule fonction objectif. Une deuxième variante appelée lexicographique, classe les objectifs du problème suivant un ordre de priorités [40].

3.2.2 Les approches non Pareto

Ces approches sont de type à posteriori, elles ne traitent pas le problème comme un véritable problème multiobjectif. Cependant elles cherchent à ramener le problème initial à un ou plusieurs problèmes mono-objectifs, parmi ces méthodes on cite la méthode lexicographique[36].

Dans la méthode lexicographique le décideur classe les objectifs par ordre de préférence. Le processus d'optimisation commence par rechercher la meilleure solution en fixant l'objectif prioritaire jusqu'à atteindre un optimum. En cas d'égalité de plusieurs solutions, la comparaison se fait par rapport à l'objectif de priorité inférieure [40, 43].

3.2.3 Les approches Pareto :

Elles ne transforment pas les objectifs du problème, ceux-ci sont traités sans aucune distinction pendant la résolution. Elles utilisent directement la notion d'optimalité de Pareto dans leur processus de recherche[36].

4 Méthodes de résolution

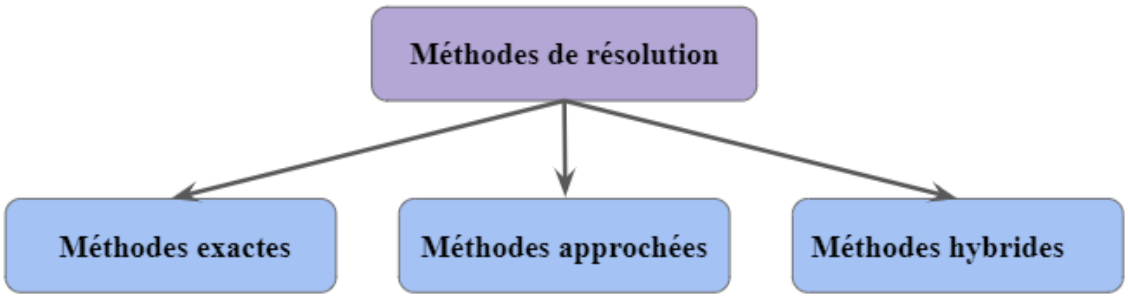


FIG. 2.3 : Classification d'algorithme de résolution d'un problème multiobjectif [42]

4.1 Méthode de resolution éxacte

Les méthodes exactes permettent de trouver des solutions optimales pour des problèmes de taille raisonnable et rencontrent généralement des difficultés face aux applications de taille importante. Parmi les méthodes de résolution exacte on trouve : branch and bound [45, 46] ; A* [47, 48] ; Programmation dynamique [49, 50].

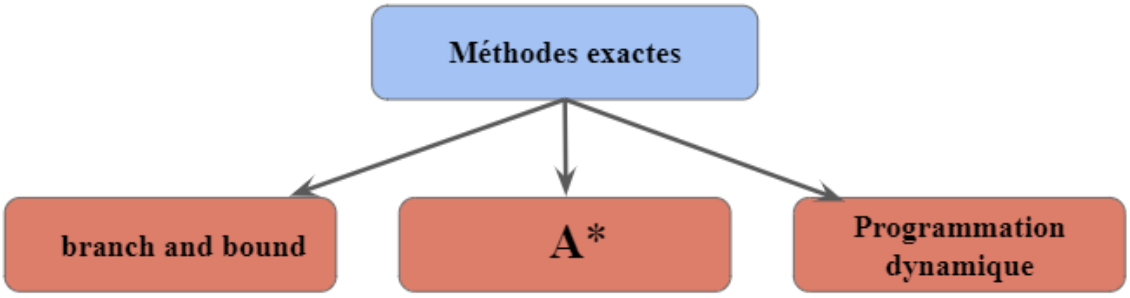


FIG. 2.4 : Les méthodes de résolution exacte[42]

4.1.1 Branch and bound

Est une méthode générique de résolution exacte de problèmes d'optimisation, et plus particulièrement d'optimisation combinatoire.

Elle repose sur une méthode arborescente de recherche d'une solution optimale par séparations et évaluations, dans laquelle on effectue les étapes suivantes [30, 36, 51] :

1. On divise l'espace de recherche en sous-espaces.
2. On cherche une borne minimale en terme de fonction objectif associée à chaque sous-espace de recherche.
3. On élimine les"mauvais" sous-espaces.
4. On reproduit les étapes précédentes jusqu'à obtention de l'optimum global.

4.1.2 Multiobjectif A^*

Un multiobjectif A * est appelé MOA *. Il prend en compte la nature de la majorité des problèmes impliquant des objectifs multiples, contradictoires et non proportionnés en identifiant l'ensemble de tous les chemins non dominés d'un nœud de départ spécifié à un ensemble donné de nœuds de but dans un graphique OU.

L'algorithme doit suivre les étapes suivantes :

1. Choisir un ensemble des nœuds qui égal à l'ensemble des nœuds de départ.
2. Terminer ou sélectionner un nœud en fonction de son coût de solution .
3. Éliminer les mauvais chemins .
4. Placer tous les chemins de solution dans un ensemble des nœuds nommées SOLUTION.

5. Effectuer la comptabilité pour maintenir les coûts courus de SOLUTION.
6. Identifier les solutions.
7. Répéter jusqu'à obtention de l'optimum global.

4.1.3 Programmation dynamique

La technique de programmation dynamique est une approche générale qui était introduite par Bellman dans les années 1940 où elle apparaît comme un outil utile pour résoudre divers problèmes en optimisation combinatoire. Elle consiste à décomposer un problème en sous problèmes plus simples, ensuite résoudre c'est sous problèmes et combiner leurs solutions afin de trouver une solution globale[40, 51].

4.2 Méthodes de résolution approchées

Une alternative raisonnable aux méthodes exactes pour résoudre les problèmes d'optimisation multiobjectifs est de dériver une méthode approchée. Une méthode approchée dans un contexte d'optimisation multiobjectif est une méthode qui trouve soit des ensembles de solutions locales potentiellement efficaces, qui sont ensuite fusionnés pour former un ensemble de solutions potentiellement efficaces, ou des ensembles de solutions globale potentiellement efficace[52].

L'heuristique à objectifs multiples et la métaheuristique à objectifs multiples sont des méthodes qui visent à fournir un bon compromis entre une approximation de l'ensemble de solutions efficace et les exigences de temps et de mémoire pour les obtenir.

- **Une heuristique**

Une heuristique est définie par Reeves (1995) comme une technique qui cherche de bonnes solutions (c'est-à-dire quasi optimales) à un coût de calcul raisonnable sans pouvoir garantir l'optimalité ou la faisabilité. Souvent, l'heuristique est spécifique au problème, de sorte qu'une méthode qui fonctionne pour un problème ne peut pas être utilisée pour en résoudre un autre[52].

- **Une métaheuristique**

Une métaheuristique est une méthode générale de recherches dédiées aux problèmes d'optimisation difficile[53]. Ces méthodes sont, en général, présentées sous la forme de concept d'inspiration. Comme nous le verrons plus tard, elles reprennent des idées que l'on retrouve parfois dans la vie courante. Ces méthodes ont des inspirations de l'éthologie comme les colonies de fourmis, de la physique comme le recuit simulé, et de la biologie comme les algorithmes évolutionnaires.

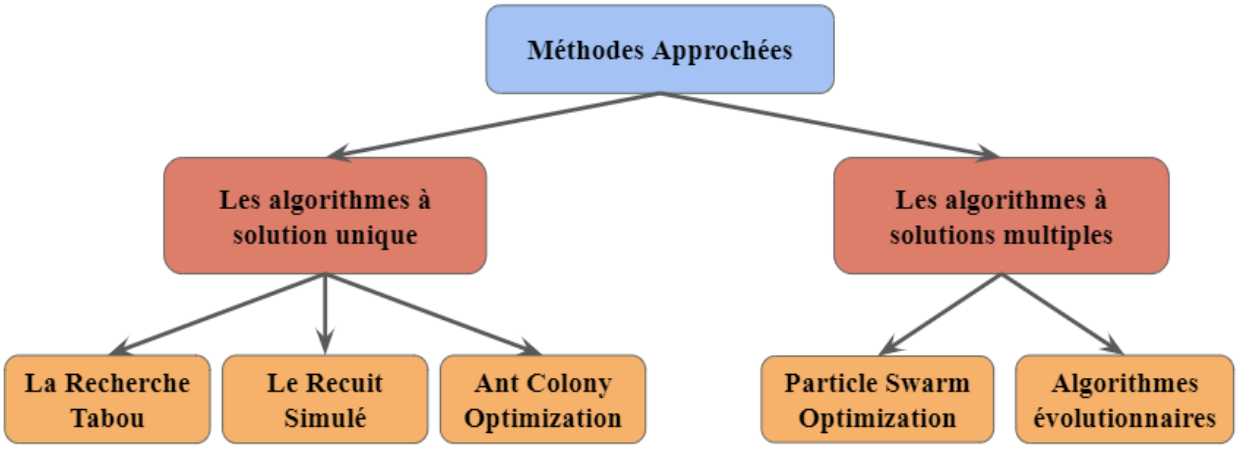


FIG. 2.5 : Méthodes de résolution approchées[42]

4.2.1 Les algorithmes à solution unique

4.2.1.1 Le Recuit Simulé

L'origine de la méthode du Recuit Simulé provient de la métallurgie, où, pour atteindre les états de basse énergie d'un solide, on chauffe celui-ci jusqu'à des températures très élevées, après on le laisse refroidir lentement. Ce processus est appelé le recuit. La méthode du recuit simulé a été développé simultanément par Kirk en 1983 et Cerny en 1985[54]. Ce processus métallurgique a été transposé à l'optimisation et a donné une méthode simple et efficace.

Le fonctionnement de cet algorithme est le suivant :

1. On commence par choisir un point de départ au hasard (X).
2. On calcule un voisin de ce point (Y = Voisin (X)).
3. On évalue ce point voisin et on calcule l'écart par rapport au point d'origine (ΔC = C(X) − C(Y)) .
4. Si cet écart est négatif, on prend le point Y comme nouveau point de départ, s'il est positif on peut quand même accepter le point Y comme nouveau point de départ, mais avec une probabilité $e^{\frac{-\Delta C}{T}}$ (qui varie en sens inverse de la température T).
5. Au fur et à mesure du déroulement de l'algorithme, on diminue la température T(T = (T)).
6. On répète toutes ces étapes tant que le système n'est figé (par exemple, tant que la température n'a pas atteint un seuil minimal).

4.2.1.2 La Recherche Tabou

La recherche Tabou est une métaheuristique développée par Glover ([Glover, 86], [Glover , 90], [Glover, 97]). Elle est basée sur des idées simples, mais elle est néanmoins très efficace. Elle combine une procédure de recherche locale avec un certain nombre de règles et de mécanismes pour surmonter l'obstacle des optima locaux, tout en évitant de cycler. Elle a été appliquée avec succès pour résoudre de nombreux problèmes difficiles d'optimisation combinatoire.

Le principe de la méthode Tabou est simple :

On part d'une solution initiale x, puis on cherche le meilleur voisin s(x) dans le voisinage de x [s(x)], l'originalité de la méthode réside dans le fait que l'on retient le meilleur voisin de x même si celui-ci est plus mauvais que x. Ce critère autorisant les dégradations de la fonction objectif évite à l'algorithme d'être piégé dans un minimum local, mais il induit un risque de cyclage. Pour régler ce problème, l'algorithme mémorise la liste des transformations interdites (telle que de s(x) à x) dans la liste Tabou interdisant la transformation inverse d'une transformation faite récemment[55]. Certains présentent cette méthode comme une alternative au recuit simulé.

4.2.1.3 L'optimisation par colonies de fourmis (Ant Colony Optimization) :

Les colonies de fourmis "ACO", et plus généralement les insectes sociaux, sont des systèmes distribués qui, dans l'aspect de la simplicité de leurs individus, présentent une forte structure d'organisation sociale. Comme un résultat de cette organisation, les colonies de fourmis peuvent accomplir les tâches complexes qui sont en quelque sorte loin dépassées les capacités d'individus pour une seule fourmi. Le champ des algorithmes de fourmis étudie les modèles dérivés à partir de l'observation du comportement réel de fourmis, et utilise ces modèles comme une ressource d'inspiration pour la conception de nouveaux algorithmes pour la résolution des problèmes d'optimisation de contrôle distribué [56]. "ACO " est une métaheuristique qui a été utilisée avec succès pour résoudre plusieurs problèmes d'optimisation combinatoires. Toutefois, la plupart des travaux ont consisté à résoudre des problèmes à objectif unique.

4.2.2 Les algorithmes à solutions multiples :

4.2.2.1 L'optimisation par Essaim Particulaire (Particle Swarm Optimization) :

L'optimisation par Essaim Particulaire (OEP, ou PSO en Anglais) s'inspire du comportement social des animaux évoluant en essaim tel que les bancs de poissons par exemple. Cette méthode a été proposée par Kennedy et Eberhart en 1995[57]. Au départ, les deux chercheurs voulaient simuler la capacité des oiseaux à voler d'une façon synchrone et leur aptitude à changer brusquement de direction tout en gardant une formation optimale. Le modèle qu'ils ont propose a ensuite été étendu en un algorithme simple et efficace d'optimisation[58]. On observe dans ce type d'organisme un déplacement relativement complexe du groupe, alors que, chaque individu dispose d'une connaissance strictement locale de sa situation dans l'essaim, et a une intelligence très limitée[59]. Grâce à une connaissance de sa propre position dans l'essaim et des informations sur la vitesse et la position de ses voisins, un individu peut décider de son propre déplacement. Ces déplacements s'effectuent en employant des règles simples telles que : **"aller a la même vitesse que les autres"**, **"se déplacer dans la même direction"** ou encore **"rester proche de ses voisins"**, qui maintiennent la cohésion de l'essaim, et permettent la mise en oeuvre de comportements collectifs complexes et adaptatifs. Ce comportement collectif propre à ce type d'animaux qui vivent en essaims, et qui se base sur l'analyse de l'environnement et du voisinage pour effectuer des choix optimaux, constitue alors, une méthode d'optimisation par l'observation des tendances des individus voisins. Chaque individu cherche à optimiser ses chances en suivant une tendance qu'il modérée par ses propres vécus[57].

4.2.2.2 Les algorithmes évolutionnaires :

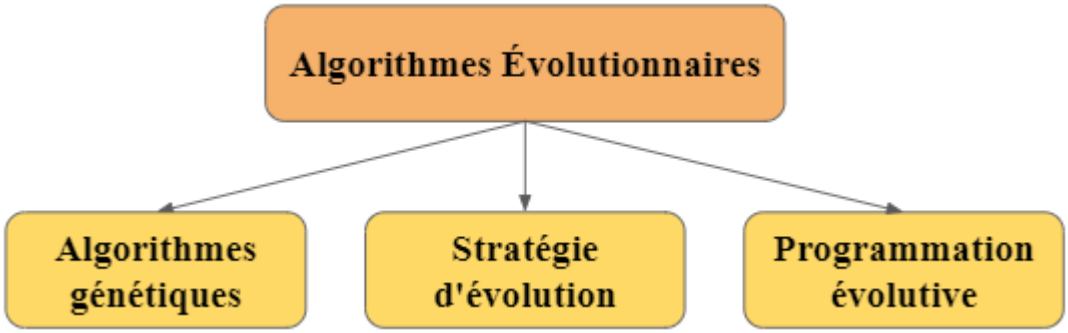


FIG. 2.6 : Les Algorithmes évolutionnaires [60]

On peut distinguer trois grandes classes d'algorithmes évolutionnaires : les algorithmes génétiques [Holland, 1975 ; Goldberg, 1989], les stratégies d'évolution [Schwefel, 1981] et la programmation évolutive [Fogel, 2000]. Ces méthodes se distinguent par la manière de représenter l'information et par la façon de faire évoluer la population d'une génération à l'autre[60]. Un algorithme évolutionnaire est typiquement composé de trois éléments fondamentaux :

- Une population constituée de plusieurs individus représentant des solutions potentielles (configurations) du problème donné.
- Un mécanisme d'évaluation des individus permettant de mesurer l'adaptation de l'individu à son environnement.
- Un mécanisme d'évolution de la population permettant, grâce à des opérateurs prédéfinis, d'éliminer certains individus et d'en créer de nouveaux.

Parmi les composants d'un algorithme évolutionnaire, l'individu et la fonction d'évaluation correspondent respectivement à la notion de configuration et à la fonction d'évaluation dans les méthodes de voisinage. Le mécanisme d'évolution est composé de plusieurs opérateurs tels que la sélection, la mutation et le croisement.

La sélection a pour objectif de sélectionner des individus qui vont pouvoir se reproduire pour transmettre leurs caractéristiques à la génération suivante. Le croisement ou recombinaison est un opérateur permettant de construire de nouveaux individus enfants à partir des caractéristiques d'individus parents sélectionnés. La mutation effectue des légères modifications de certains individus.

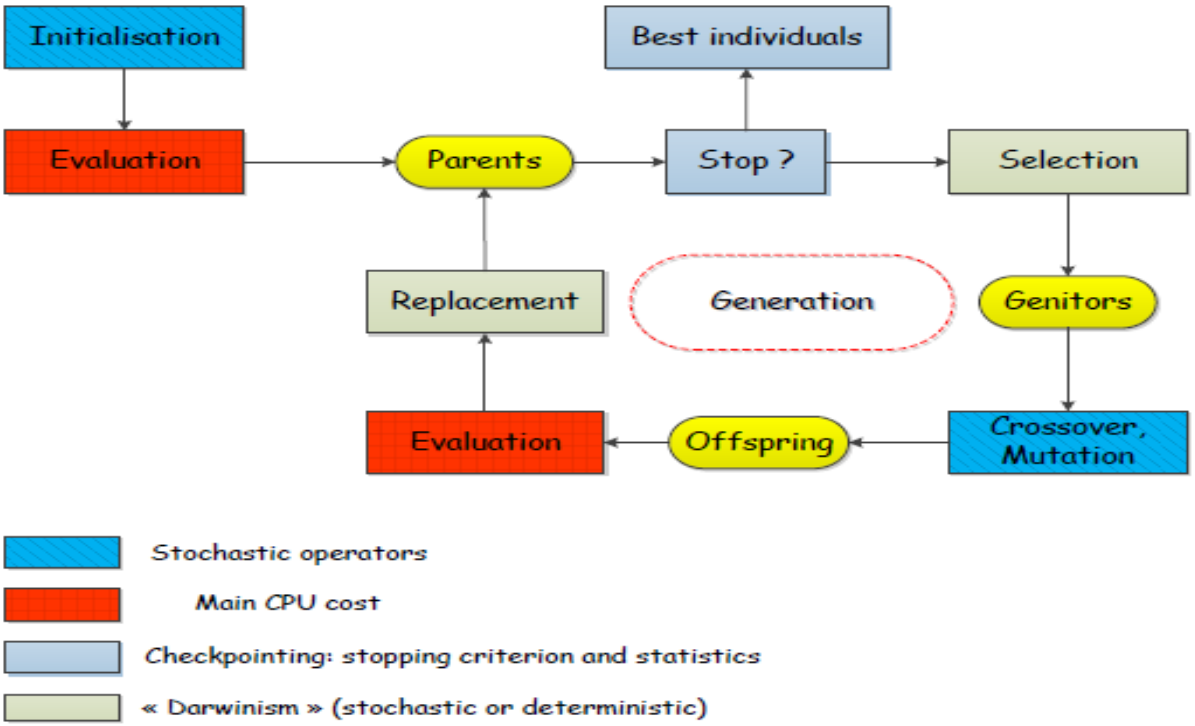


FIG. 2.7 : Architecture d'un algorithme évolutionnaire[59]

4.2.2.2.a Algorithmes génétiques " AG " :

Élaborés par J. Holland, 1975 et D.E. Goldberg, 1989, Michigan, USA. Les AG ont été imaginés comme outils de modélisation de l'adaptation. Ce sont les plus connus des algorithmes évolutionnaires.

Les algorithmes génétiques sont des algorithmes d'exploration fondés sur les mécanismes de la sélection naturelle et de la génétique. Ils sont basés sur les principes de survie de structures les mieux adaptés et les échanges d'information. À chaque génération, un nouvel ensemble de créatures artificielles (codées sous forme de chaînes de caractères) est construit à partir des meilleurs éléments de la génération précédente. Bien que reposant fortement sur le hasard (et donc sur un générateur de nombres aléatoires) ces algorithmes ne sont pas purement aléatoires[58].

4.2.2.2.b Stratégies d'évolution "ES" :

Inventées par I. Rechenberg et H.P. Schwefel, 1981, Berlin. Les ES ont été mises au point par deux jeunes ingénieurs travaillant sur des problèmes numériques. Un énorme progrès a été apporté par les techniques adaptatives d'ajustement des paramètres de mutation, et ce sont sans contestation les meilleurs algorithmes pour les problèmes purement numériques.

4.2.2.2.c Programmation évolutive "EP " :

Inventée par L.J. Fogel, 1966 et D.B. Fogel, 1995-2000, Californie, USA. Imaginée par Larry Fogel pour la découverte d'automates à états finis pour l'approximation de séries temporelles, EP a rapidement travaillait sur des espaces de recherche très variés.

4.3 Méthodes de résolution hybrides

Clairement les différentes méthodes ont différents avantages et il est donc attractif de produire des méthodes hybrides[61]. Classifie les algorithmes hybrides en quatre catégories, qui vont pour classifier d'autres hybridations comme suit :

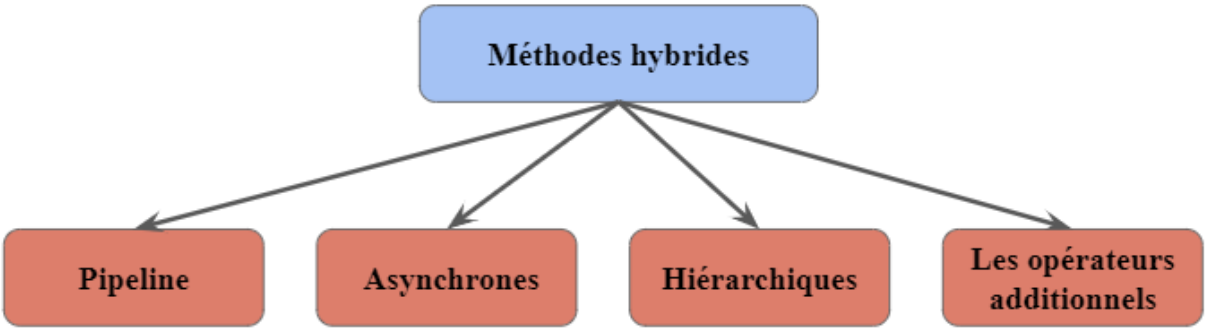


FIG. 2.8 : Les méthodes hybrides[61]

4.3.1 Les méthodes hybrides en pipe-line

C'est la plus simple et la plus directe manière pour l'implémentation hybride, premièrement, on commence par explorer tout l'espace de recherche avec une méthode comme l'identification de l'optimum global mais peut être avec une convergence lente. Après l'identification des régions prometteuses, on peut se reporter à une méthode avec une convergence rapide pour faire accélérer la recherche. Ça peut être accomplis dans l'optimisation multiobjectifs par une combinaison d'une instance d'un algorithme génétique avec un algorithme de gradient.

4.3.2 Les méthodes hybrides asynchrones

Les différentes méthodes hybrides asynchrones travaillent en différents sous ensembles d'espace de solutions. Les différentes méthodes peuvent travailler sur un sous ensemble d'une population partagée pour quelques itérations. Si les individus pour lesquels un sous-ensemble est plus performant que les autres dans la population partagée, ils ont alloués pour immigrer vers eu.

Avec cette approche, une méthode qui converge lentement peut être combinée avec une qui converge rapidement, ou une méthode qui performe bien sur un sous-ensemble de l'espace de recherche peut être combinée avec une méthode qui performe bien un autre sous-ensemble.

La même idée est employée dans les algorithmes génétiques avec des populations multiples, pour trouver des solutions multiples dans un espace multimodal.

4.3.3 Les méthodes hybrides hiérarchiques

Les méthodes hybrides hiérarchiques, des méthodes d'optimisation différentes sont appliquées dans des niveaux différents de l'optimisation. Dans un niveau global, on peut appliquer une stratégie robuste d'optimisation pour trouver une disposition optimale. Dans les niveaux inférieurs,

où le système doit être moins complexe et sensible il est plus approprié d'employer une instance de méthode purement analytique.

4.3.4 Les opérateurs additionnels :

Dans la littérature il y a quelques exemples d'hybridation entre les différentes méthodes d'optimisation où des opérateurs d'une méthode sont ajoutés à ces méthodes ou même sont remplacés par les opérateurs standard des autres méthodes.

Par exemple, Yen et al. [61] ont introduit une méthode de simplexe comme une nouvelle manière de générer les enfants dans un algorithme génétique. Dans chaque itération un certain pourcentage de la population est généré par la méthode de simplexe sur un groupe d'individus prometteurs. Le reste de la population est généré en utilisant les opérateurs usuels génétiques. Ca permet d'améliorer la vitesse de convergence et la competence de trouver l'optimum absolue[62].

5 Analyse de performance en optimisation multiobjectif

L'existence de plusieurs solutions optimales et l'absence d'ordre total entre les solutions rendent l'analyse de performance une opération difficile et délicate. Dans ce cas, un ensemble de mesures ont été proposées pour évaluer les performances des algorithmes multiobjectifs.

5.1 Schott's spacing metric

Cette métrique, basée sur un calcul de distance entre les solutions, a pour objectif de mesurer la distribution des solutions le long du front utilisé avec d'autres mesures, elle donne une indication intéressante.

5.2 Entropie

L'entropie utilise la notion de niche pour évaluer la distribution des solutions sur le front. Plus proche de 1 est la valeur obtenue, meilleure est la distribution.

5.3 Mesures utilisant une référence

Ce type de mesures utilisé une référence, qui peut être un point ou l'ensemble Pareto optimal (lorsqu'on a la chance de le connaître), pour évaluer la qualité d'un front. Pour plus de détails, nous proposons de consulter [30, 40, 63].

6 Conclusion

Dans ce chapitre, nous avons essayé de rassembler un état de l'art sur les problèmes d'optimisation multiobjectifs, tout en montrant la manière de définir un problème pareil, ses contraintes, ses objectifs ainsi que les approches utilisées pour le résoudre. Autres concepts connexes tels que la dominance et le front de Pareto sont également introduits. Les méthodes exactes peuvent être appliquées pour des problèmes de petite taille en donnant toujours des solutions optimales. Cependant pour les problèmes de grande taille, les méthodes approchées sont plutôt utilisées.

Dans le cadre de notre travail on s'intéresse au choix de la topologie du réseau et au placement de composants communiquant sur cette topologie (appelée la phase de mapping) nécessaire pour la conception des NoCs. De ce fait, une étude approfondie est présentée dans le chapitre suivant.

Chapitre 3

Phase de mapping et choix de topologie : état de l'art

1 Introduction

Le mapping consiste au placement des composants communiquant sur la topologie du réseau de sorte à maximiser ces performances. De ce fait, les composants qui communiquent le plus entre eux doivent être placés le plus proches possible. Par conséquent, le choix de la topologie du réseau influence étroitement ses performances.

Une étude sur les mécanismes et méthodes adoptées dans la littérature pour le choix de la topologie et la phase de mapping sera menée dans ce chapitre. L'étude sera accomplie par une synthèse et suggestions pour notre contribution.

2 Le choix de la topologie

Le choix de la topologie est considéré comme étant une étape très importante pendant la conception des réseaux. Elle spécifie la façon dont les nœuds du réseau et ses liens sont interconnectés. Cette dernière peut être représentée par un graphe G (N, C), où N est l'ensemble des routeurs qui composent le réseau et C est l'ensemble des canaux de communication[20].

2.1 Les paramètres d'une topologie

La topologie définit la manière dont les nœuds du réseau sont connectés. Elle a quelques paramètres qui ont un impact important sur les performances et la fiabilité du réseau. Ces paramètres sont :

2.1.1 Degré de nœud

Le nombre de ports connectés à un nœud est appelé Degré de nœud, il montre la complexité d'entrée-sortie d'un nœud. Un réseau est dit régulier si tous les nœuds ont le même degré ; sinon, c'est un réseau irrégulier. Le degré de nœud peut être constant ou varie en fonction de la taille du réseau. Un degré de nœud élevé réduit la longueur moyenne du chemin mais augmente la

complexité du réseau et un degré de nœud plus petit nécessite moins de coûts matériels sur les liaisons. Dans la plupart des cas, il existe une contrainte sur le degré de nœud, qui est le nombre de voisins directs d'un nœud[64].

2.1.2 Diamètre

Le diamètre d'un réseau est la distance entre les nœuds, le nombre maximum de liens qui doivent être traversés pour envoyer un message à tous les nœuds à travers le chemin le plus court. S'il n'y a pas de connexion directe entre deux nœuds, le message doit traverser des nœuds intermédiaires qui introduiront un délai de saut, cela provoquera un retard de message et il est proportionnel au nombre de sauts, la longueur du diamètre devient un facteur important. Un petit diamètre aide à fournir une latence faible et prévisible, à prévoir les chemins de routage et le flux de trafic[65]. Plus le diamètre d'un réseau est petit, moins il faut de temps pour envoyer un message d'un nœud au nœud le plus éloigné.

2.1.3 Complexité du lien

Il existe des liens pour connecter les nœuds les uns aux autres dans le réseau. À mesure que le réseau évolue, la complexité de la liaison augmente. L'ajout de liens supplémentaires à un réseau peut réduire le diamètre du réseau et améliorer la communication entre les nœuds. Mais les liaisons sont coûteuses, une complexité de liaison plus élevée peut induire une complexité et une surface matérielles plus élevées[66]. Dans toutes les topologies, le réseau entièrement connecté présente la plus grande complexité de liaison.

2.1.4 Largeur de bissection

La largeur de bissection et la bande passante de bissection des réseaux d'interconnexion sont deux paramètres importants d'un réseau. La bissection reflète le nombre minimal de liens qu'il faut neutraliser pour diviser la topologie en deux réseaux de taille égale approximative. Une grande largeur de bissection fournit plus de chemins entre deux sous-réseaux, et améliore ainsi les performances globales du réseau. La largeur de la bissection a été un excellent paramètre typique pour évaluer et comparer les réseaux d'interconnexion pour les architectures parallèles. Seuls l'anneau, l'étoile et l'arbre ont une largeur de bissection fixe, en fonction de la taille du réseau [67].

2.2 Classification des topologies standards

D'après plusieurs recherches et revue une topologie peut être classée selon :

2.2.1 Régularité de la topologie

2.2.1.1 Topologie régulière

On dit qu'un réseau ou une topologie est régulière lorsque tous les nœuds ont le même degré (tous les nœuds ont le même nombre de voisins). La topologie est régulière quand tous les noeuds du graphe de réseau ont exactement le même nombre d'arc les reliant aux noeuds adjacents. Elle présente l'intérêt d'être une topologie de structure mathématique simple, ce qui permet l'utilisation des règles de routage standards.

L'avantage de ce type de topologie est qu'elle suppose une distribution homogène des routeurs donc il permet de réduire le temps et le coût de conception. Ce qui conduit la possibilité d'incorporer une telle topologie à des conceptions de SoC à usage général.

De plus les topologies régulières sont hautement réutilisables et imposent l'effort minimum de la reconception, au cas où elles seraient utilisées pour autres applications.

Cependant, en pratique, le placement et la taille des composants IPs du SoC permettent rarement d'intégrer une topologie régulière. De plus, l'utilisation non optimale des interconnexions entraînent à son tour un retard et une consommation d'énergie accrus[**?**, 20, 68].

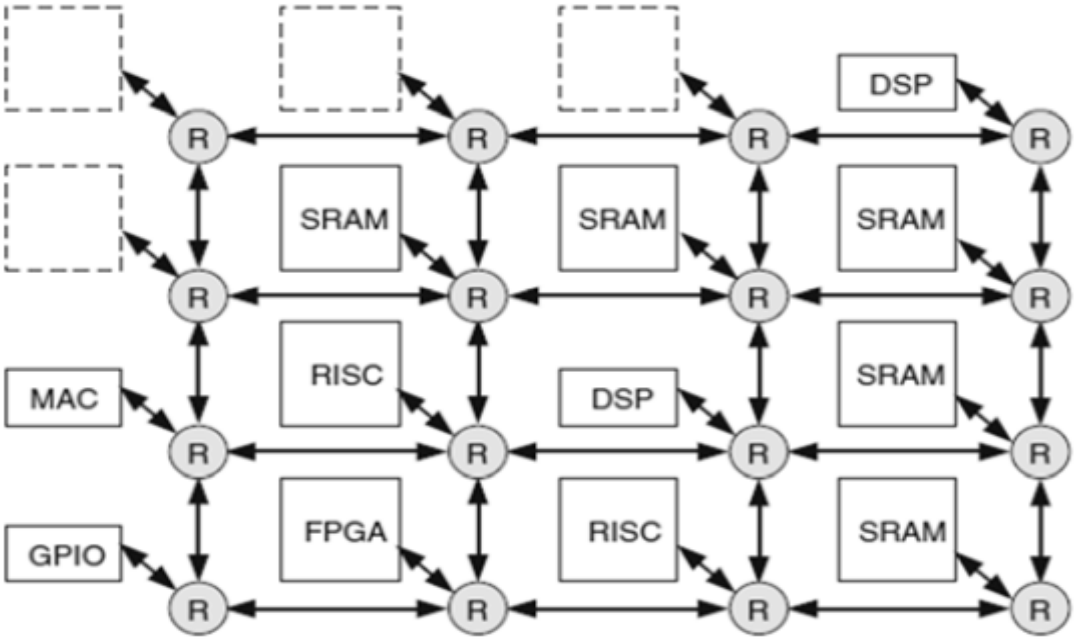


FIG. 3.1 : Un exemple d'une topologie régulière[4]

2.2.1.2 Topologie irrégulière

Une topologie irrégulière permet de tailler précisément le réseau requis. En effet elle peut être issue d'une topologie régulière qui a été retaillée au plus juste de façon à enlever les éléments non utilisés.

Une topologie irrégulière nécessite en revanche une plus grande attention pour le routage car les règles à appliquer ne sont plus triviales ce qui rend son implémentation et son contrôle plus complexe.

Ce type de topologie est plus approprié pour des réseaux spécifiques à une application. Un autre moyen d'obtenir des NoCs irréguliers optimisés est de supprimer les routeurs et les liens non nécessaires d'un réseau régulier[**?**, 20].

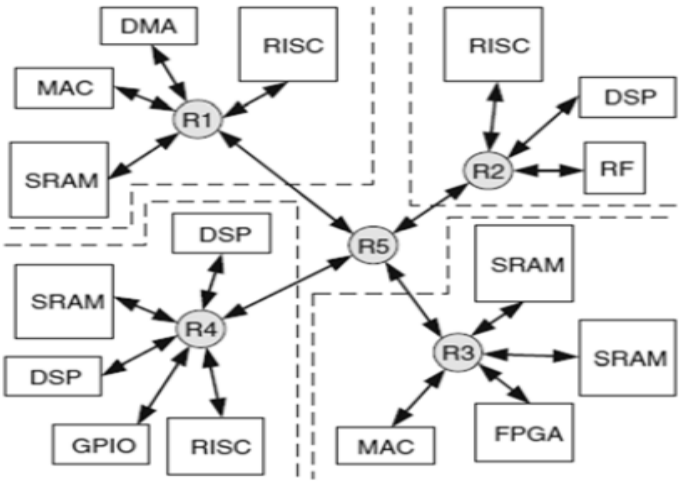


FIG. 3.2 : Un exemple d'une topologie irrégulière[4]

2.2.2 Directe et Indirecte [?, 17, 20]

2.2.2.1 Topologie Directe

Pour les topologies directes (Mesh, Tore, Anneau), chaque élément de calcul (IP) est associé à un routeur ; donc tous les routeurs sont des sources et des destinations. Les paquets sont transmis directement entre les nœuds terminaux.

Les topologies directes conduisent généralement à une plus grande disponibilité de la bande passante de communication, mais elles imposent une augmentation des nœuds dans le système. Par conséquent, il existe un compromis fondamental entre la connectivité et le coût. Un certain nombre de produits NoC existants incorporent une topologie directe.

2.2.2.2 Topologie Indirecte

Dans une topologie indirecte (Papillon, Arbre élargi) est une alternative pour la conception d'interconnexions extensibles où les routeurs peuvent être connectés seulement à d'autres routeurs. Les éléments de calcul sont les sources et les destinations du trafic, les nœuds intermédiaires changent simplement le trafic vers et à partir des éléments de calcul.

2.2.2.3 Topologie hybride

Pour les réseaux hybrides, la topologie est basée sur une combinaison entre les deux types de topologies (directe et indirecte).

2.2.3 Standard et Personnalisée

2.2.3.1 Topologie standard

Les topologies standards(mesh 2D, torus, Hypercube, Butterfly,..) se caractérisent par sa régularité et peut-être une topologie directe ou indirecte.

La topologie 2D maillée est la plus facile à implémenter car elle introduit une complexité de conception minimale. Plus spécifiquement, la topologie maillée présente certaines propriétés souhaitables, telles qu'un schéma d'adressage très simple et de multiples routes source-destination, qui confèrent une robustesse aux perturbations du réseau. Cependant, les topologies maillées 2D sont plus susceptibles de présenter des problèmes d'encombrement au centre de l'architecture, plutôt qu'à la périphérie, en raison des algorithmes de routage[**?**, 69].

2.2.3.2 Les topologies personnalisées

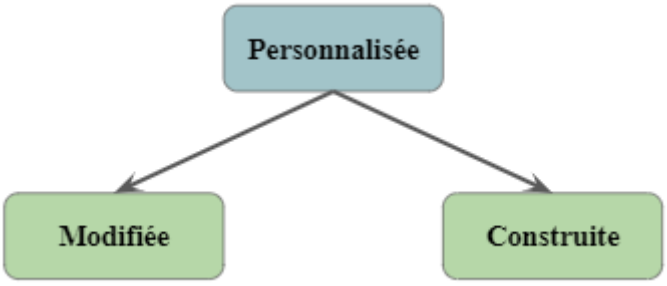


FIG. 3.3 : Les types de topologie personnalisée[70]

Une topologie peut etre personnalisée à partir d'une ou plusieurs topologies régulières, dont des leins sont ajoutés ou supprimés (figure 3.4 (A)).

Dans le cas la topologie construite figure 3.4(B), après l'affectation des IPs aux routeurs, ces derniers sont reliés par les liens nécessaires à leur communication.

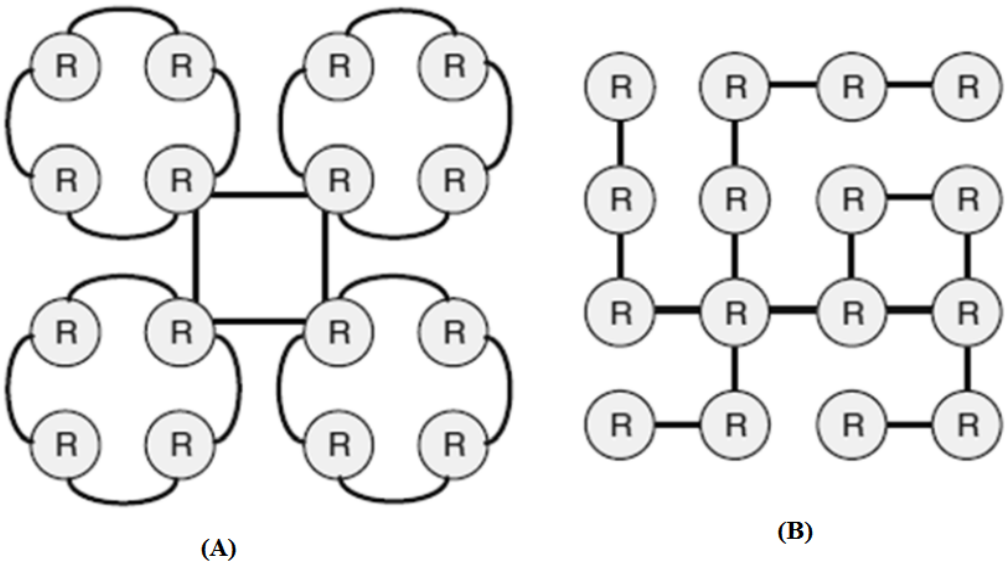


FIG. 3.4 : Exemples de topologies personnalisées[4]

2.3 La technologie 3D

La technologie des circuits intégrés à 3 dimensions (3D IC) est définie par des multiples couches en silicium qui permet d'empiler et de lier des puces semi-conductrices pour former un empilement vertical connecté électriquement de puces de circuits intégrés, formant une seule puce 3D[71, 72]. En effet les caractéristiques de l'intégration 3D sont la diminution de la longueur des fils et donc réduction du délai d'interconnexion, augmentation du nombre d'interconnexions entre les modules et capacité de permettre divers matériaux, technologies de processus et fonctions sur une seule puce[73, 74].

Les connexions électriques entre les matrices empilées sont réalisées à l'aide des liens d'interconnections verticaux qu'on nomme TSV (Through Silicon Via).

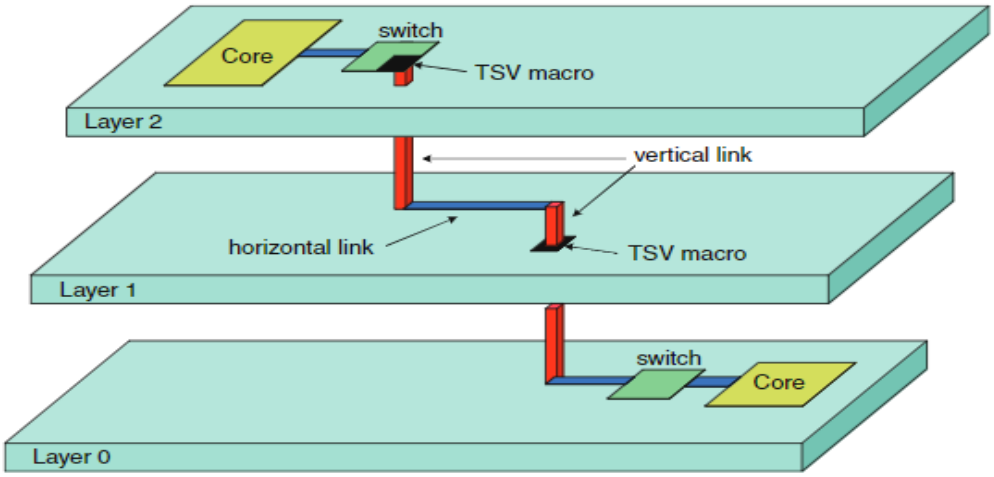


FIG. 3.5 : Un circuit intégré 3D basé sur le TSV[71]

2.3.1 Les avantages de 3D par rapport à 2D

L'avantage principal des réseaux 3D est la réduction du nombre moyen de sauts due à l'augmentation du degré de routeur, ce qui permet l'amélioration de ces performances. En effet, le routeur utilisé en 2D NoC à 4 port, cependant, le routeur dans le 3D a minimum un port supplémentaire et ce qui conduit à l'augmentation du nombre de liens d'interconnections qui connecte les nœuds et les routeurs.

Dans le réseau 2D mesh le nombre total des nœuds N est $N = n_1 + n_2$ où $n_i$est le nombre des nœuds dans $i^{ème}$ dimension et pour le réseau 3D mesh $N = n_1 + n_2 + n_3$.

Les chercheurs Pavlidis et Friedman [75] ont proposé l'équation (1) pour calculer le nombre moyens de hops dans un réseau de 3D.

$$Hops_{NOC} = \frac{n_1 n_2 n_3 (n_1+n_2+n_3) - n_3(n_1+n_2) - n_1 n_2}{3(n_1 n_2 n_3 - 1)} \quad (1)$$

Prenons un exemple où on réseau 2D mesh 16x16 avec N_1 =64 nœuds et réseau 3D mesh 8x8x4 avec N_2 =64 nœuds donc le nombre moyen de hops pour ces deux réseaux sont respectivement 10.67 et 6.52 alors on a 40% plus de hops pour un réseau à 2 dimensions par rapport à un réseau à 3 dimensions. Feero et Pande [76] ont proposé l'équation (2) pour calculer le nombre des liens d'interconnections.

$$Links = (n_3 - 1) + n_1 n_3 (n_2 - 1) + n_2 n_3 (n_1 - 1) \quad (2)$$

Pour l'exemple précédent, le nombre des liens dans le réseau 2D est 480 links, cependant le nombre dans le réseau 3D est 640 links.

La même recherche [75] a montrée qu'une réduction de 62% et de 58% en consommation de puissance est obtenue avec un réseau en 3D par rapport à son équivalent en 2 dimensions (ayant le même nombre des cœurs connectés) en considérant des réseaux de taille $N = 128$ et $N = 256$ (où N est le nombre de cœurs connectés).

Selon [77] Les lignes d'interconnexion globale peuvent être remplacées par through-silicon-vias si le circuit est correctement plié. En raison de la réduction de la longueur des fils comme le montre

la figure 3.6, les systèmes 3D sont censés pouvoir fonctionner à une fréquence d'horloge plus élevée ou consommé moins d'énergie à la même fréquence d'horloge.

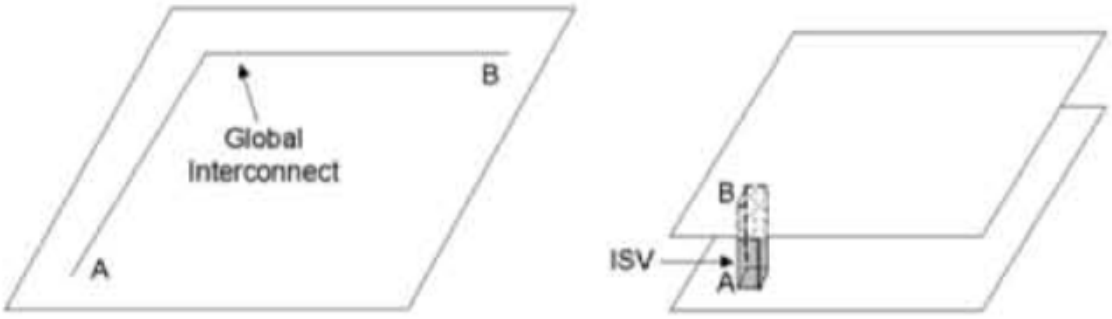


FIG. 3.6 : Exemple de réduction de longueur de fil dans 3DIC[77]

Feero et Pande [78] démontrent que les NoC 3D sont capables d'atteindre un débit plus élevé, une latence plus faible et une dissipation d'énergie plus faible au coût de petite surface de silicium.

Une étude comparative entre les circuits 2D et 3D pour la consommation d'énergie, les performances, le coût de production de masse et la facilité de fabrication est réalisée et présentée dans le tableau 3.1.

TAB. 3.1 : 2D vs 3D[79]

Paramètre	2D	3D
Fonctionnalité	+	++
Puissance	++	+++
Performance	+	+++
Coût de production de masse	++	+
Facilité de fabrication	++	+++

+++ : Meilleur ; ++ : Bonne , + :Pire

Notons que l'utilisation des circuits 3D offre des meilleurs résultats dans la plupart des paramètres par rapport aux circuits 2D. Cependant, la technologie 3D est la plus coûteuse.

2.3.2 Through-Silicon-Vias (TSVs) :

Les interconnexions verticales mises en œuvre sous la forme de vias en silicium (TSV) sont considérées comme la technologie d'interconnexion verticale la plus prometteuse pour les CI 3D. Elles offrent la plus grande bande passante d'interconnexion dans les circuits intégrés 3D et une plus haute densité d'intégration par rapport au câblage[80].

Cependant, la surface occupée par les TSVs est très importante. La figure 3.7 montre la différence entre la surface d'un TSV et celle d'une porte logique ou une cellule mémoire. De ce fait, le nombre total de TSVs utilisés dans la configuration affecte de manière significative le coût global des circuits intégrés 3D[81, 82].

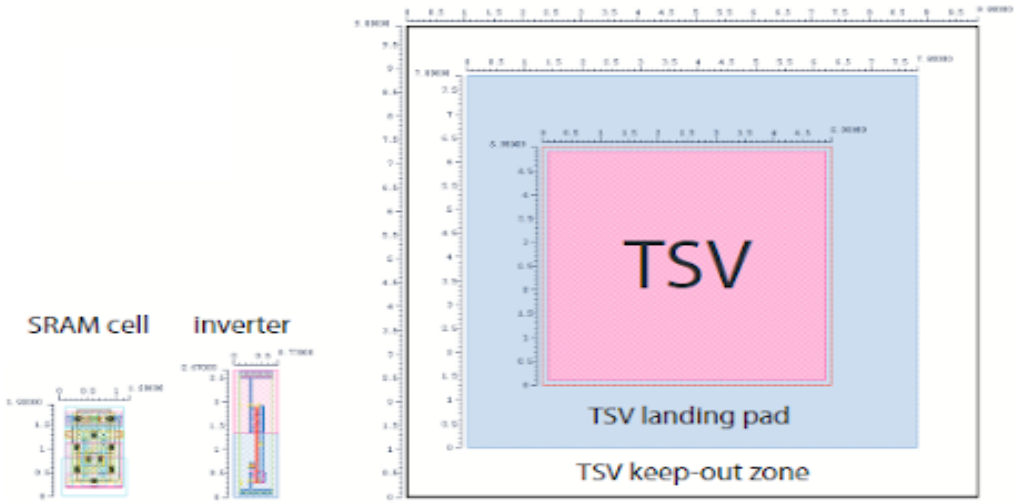


FIG. 3.7 : La différence de surface entre TSVs et des portes logiques et les cellules mémoires[82]

L'utilisation d'un nombre important de TSVs diminue considérablement la longueur des fils des circuits 3D mais augmente la taille des puces et annulera les avantages apportés par les TSV[83].

2.4 Les topologies personnalisées : les solutions proposées dans la littérature

Cette approche vise à générer une topologie sur mesure du type irrégulièr pour une application spécifique dont les caractéristiques sont connues à priori.

Nous exposons dans les sections suivantes, quelques solutions proposées dans la littérature pour les topologies 2D et 3D.

2.4.1 Topologies 2D personnalisées

Dans [84], la topologie du réseau est personnalisée en insérant de longs liens sur la topologie standard 2D Mesh. Pour [85] la personnalisation a été réalisé en utilisant une méthode de clustering, les noeuds qui communiquent fréquemment entre eux sont mis dans le même cluster et les clusters sont reliés par un lien. Dans [86], une approche pour adapter les liaisons transversales afin de réduire la distance moyenne du réseau entre les noeuds mappés (augmentation de la performance). D'autres approches structurelles pour la personnalisation de la topologie NoC pour une application spécifique ont été présentées dans[85, 87].

Des approches pour des réseaux sur puce reconfigurable qui permettent l'adaptation en cours d'exécution de la topologie aux exigences des applications sont proposées dans [88, 89, 90]. Ces approches sont compilées sur des composants FPGA (Field-Programmable Gate Array) et sont, cependant, limitée à des composants reconfigurables. Une architecture NoC, appelée ReNoC (ReNoC : Reconfigurable NoC), est proposée dans [91], pour permettre à la topologie du réseau d'être configurée par l'application exécutée sur l'architecture SoC. ReNoC combine la commutation de paquets et la commutation de circuits dans la même topologie. Cela permet de créer des topologies d'applications spécifiques dans une plateforme générale de SoC à base de NoC.

D'autres approches de personnalisation de la topologie par insertion de liens ont été proposées. Ces approches peuvent être classées en deux catégories principales : des approches de synthétisation et des approches de personnalisation [92]. La première classe concerne les techniques proposées pour personnaliser entièrement l'architecture, elles sont plus appropriées pour une application spécifique, et elles ne sont pas nécessairement adaptées aux topologies régulières (ex., 2D Mesh) [93, 94, 95, 96]. Par exemple, dans[93], un algorithme génétique est présenté pour la synthèse d'une application spécifique pour SoC avec l'objectif d'optimiser la consommation d'énergie et la surface. Cet algorithme est multiobjectif, il permet de générer un nombre solutions non dominées.

L'autre classe concerne les approches de personnalisation partielle de la topologie. En revanche, elles commencent principalement par une architecture d'interconnexion sur puce standard comme le 2D Mesh. Puis, elles personnalisent partiellement l'architecture de base en insérant des longs liens. Par exemple, une approche de personnalisation d'un 2D Mesh standard en insérant quelques longs liens spécifiques à l'application a été proposée dans [97]. L'algorithme commence par une architecture régulière de 2D Mesh et ajoute progressivement des liens à longue portée entre les commutateurs selon la fréquence de communication la plus élevée jusqu'à ce que toutes les ressources (liens) disponibles sont utilisées. Les résultats analytiques et de simulation ont été présentés pour montrer que cette approche permet de réduire considérablement la latence moyenne des paquets. Toutefois, le choix stratégique des liens à insérer sur le réseau d'interconnexion sur puce qui convient le mieux aux besoins de plusieurs applications est difficile sans connaître à l'avance leurs correspondances avec les modèles de trafic.

2.4.2 Topologies 3D personnalisées

3D partiellement connecté ou bien Vertically-Partially-Connected 3D-NoC est une solution proposée pour réduire le nombre de TSV utilisés, ce type de réseau se caractérise par l'élimination de certains liens verticaux. De nombreuses topologies 3D NoC sont proposées dans[98, 99, 100]. Cependant, selon [101] trouver une topologie appropriée pour 3D NoC reste un problème ouvert.

Parmi les topologies, les topologies 3D Mesh standard entièrement et partiellement connectées sont les plus présents dans de nombreuses littératures actuelles. La 3D RNT (RNT :Recursive Network Topology) a été identifié comme une topologie appropriée pour les NoC 3D car il offre un haut degré d'évolutivité, et elle est bien adapté à une approche modulaire et à la mise en œuvre de blocs IP hétérogènes[102, 103]. La RNT 3D est dérivé de la topologie de réseau récursif WK où quatre liens verticaux supplémentaires sont utilisés dans chaque couche sauf dans les couches supérieure et inférieure de la topologie, seuls 25% des blocs de chaque couche sont interconnectés avec les couches voisines à l'aide des liaisons verticales.

La MMT 3D(3D Modified Mesh Topology) est une topologie 3D Mesh partiellement connectée ($4 \times 4 \times 3$) où 50% des blocs de chaque couche sont interconnectés avec leurs blocs homologues des couches voisines. Dans les deux topologies données, chaque nœud comprend un commutateur et un bloc fonctionnel ou de stockage [101]. Et chaque quatre nœuds sont regroupés pour créer quatre clusters dans chaque couche, donc le nombre total de nœuds dans une couche est de seize.

Dans [19] ils ont proposé un outil de conception, SunFloor 3D, pour synthétiser des NoC 3D spécifiques à l'application. Afin d'avoir une meilleure topologie NoC pour l'application et trouver

des chemins pour les flux de communication. Leur étude montre également que les topologies synthétisées entraînent une grande puissance (54% en moyenne) et (21% en moyenne) par rapport aux topologies standards 3D.

TAB. 3.2 : Les topologies personnalisées

Référence	Méthode	Topologie	Objectif
[67]	Insertion de liens	2D Mesh personnalisée	Latence
[84]	Méthode de décomposition	2D Mesh personnalisée	Energie
[85]	Insertion de liaison transversal	personnalisée	Latence, Bande passante
[94]	Algorithme Génétique	personnalisée	Energie
[97]	Insertion de liens	2D Mesh personnalisée	Bande passante, Latence
[99]	Suppression de liens	3D Mesh personnalisée	Surface, Energie
[104]	Algorithme basé sur la programmation linéaire	Personnalisée	Energie, Nombre de routeur
[105]	Algorithme Génétique	Personnalisée	Energie + Débit
[106]	Recuit Simulé	Personnalisée	Latence,Énergie, Nombre de routeur
[101]	Heuristique basé sur les clusters	3D personnalisée	Nombre de TSV, Latence
[107]	Branch-and-bound	3D personnalisée	Energie

Après le choix de la topologie du réseau sur puce, nous abordons dans la section suivante la phase de mapping. Le mapping consiste au placement des composants communiquant sur la topologie du réseau de sorte à maximiser ces performances.

3 Phase de mapping

3.1 Définir la phase mapping

La phase de mapping est une phase centrale lors de la mise en œuvre d'un réseau sur puce dont le résultat a un impact direct sur les performances du système. Elle consiste à placer chaque élément de l'application sur l'architecture du réseau sur puce de sorte à minimiser le coût de communications, la consommation d'énergie, la latence du système et respecter les contraintes telles que la bande passante ou le temps réel.

Ce processus est très complexe et difficile surtout lorsque la taille de l'application est importante[17, 108].

3.2 Type de mapping

On distingue deux grandes classes de techniques de mapping : mapping statique et mapping dynamique.

3.2.1 Mapping statique (Design-Time Mapping)

Dans cette catégorie de mapping, le placement se fait lors de la conception de l'architecture. Les méthodes de mapping statiques bénéficient donc d'une vue globale du système ce qui facilite la prise de décision[108]. Il est adapté à des plateformes spécifiques. Toutes les IPs sont affectées aux tuiles de l'architecture avant que l'application ne soit exécutée[17].

3.2.2 Mapping dynamique (Run-Time Mapping)

Contrairement au mapping statique, le mapping dynamique se fait en temps réel (durant l'exécution de l'application.). En effet on peut ajouter, modifier et même supprimer une tâche de l'application[17, 108].

Le but de ce type est d'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[109].

Le tableau suivant montre une comparaison entre la mapping statique et dynamique.

TAB. 3.3 : Comparaison entre mapping statique et mapping dynamique[70]

	Mapping statique	Mapping dynamique
Mapping des IP	+	-
Mapping des tâches	+	+
Migration de tâches	-	+
Le mapping est exécuté au moment de la conception	+	-
Le mapping est exécuté au moment de l'exécution	-	+
Architecture homogène (IP)	+	+
Architecture hétérogène (IP)	+	+
Topologie standard	+	+
Topologie personnalisée au moment de la conception	+	-
Mono-application ou multi-applications	+	+

3.3 La problématique de la phase mapping

Le problème du mapping est un problème d'optimisation combinatoire NP difficile. L'espace de recherche croit avec la taille du système. Théoriquement, placer N IPs dans M noeuds du réseau ($N \leq M$) implique $\frac{M!}{(M-N)!}$ arrangements d'IPs possibles dans les noeuds du NoC. La solution optimale d'un problème d'optimisation peut rarement être déterminée en un temps polynomial. Il est nécessaire de développer une heuristique ou métaheuristique afin de déterminer une solution optimale ou pré-optimale dans un temps CPU raisonnable[70].

3.4 La mapping dans la littérature

En raison de leurs potentiels élevés, les problèmes de mapping ont attiré l'attention de nombreux chercheurs. Une enquête détaillée à ce sujet peut être trouvée dans [110]. Les stratégies vont des méthodes exactes à l'aide de la programmation linéaire entière (ILP), aux méthodes approchées heuristiques et métaheuristiques.

Parmi les techniques métaheuristiques notables on trouve CGMAP [111], il utilise l'opérateur de mapping chaotique au lieu des processus aléatoires dans l'algorithme génétique (AG). Il ya aussi GAMR [112], qui est une approche de mappage et de routage basée sur l'AG, traite un mappage biphasé des cores IP sur l'architecture NoC. Une approche évolutive appelée GBMAP [113] réduit la consommation d'énergie et les besoins en bande passante totale du NoC. Dans[114], un algorithme d'optimisation basée sur les colonies de fourmis (ACO) a été proposé pour le

mapping des tâches d'application sur un NoC. Les résultats ont été comparés à des techniques de mapping aléatoire.

PSMAP, qui est une stratégie métaheuristique utilisant la technique PSO, a été proposée dans [115] pour réduire les coûts statiques et dynamiques des NoCs pour le mapping d'application basé sur mesh 2D.Tous ces travaux adressent le mapping d'application sur les NoCs 2-D. Dans [116], les auteurs ont proposé une technique de gestion de noeud pour une applications du traitement numérique du signal (DSP), qui est une extension de NMAP [117], afin de prendre en charge le NoC 3D qui combine le dimensionnement du tampon et l'attribution de priorité pour les applications DSP sur l'architecture NoC 3D. La technique de mapping 3D [118] montre des économies d'énergie en appliquant différentes valeurs de tensions d'alimentation (Vdd)aux différentes couches de l'architecture NoC 3D.

TAB. 3.4 : Carctéristiques des travaux sur la phase de mapping

Réference	Classe de mapping	Méthode utilisé	Objectif	Topologie visée	Les performances considérés
[110]	Statique	PSO	Multiobjectif	2D/3D Mesh	Consommation d'énergie +Latence+Surface
[111]	Statique	AG	Multiobjectif	2D Mesh	latence + consommation d'énergie
[112]	Statique	AG	mono-objectif	2D Mesh	Consommation d'énergie
[113]	Dynamique	EP	Multiobjectif	2D Mesh	Consommation d'énergie + Bande passante
[114]	Dynamique	ACO	Multiobjectif	2D Mesh	Temps d'exécution+ Consommation d'énergie
[115]	Statique+ Dynamique	PSO	Multiobjectif	2D Mesh	Consommation d'énergie +Latence
[116]	Dynamique	Multiple vdd	mono-objectif	3D Mesh	Consommation d'énergie
[119]	Dynamique	le recuit simulé	mono-objectif	Standard homogène	Consommation d'énergie
[120]	Statique	MOGA	Multiobjectif	Régulière	Energie+Bande passante
[121]	Dynamique	ACP	Multiobjectif	Régulière	Energie + La latence
[122]	Dynamique	SNN	Multiobjectif	Standard	Temps d'exécution + Consommation d'énergie
[123]	Dynamique	CMAP	mono-objectif	Standard	Energie
[124]	Statique	NSGA-II et MicroGA modifié	Multiobjectif	Personnalisé	Surface + Temps d'exécution + Consommation électrique
[125]	Statique	AG	mono-objectif	2D/3D	Latence
[126]	Statique	ILP	mono-objectif	2D mesh	Coût de communication
[127]	Statique	B&B	mono-objectif	2D/3D	consommation d'énergie
[128]	Statique	DOL	Multobjectif	2D/3D	Temps de calcul + Temps de communication
[129]	Dynamique	B&B	Multiobjectif	2D Mesh 2D Torus	Latence +Débit+ Consommation d'énergie

[130]	Statique+ Dynamique	CasNet	mono-objectif	Mesh	consommation d'énergie
[131]	Dynamique	Path Load	Multiobjectif	2D Mesh	latence et bande passante
[132]	Statique+ Dynamique	DI_GA	mono-objectif	2-D 4×4 Mesh	consommation d'énergie
[133]	Dynamique	CHMAP	mono-objectif	Mesh	L'énergie
[134]	Dynamique	GHA	Multiobjectif	2D Mesh	consommation d'énergie et latence

4 Synthèse :

D'après les travaux de recherche que nous avons cités dans ce chapitre, nous notons que :

Concernant le choix de topologie :

- La topologie peut suivre la classification suivante régulière ou irrégulière, directe ou indirecte et finalement standard ou bien personnalisé.
- La topologie 2D Mesh est la plus couramment utilisée dans la conception des NoC. Cela est dû à sa facilité de mise en œuvre et à son indépendance par rapport à la taille du réseau.
- Le principal avantage d'une topologie NoC standard est sa réutilisation et la réduction du temps de conception. Ils conviennent aux MPSoCs qui incluent des blocs homogènes.
- Pour les ASICs qui sont composées de blocs hétérogènes, les topologies NoC personnalisées leurs conviennent le mieux en termes de consommation d'énergie et de surface.
- Pour les NoC 3D la topologie la plus utilisée est la MMT 3D.

Le tableau 3.5 résume les avantages et inconvénients des différents types de topologies.

TAB. 3.5 : Comparaison entre les topologies standards et les topologies 3D personnalisées

	Avantage	Inconvénient
2D Standard	-La facilité de mise en œuvre -La réutilisation -La réduction du temps de conception.	-Performance limité -Non scalable
3D Standard	-Scalabilité - La réutilisation	-La surface -Mise en oeuvre
3D Personnalisée	-Amélioration des performance - Réduction de surface (en limitant le nombre de TSVs) - Scalabilité	-Non réutilsable -Nécessite un long temps de conception

Concernant la phase de mapping, les techniques discutées préalablement, les méthodes exactes construites autour d'ILP ne sont pas pratiques pour les NoC ayant un grand nombre de cœurs, en raison de leur temps d'exécution élevé.

Les stratégies heuristiques basées sur la politique d'amélioration itérative dépendent fortement de la solution initiale. Les heuristiques constructives, en revanche, fonctionnent avec une notion prédéterminée de la façon pour parvenir à une bonne solution. Par exemple, presque toutes ces méthodologies fonctionnent avec un ordre des blocs / arêtes du APG, triés par ordre décroissant des besoins de communication.

L'approche évolutive basée sur GA, ACO et PSO permet une meilleure exploration de l'espace de recherche, car de multiples solutions évoluent simultanément avec des interactions mutuelles entre elles. Cependant, ces stratégies exploratoires doivent également être guidées. Une observation générale à propos de la PSO est qu'elle a une convergence plus rapide que des techniques similaires telles que les AG et peut fonctionner avec une population relativement petite[110].

Bien que PSMAP [109] utilise la formulation basée sur le PSO, il ne contrôle pas la population initiale (qui est entièrement aléatoire dans [109]) ou l'exploration utilisant plusieurs essaims.

Nous proposons dans le cadre de notre travail de traiter en parallèle la phase de mapping et la personnalisation de la topologie en considérant la technologie 3D. Notre idée considère plusieurs objectifs à optimiser en même temps : le coût de communication (délai/énergie) pendant le placement des composants (mapping). Ainsi que, la surface du réseau en optimisant le nombre de liens verticaux. Et ceci permettra au final de spécifier une topologie 3D personnalisées.

5 Conclusion :

Dans ce chapitre, nous avons fait un état d'art sur le choix de topologie et la phase de mapping dans les réseaux sur puce. Dans la première partie, on a présenté les différentes topologies possibles pour un NoC ainsi que les méthodes de personnalisation. Nous avons aussi introduit le principe du réseau à 3D, nous avons cité les avantages qu'il présente par rapport à la structure classique basée sur les circuits à 2D et présenté ses limitations surtout en ce qui concerne l'emploi des TSVs.

Pour la deuxième partie de ce chapitre, on a abordé le processus de mapping. Le type de mapping change selon le type de l'application et sa complexité, il peut être statique ou dynamique.

On conclut qu'il existe une relation étroite entre le mapping et le choix de topologie, de ce fait la synthèse présentée dans ce chapitre nous a permis de concrétiser notre idée de base et de proposer une nouvelle solution (algorithmique, application) qui sera détaillée dans le chapitre suivant.

Chapitre 4

Conception de la technique proposée

1 Introduction

L'apparition de la technologie des circuits intégrés à 3 dimensions (3D IC), qui est défini par des multiples couches en silicium empilé ensemble et communiquant à travers des liens d'interconnections verticaux qu'on nomme TSV, a encouragé les chercheurs à implémenter leurs réseaux en utilisant cette nouvelle technologie pour en exploiter les avantages par rapport aux circuits à 2 dimensions. Comme c'est une technologie récente, les circuits à 3 dimensions montrent certaines limitations concernant en particulier la mise en place des TSV.

Notre proposition consiste à traiter en parallèle la phase de mapping et la personnalisation de la topologie en utilisant la technologie 3D. Cette problématique est considérée comme un problème d'optimisation multiobjectif. Pour cela que dans ce chapitre, nous présentons une méthode d'optimisation multiobjectif, basée sur la PSO, qui vise à trouver des solutions de compromis (non dominées) qui optimisent toutes nos fonctions objectives (coût de communication et coût de surface).

2 Formulation du problème

Traiter en parallèle la phase de mapping avec le choix de la topologie en considérant la technologie 3D est classé comme un problème d'optimisation combinatoire multiobjectif vu le nombre de critères qu'on cherche à optimiser.

La modélisation de l'application en tant qu'IP exécutant les tâches et les ressources d'une architecture est réalisée sous forme de graphes dirigés. Le mapping implique de placer les composants du graphe d'application (IP) à chaque sommet du graphe d'architecture (Tuile) tout en minimisant les chemins de communication entre les IPs[70, 109].

Un graphe d'application (APG)

$G = G(C, A)$ est un graphe où chaque lien est affecté d'un poids qui représente le volume de communication (nombre de paquets envoyés) entre c_i à c_j.

Un graphe d'architecture (ARG)

$G = G(T, P)$ est un graphe où chaque vertex t_i représente une tuile de l'architecture, et chaque arc directionnel p_{ij} représente le chemin de routage entre t_i et t_j.

En utilisant les deux graphiques, le problème du mapping est d'assigner chaque IP de l'application à une tuile d'architecture afin de minimiser le coût des communications. Par conséquent, les IP qui ont un volume de communication élevé doivent être placés aussi près que possible les uns des autres.

La figure 4.1 montre un exemple sur la façon de mappé le graphe d'application sur un graphe d'architecture.

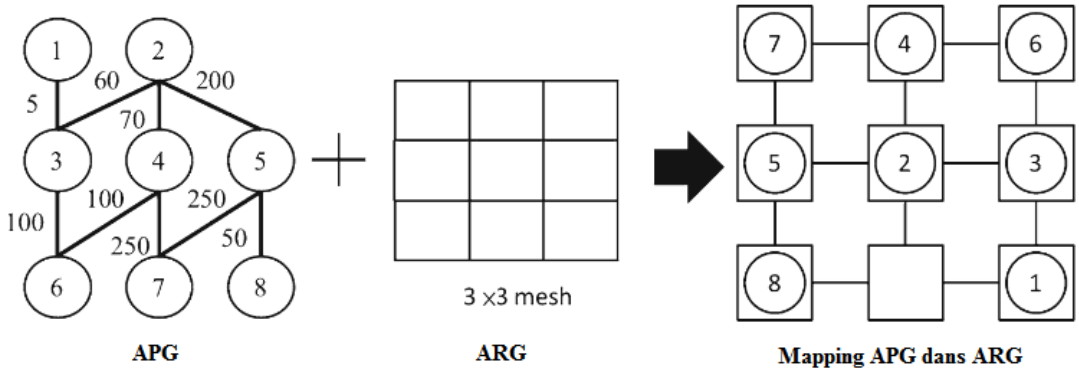


FIG. 4.1 : Exemple d'un mapping

Deux fonctions objectives seront traitées dans notre travail, qui sont : le coût de communication et la surface consommée.

$x \in X = x_1, x_2, ..., x_n$ tels que F(x) = le coût de communication à optimisé.

-X est ensemble des solutions de mapping générer par l'algorithme.

$y \in Y = [1, Max(TSV)]$ tels que S(x) = le coût en surface à optimisé.

-Y représente le nombre des liens verticaux dans la topologie du réseau. où :

$$Max(TSV) = \frac{Nombre\ Totale\ Des\ IPs}{2}$$

- **Le coût de communication F(x) :**

La fonction objective F(x) du coût de communication qui assure la résolution du problème du mapping (placement des IP) d'une manière optimale.

$$F(x) = min\,\{Coût : coût\ de\ communication\ entre\ les\ IPs\}$$

Où le coût des communications pour chaque solution est calculé comme suit :

$$Cost = \sum_{K=1}^{|E|} value(e_{(i,j)}) * hopcount(t_i, t_j)$$

Ici, |E| représente le nombre d'arêtes dans le graphe d'application.

$Value(d^k)$ représente le volume de transfert entre les IPs i et j qui sont mappés aux tuiles respectivement t_i et t_j

$hopcount(t_i, t_j)$ est le nombre de sauts entre les nœuds de topologie t_i et t_j .

Le $hopcount(t_i, t_j)$ qui peut être calculé sur deux façon :

1. Quand la communication est entre deux tuiles qui se trouvent dans le même plan, la formule mathématique est :

$$hopcount(t_i, t_j) = dManhattan(t_i, t_j) = |x_2 - x_1| + |y_2 - y_1|$$

2. Quand la communication se fait entre deux tuiles qui se trouvent dans différents plans la formule mathématique

$$hopcount = dManhattan(t_i, t_k) + dManhattan(t_k, t_j) + \alpha$$

Avec t_k est la position du lien vertical et α est coût du lien vertical.

Puisque un TSV offre des liaisons verticales courtes et rapides par rapport au fil 2D long, le coût de communication verticale est inférieur au coût de communication horizontale selon[135, 136] est :

$$Coût\ Communication = Coût\ Communication\ Horizontal + \alpha\ Coût\ Communication\ Vertical$$

Nous avons fixé dans la phase de test les valeurs de α à 1 et 0.8.

- **Le coût de surface S(x) :**

La fonction objective S(x) indique la surface utilisée par les TSV de l'architecture qui assure l'utilisation d'un minimum nombre de liens verticaux (nombre TSV).

$$S(x) = min\,\{Surf : surface\ de\ silicium\ utilisé = optimisé\ nombre\ des\ liens\ vertical\}$$

3 Supposition

Notant que les critères physiques de positionnement de tuiles, liens, l'espace à respecter entre les couches empilées, taille des routeurs, commutateurs, sont ignorés on s'intéresse seulement au coût de communication (énergie, latence) et au coût de surface (nombre de TSV).

Dans notre travail nous avons considéré un réseau 3D constitué de 2 niveaux seulement relié par des liens verticaux. Le nombre de plans est limités en raison des problèmes thermiques engendrés par l'empilement de ces plans[137].

Chaque niveau contient la moitié des IPs de l'application. Ces IPs sont mappés sur les tuiles suivant une topologie Mesh. Une tuile est un placement candidat d'un lien vertical (TSV). L'affectation de TSV se fait d'une manière aléatoire pour les tuiles (candidates) qui sont dans des positions identiques et qui se trouvent dans différents plans.

La figure 4.2 représente un exemple illustrant une application à 12 IPs mappé sur une topologie 3D en utilisant 3 TSV.

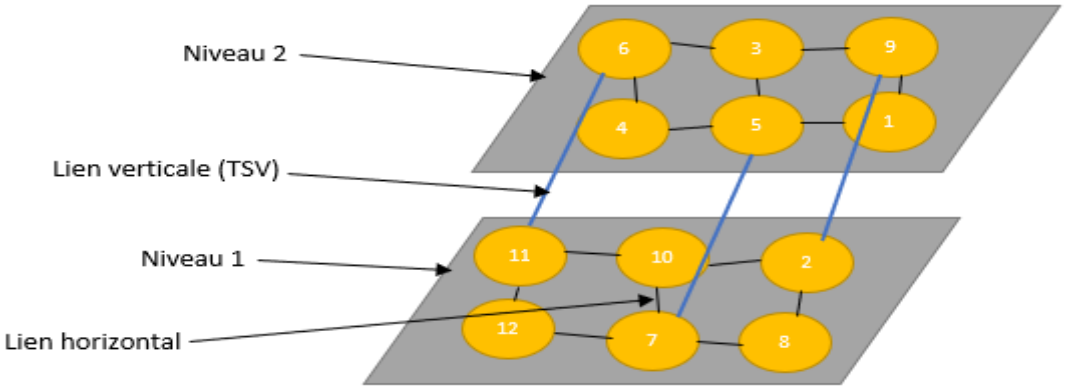


FIG. 4.2 : Représentation explicatif de notre supposition

4 L'Optimisation par Essaim Particulaire

PSO est un algorithme basé sur la population, c'est-à-dire il exploite une population d'individus pour explorer les régions prometteuses de l'espace de recherche. Dans ce contexte, la population est nommée les essaims et les individus (c'est-à-dire les points de recherche) sont nommés les particules. Chaque particule meut avec une vitesse adaptable avec l'espace de recherche, et garde une mémoire de la meilleure position dont elle rencontre. Dans les variants globaux de PSO, la meilleure position qui est retenue par les individus de l'essaim est communiquée aux autres particules. Dans le variant local, chaque particule est assignée à un voisinage topologique consisté d'un nombre pré-spécifié de particules. Dans ce cas, la meilleure position retenue par les particules qui comprend le voisinage est communiquée entre eux.

4.1 Les caractéristique d'une particule :

Chaque particule est considérée comme une solution du problème, où elle est dotée d'une position, la valeur de la fonction objectif pour cette position, une vitesse, une mémoire personnelle retenant la meilleure position visitée et une mémoire collective retenant la meilleure position visitée par le voisinage.

Les particules dans ce cas changent leur vitesse en se basant sur leur mouvement actuel, leur mémoire personnelle et leur mémoire collective. La position change en appliquant cette vitesse à la position courante. De ce fait, le comportement de la particule est un compromis entre les trois possibilités suivantes :

- Une composante d'inertie : La particule suit son chemin personnel.
- Une composante cognitive : La particule tend à retourner vers sa meilleure position.

- Une composante sociale : La particule tend à suivre la meilleure position trouvée par le voisinage.

La stratégie de déplacement d'une particule est illustrée dans la figure 4.3.

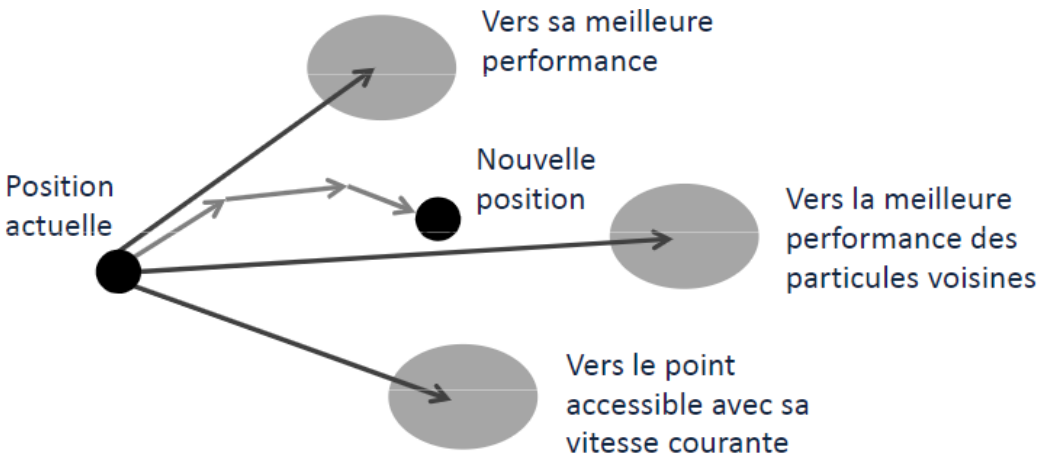
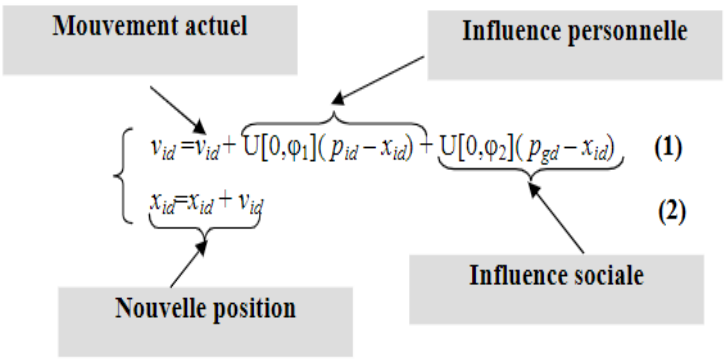


FIG. 4.3 : Déplacement d'une particule[138]

Une particule i de l'essaim dans un espace de dimension D est caractérisée, à l'instant t, par [139] :

- X_{id} : sa position dans l'espace de recherche ;
- V_{id} : sa vitesse ;
- Pb : la position de la meilleure solution par laquelle elle est passée ;
- Pg : la position de la meilleure solution connue de tout l'essaim ;
- f (Pb) : la valeur de fitness de sa meilleure solution ;
- f (Pg) : la valeur de fitness de la meilleure solution connue de tout l'essaim.



sachant que :

$U[0, \varphi_i]$ est une distribution entre une borne inférieure, qui est dans ce cas [0, et une borne supérieure, qui est dans ce cas φ_i].

φ_1 et φ_2 représentent des accélérations pondérées mesurant l'influence de la mémoire personnelle et l'influence du voisinage de i.

4.2 Algorithme PSO classique

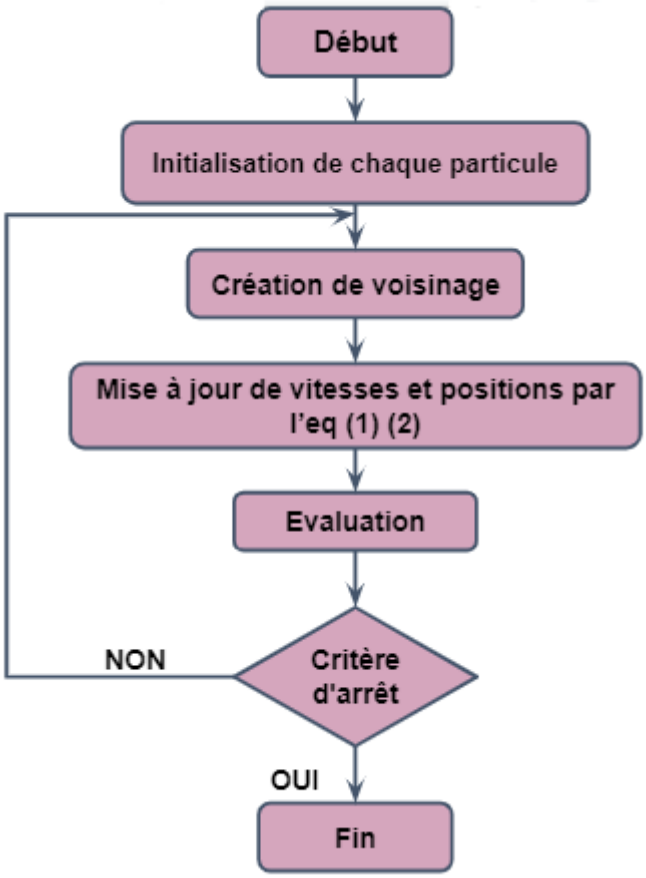


FIG. 4.4 : Schéma de principe de l'algorithme PSO[140]

Début

Initialiser les paramétres et Ia taille S de I' essaim ;

Initialiser Ies vitesses et Ies positions aléatoires des particules dans chaque dimension de I 'espace de recherche ;

Pour chaque particule, Pb X ;

Calculer f(X) de chaque particule ;

Calculer Pg ; // la meilleure pb

Tantque (Ia condition d 'arrét n 'est pas vérifiée) faire

Pour (i allant de I à S) faire
Calculer Ia nouvelle vitesse I 'aide de l'equation (1)
Trouver Ia nouvelle position I 'aide de l'équation (2) ;

Calculerf(X) de chaque particule ;
Si f (J esr meilleure que f(Pb) alors
Pb = X ;
Si f (Pb) est meilleure que f (Pg) alors
Pg= Pb ;

Fin pour

Fin tant que

Fin

Aficher Ia meilleure solution trouvée Pg.

4.3 L'application de l'PSO à un problème :

Afin d'utiliser PSO efficacement, plusieurs paramètres sont mis en jeu, ils doivent être définis avec succès. Parmi ces paramètres, on peut citer essentiellement :

4.3.1 Le codage des solutions :

Le type de codage dépend du problème. Les solutions sont généralement codées comme étant des vecteurs appartenant à un espace vectoriel de dimension D, qui est la dimension du problème. Cela induit la définition de deux bornes (inférieure et supérieure) suivant chaque dimension.

4.3.2 La fonction objectif :

Elle dépend également du problème. Elle représente un moyen pour mesurer la qualité de chaque solution. Cette fonction doit être capable de créer un ordre total dans l'espace des solutions.

4.3.3 La taille de la population :

Ce paramètre a une influence sur le comportement de l'algorithme. En effet, une petite population ne crée pas assez d'interactions garantissant le bon fonctionnement de l'algorithme.

4.3.4 Les coefficients d'accélération :

Ils prennent souvent la même valeur. Si φ_i est trop petit, alors l'algorithme explorera très lentement, ce qui dégrade sa performance.

4.3.5 Le meilleur voisinage :

C'est un aspect de l'algorithme qui est le moins étudié dans la littérature. Dans la première présentation de l'algorithme, le meilleur voisinage est le même pour toute la population.

4.3.6 Critère d'arrêt :

Le critère d'arrêt diffère suivant le problème d'optimisation posé et les contraintes de l'utilisateur. De ce fait, il est fortement conseillé de doter l'algorithme d'une porte de sortie en définissant un nombre maximum d'itération (que nous noterons bItermax). L'algorithme doit alors s'exécuter tant que l'un des critères de convergence suivant n'a pas été atteint [141]. Cela peut être :

1. Un nombre fixe d'itérations ;
2. En fonction de la fonction objectif ;
3. Lorsque la variation de vitesse est proche de 0.

5 Optimisation multiobjectif par essaims particulaires

Il est évident que l'algorithme original PSO doit être modifié pour être adapté à la résolution des problèmes d'optimisation multiobjectifs. Comme on a vu, l'ensemble des solutions d'un problème avec multiples objectifs ne se compose pas d'une seule solution (comme dans l'optimisation globale).

Cependant, dans l'optimisation multiobjectif, il est nécessaire de trouver un ensemble de solutions (l'ensemble Pareto optimal).

Plusieurs méthodes d'adaptation du PSO mono-objectif au PSO multiobjectif (MOPSO) ont été proposées et référencées par Coello Coello et Reyes-Sierra [142]. Trois points sont spécifiquement traités par les méthodes du MOPSO concernant :

1. Le choix des particules meneuses du voisinage (gbest) ou du souvenir (pbest) de chaque particule parmi l'ensemble des particules non dominées ;
2. La méthode de conservation des particules non dominées selon Pareto trouvées durant l'ensemble du processus de recherche ;
3. Le maintien d'une bonne diversité dans l'essaim pour éviter la convergence vers une solution unique.

6 Résolution du problème

6.1 Présentation de la méthode PSO proposée

6.1.1 La structure d'une particule

Pour la formulation MOPSO du problème de mapping et de personnalisation de topologie, une particule correspond à un mapping de coeur IP sur les tuiles et une affectation de TSVs sur des positions candidates. Par exemple, la figure 4.5 montre une application de 8 cœurs mappé sur une topologie Mesh 2×2×2 avec 2 TSVs.

	Plan 1				Plan 2			
Tuile	1	2	3	4	5	6	7	8
IP	IP4	IP6	IP5	IP7	IP1	IP3	IP8	IP2
TSV	1	0	0	1	1	0	0	1

FIG. 4.5 : Structure d'une particule

En supposant que les tuiles sont numérotées dans l'ordre croissant du coin supérieur gauche au coin inférieur droit dans une topologie 2D, une particule peut être considérée comme un vecteur, dans lequel la particule[i] note le noyau mappé sur la $i^{ème}$ tuile. Pour les topologies 3D, la numérotation augmente de la couche la plus basse à la couche la plus élevée.

6.1.2 Évolution des générations

Dans le cadre général du MOPSO, supposons que la position d'une particule (dans un espace à n dimensions) à la $k^{ème}$ itération soit $p_k =< p_{k,1}, p_{k,2}, ..., p_{k,n} >$. Pour la $i^{ème}$ particule, la quantité est noté p_k^i. p_k^i se déplace en prenant en compte :

- Sa meilleur position PLocalBest ;
- La meilleur position de l'essaim PGlobalBest.

La position d'une particule représente le mapping des IPs et l'affectation des TSVs, la vitesse représente l'ensemble de permutations nécéssaires pour obtenir une nouvelle particule (position). La nouvelle position de la particule i calculée comme suit :

$p_{k+1}^i = (s_2 * (pk \to PLocalBest) \oplus s_3 * (p_k \to PGlobalBest)) \cdot p_k^i$ [115](3)

Dans l'expression ci-dessus, a → b représente une séquence de permutations appliquées sur les composants de a pour le transformer en b. Par exemple, si a = <1, 3, 4, 2> et b = <2, 1, 3, 4>, a → b = <permute (1, 4), permute (2, 4), permute (3 , 4)>. L'opérateur ⊕ est l'opérateur de fusion. Pour deux séquences de permutation a et b, $a \oplus b$ est égal à la séquence dans laquelle la séquence de permutation en a et suit de la séquence de permutation a en b. Les constants s_2, s_3 sont les valeurs de confiance en soi et de confiance en essaim. La quantité $s_i * (a \to b)$ signifie que les permutations dans la séquence $a \to b$ seront appliqués avec un coefficient s_i. La permutation final correspondant à $(s_2 * (pk \to PLocalBest) \oplus s_3 * (p_k \to PGlobalBest))$ est appliqué sur une particule p_k^i pour générer p_{k+1}^i.

Pour appliquer une séquence de permutation a → b avec un coefficient s on supprime (tronque) m permutations de la séquence, tel que : m=E($s * n$ombre de permutation dans la séquence) avec E est la partie entière.

Les procédures Calculer_Sequence_Permutation_mapping et Calculer_Sequence_Permutation_TSV montrent la stratégie pour déterminer une séquence de permutation.

Procédure Calculer_Sequence_Permutation_Mapping

Entré : Séquence **Source**, Séquence **Cible** ;

Sortie : **Séquence_Permutation** pour Aligner Source vers Cible ;

Début

Source_mod = Source ;

Pour i = 1 jusqu'à longeur(Source)-1
cible_i= *Cible*[*i*] ; // numéro cible
position_cible_i= Chercher(cible_i ; Source_mod) ;
Si (i != position_cible_i) // permuter et insérer dans Séquence_Permutation
Source_mod[position_cible_i] = Source_mod[i] ;
Source_mod[i] = cible_i ;
Insérer(i, position_cible_i , Séquence_Permutation) ;

FIN.

Procédure Calculer_Sequence_Permutation_TSV

Entré : Séquence Source, Séquence Cible ;

Sortie : Séquence_Permutation pour Aligner Source vers Cible ;

Début

Pour i = 1 jusqu'à longeur(Source) faire
Si (Source[i] !=Cible[i]) alors
Insérer(i , Séquence_Permutation) ;

FIN

6.1.3 Algorithme MOPSO proposée

Initialisation :

Pour chaque particule
Initialiser la particule avec une solution ;
Evaluer chaque particule ;
Mettre à jours **l'archive local** de chaque particule avec elle même ;
Mettre à jours **l'archive global** avec les meilleures particules ;

Evolution :

Faire

Pour chaque particule pi

Choisir un leader de **l'archive local** de p_i ; // de façon aléatoire

Choisir un leader **l'archive global** ; // de façon aléatoire.

Calculer **SPl_mapping** et **SPl_TSV** ;

Calculer **SPg_mapping** et **SPg_TSV** ;

p_i^{new} = Modifier p_i en appliquant **SPl_mapping** et **SPl_TSV** avec un coefficient s_2 suivi par **SPg_mapping** et **SPg_TSV** avec un coefficient s_3 ;

Evaluer le fitness de p_i^{new} ;

Mettre à jours **l'archive local** de p_i ;

Mettre à jours **l'archive global** ;

Jusqu'à (nombre maximum d'itération)

6.1.4 Heuristique pour la génération de la population initiale[115]

Initialiser l'algorithme du mapping Map_Graph

Entré : graphe d'application APG , graphe d'architecture ARG

Sortie : mapping APG dans ARG

Début

trier les bords d'APG par ordre décroissant des coûts de communication.

Pour chaque position de routeur T de ARG

Marquer tous les noeud de APG comme non mappé

Meilleur_Coût = ∞

Meilleur_Mapping = Ø

Mapping = Trouver_Mapping(APG ,ARG ,T) ;

sortie Mapping comme particule

FIN

Procedure Trouver_Mapping

Entré : graphe d'application APG, graphe d'architecture ARG

noeud : noeud à mapper

Commencer_pos : Position dans ARG où le 1ere noeud mappe

Sortie : le mapping de tous les noeuds de APG dans ARG et le 1ere noeud orienté vers Commencer_pos

Début :

soit (c1, c2) le bord d'APG avec la bande passante requise la plus élevée

$coût_1 = \sum_{c_i \in voisin(c_1)}$ *la bande passante requise de* (c_1, c_i)

$coût_2 = \sum_{c_i \in voisin(c_2)}$ *la bande passante requise de* (c_2, c_i)

si ($coût_1 > coût_2$) alors noeud = c_1 sinon noeud= c_2 ;

Mapping [Commencer_pos]= noeud ;

marquer noeud comme mappé

tant qu'il existe unmapped noeuds dans APG alors

soit (c_i, c_j) le bord d'APG avec la bande passante requise la plus élevée tel qu'exactement l'un des $c_i$et c_j est déjà mappé

soit c = c_i if c_j est déjà mappé sinon $c = c_j$

Positions = ensemble de position dans ARG avec une distance de saut de la position déjà mappée

Evaluer_Positions(Positions) ;

Min_Positions= ensemble de Positions avec un coût minimal

si (cardinalité de l'ensemble Min_Positions =1)

Meilleur_Position= Min_Positions[0] ;

sinon Meilleur_Position= Prédire_meilleur(Min_Positions,APG ,ARG, c) ;

Mapping[Meilleur_Position] =c ;

marquer c mappé

return Mapping

Fin

Procédure Prédire_meilleur

Entré : graphe d'application APG, graphe d'architecture ARG

noeud : noeud à mapper

Posn : ensemble de positions rivales d'ARG

Sortie : meilleure position prédite du noyau parmi les posn

Début

Min_coût =∞

noeuds_nouvellement_mappés = ∅

pour chaque position p dans posn faire

Mapping[noeud]= p ;

noeuds_nouvellement_mappés =noeuds_nouvellement_mappés {noeuds}

marqué noeud mappé

tant qu'il existe un noeud non mappé dans APG alors

soit (c_i, c_j) le bord d'APG avec la bande passante requise la plus élevée tel qu'exactement l'un des c_i et c_j est déjà mappé

soit c = c_i if $c_j$est déjà mappé sinon $c = c_j$

Positions = ensemble de position dans ARG avec une distance de saut de la position déjà mappée

Evaluer_Positions(Positions) ;
Meilleur_Position = premiere Position avec un coût minimal
Mapping[Meilleur_Position]=c
marqué c mappé
coût = totale coût total de communication pour ce mapping
si Min_coût = coût alors ;
Min_Posn=p ;
décoché tous les noeuds dans noeuds_nouvellement_mappés
noeuds_nouvellement_mappés = ∅
retourné Min_Posn

FIN

6.2 Méthode exacte

Les méthodes exactes cherchent à trouver de manière certaine la solution optimale en examinant de manière explicite ou implicite la totalité de l'espace de recherche. Elles ont l'avantage de garantir la solution optimale néanmoins le temps de calcul nécessaire pour atteindre cette solution peut devenir très excessif en fonction de la taille du problème et le nombre d'objectifs à optimiser. Le but de l'utilisation de ce type de méthode afin de comparer les résultats obtenus avec les résultats obtenus en utilisant la méthode précédente et garantit sa fiabilité et efficacité.

7.2.1 La méthode proposée

En entrée : **APG** .
En sortie :
Le temps d'exécution pour trouver les solutions Pareto.
Pour chaque solution : Coût de communication Nombre de LV (TSV) Mapping Topologie 3D personnalisée (positions des TSVs)

Début :

Pour chaque casMapping jusqu'à le nbr_mapping_max// Le nombre des cas de mapping possibles est le factoriel des nombres des IPs d'un APG

Début

Pour chaque topologie jusqu'à nbr_topologie_max// cette boucle représente le nombre et l'emplacement des TSVs

Début

Pour chaque transfert k jusqu'à nbr_transfert_max // cette boucle pour calculer le coût de communication d'un mapping

Début

coût total +=coût du transfert k

Fin

Fin

Comparer la solution obtenu avec l'ensemble des solutions

Solution acceptée ssi elle vérifie les conditions de dominance au sens de Pareto

Fin

Fin

7 Conclusion

Dans ce chapitre, nous avons présenté les deux méthodes proposées. Nous avons commencé par introduire notre problème en formule mathématique, puis on a présenté notre supposition. Ensuite, on a décrit les algorithmes utilisés. Dans le chapitre suivant, après l'implémentation de nos algorithmes nous terminerons notre travail avec des tests de validation de notre proposition et nous réaliserons une étude comparative entre les deux méthodes.

Chapitre 5

Tests et résultats

1 Introduction

Après avoir implémenté les différents algorithmes détaillés dans le chapitre précédent, nous présentons les différents résultats obtenus lors de l'expérimentation. Nous commençons par une présentation des différents benchmarks sur lesquelles nous avons effectué nos tests, puis nous allons étudier les résultats obtenus après avoir fait une série de tests sur ces benchmarks.

La méthode exacte proposée permet de donner l'ensemble optimal des solutions de mapping, ainsi que l'emplacement et le nombre des TSVs. Étant donné les deux objectives sont antagonistes nous avons opter pour l'optimisation multi objective afin d'obtenir des solutions de compromis.

Pour l'implémentation de la méthode exacte on a opté pour le langage C sous Unix vu sa rapidité d'exécution (langage compilé), pour la méthode MOPSO on a choisi le langage JAVA car il est multiplate-forme (langage interprété). Les tests sont réalisés sur une machine équipée d'un processeur Intel Core i3 2 GHz et d'une RAM de 6 GO sous le système d'exploitation Ubuntu de 64 bits.

2 Présentation des benchmarks

Dans notre expérimentation on a utilisé 4 benchmarks standards qui sont VOPD, MPEG, MWD, PIP et 2 aléatoires.

2.1 Benchmark VOPD (Video Object Plane Decoder)

L'application VOPD est composée de 16 composants (IPs) qui communiquent entre eux à travers 21 liens.

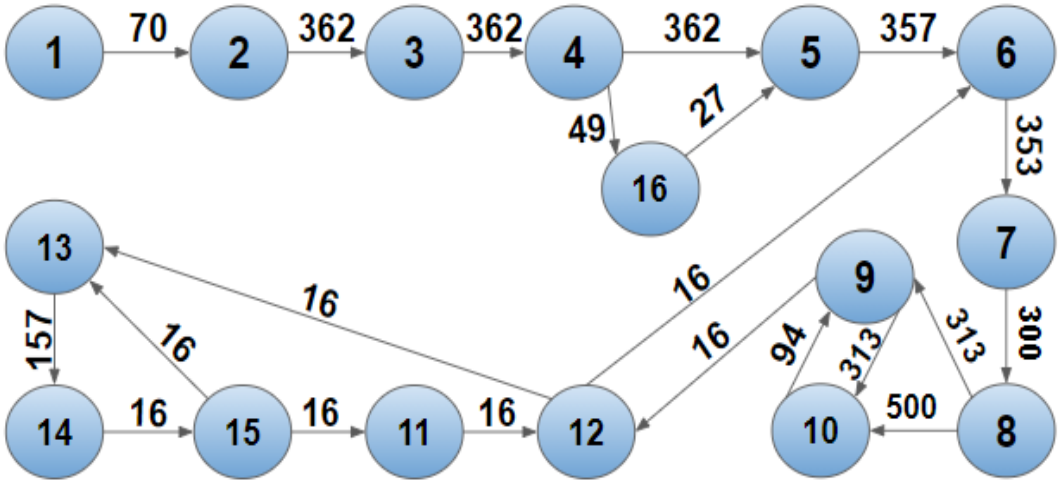


FIG. 5.1 : Benchmark VOPD

2.2 Benchmark MPEG

Le benchmark MPEG possède 12 nœuds et 13 liens, la figure suivante montre le graphe d'application de ce benchmark.

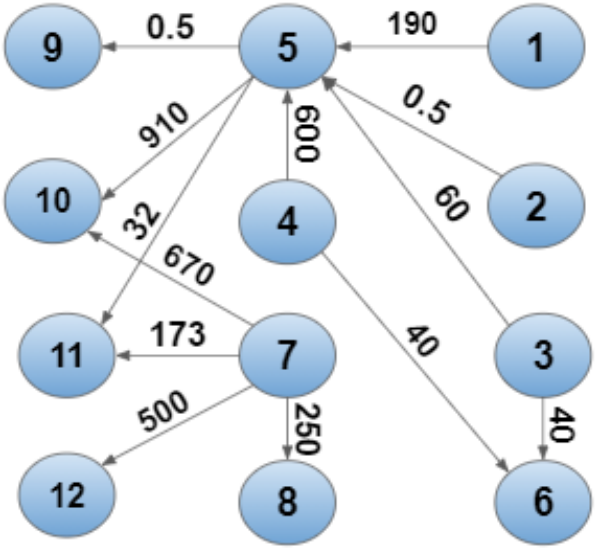


FIG. 5.2 : Benchmark MPEG

2.3 Benchmark MWD

Le benchmark MWD possède 12 nœuds et 13 liens, la figure suivante montre le graphe d'application de ce benchmark.

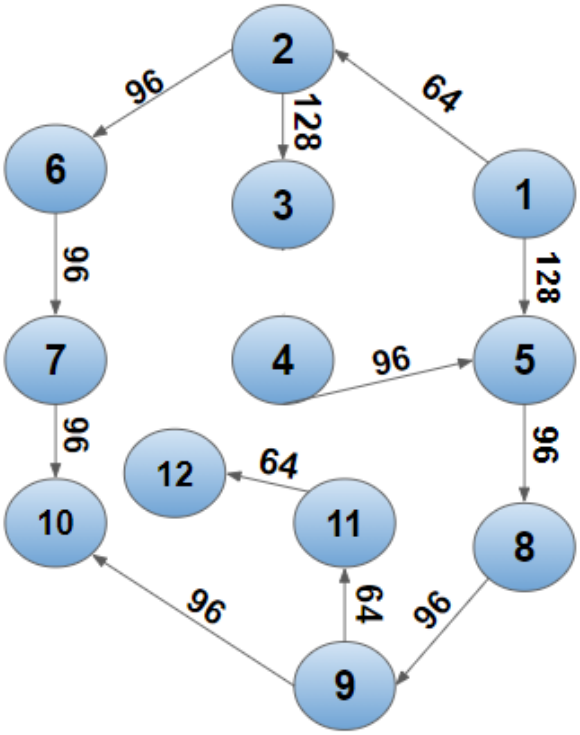


FIG. 5.3 : Benchmark MWD

2.4 Benchmark PIP

Le benchmark PIP possède 8 nœuds et 7 liens, la figure suivante montre le graphe d'application de ce benchmark.

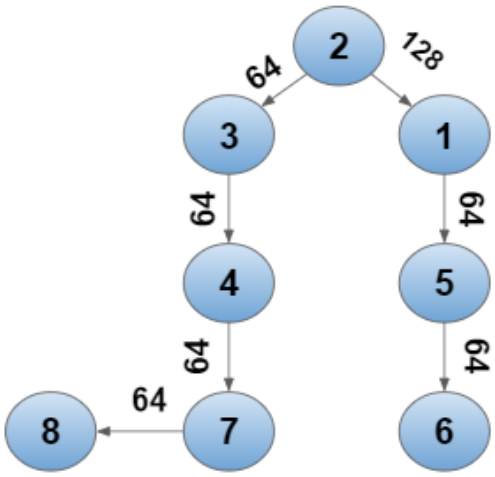


FIG. 5.4 : Benchmark PIP

2.5 Benchmark Aléatoire

Deux benchmarks aléatoires vont être utilisés pour tester la méthode proposée. Benchmark application 1 possède 40 nœuds et 40 liens. Benchmark application 2 possède 80 nœuds et 81 liens (voir l'annexe A pour l'APG de ces benchmarks).

2.6 Architecture des benchmarks

Les applications sont mappées sur des structures de mapping 2D et 3D avec des tailles de mapping indiquées dans le tableau 5.1.

TAB. 5.1 : Différentes applications et tailles des topologies 2D et 3D Mesh

Benchmark	2D MESH	3D MESH
PIP	4x2	2x2x2
MWD	4x4	2x3x2
MPEG	4x4	2x3x2
VOPD	4x4	2x4x2
40IP	8x5	4x5x2
80IP	10x8	5x8x2

3 Etude et résultats des tests de la méthode exacte

Pour tester la méthode implémentée nous considérons le coût de communication et le coût en surface comme fonctions objectives, le coût de TSVs va prendre deux valeurs, 1 pour comparer les résultats avec la littérature, et 0.8 étant donné le coût d'un TSV (α) est inférieur au coût d'un lien horizontal, les valeurs liées aux unités à savoir le coût de communication et la surface sont ignorés, ainsi que les contraintes physiques de fabrications.

3.1 Mono-objectif

Dans un premier temps on fixe le pourcentage de TSVs selon les pourcentages fixés dans la littérature (25%, 50%, 100%) et on calcule le coût de communication et le temps CPU, le tableau suivant représente les résultats obtenus :

TAB. 5.2 : Résultats de la méthode exacte version monoobjectif

	Topologie	α	Le coût de communication	Le temps d'exécution
PIP	2D [115]		640	1.340 s
	3D 25%	0.8	742.2	6.2s
		1	768	6.2s
	3D 50%	0.8	588.8	11s
		1	640	11 s
	3D 100%	0.8	563.2	7.4s
		1	640	7.4s
MWD	2D [115]		1120	200.510 s
	3D 25%	0.8	1177.6	76510 s
		1	1216	76510s
	3D 50%	0.8	1139.2	98060 s
		1	1216	98060s
	3D 100%	0.8	1113.6	69201s
		1	1120	69201s
MPEG	2D [115]		3567	21.530s
	3D 25%	0.8	3342.4	75200s
		1	3567	75200s
	3D 50%	0.8	3322.5	63800s
		1	3567	63800s
	3D 100%	0.8	3322.5	27000s
		1	3567	27000 s
VOPD	2D [115]		4119	4474.730s
	3D 25%	0.8	3937.4	86445s
		1	4141	86445s
	3D 50%	0.8	3846.2	107240s
		1	4141	107240s
	3D 100%	0.8	3700.2	95426s
		1	4103	95426s

Discussion

Les pourcentages fixés sont les mêmes utilisés dans la littérature mais pour d'autres objectifs (la latence, consommation d'énergie), donc on les a testés pour une future comparaison.

On peut remarquer l'utilité de la topologie 3D lorsque le nombre d'IPs est grand, le coût dans le benchmark VOPD avec la topologie 2D maillée égale à 4119 tandis que avec la topologie 3D maillée égale à 4103. On conclut qu'il y a une relation directe entre le temps CPU et le nombre d'IPs avec une augmentation exponentielle dans le temps, en revanche la possibilité d'obtenir le coût minimal sans utiliser un pourcentage de 100 par exemple le benchmark MPEG lors d'affecter un % de 50 on obtient un coût égal à 3322.5 et on obtient le même coût avec un pourcentage de 100 %.

3.2 Multiobjectif

Afin de pouvoir tester l'impact des réseaux sur puce 3D partiellement interconnecté en considérant plusieurs critères en même temps, on a proposé un outil de calcul qui permet de trouver un ensemble de solutions exactes.

Le tableau 5.3 représente le nombre de solution trouvé , le coût de communication minimal en montrant le % et le nombre de TSVs utilisé , le coût lorsqu'on utilise un nombre minimal de TSV et le temps CPU pour chaque benchmark (Pour consulter les archives de solutions voir Annexe B).

TAB. 5.3 : Résultats de la méthode exacte version multiobjectif

	Taille de l'archive (nombre de solutions)	La moyenne (coût de communication, nombre de TSVs)	α	Meilleur coût nombre de communication (avec de TSVs de % TSVs)	Meilleur coût en nombre de TSVs (coût de communication correspondant)	Le temps CPU
PIP	4	(617.6, 2.5)	0.8	563.2 (4, 100%)	1 (742.2)	13s
			1	640 (2, 50%)	1 (768)	
MWD	4	(1208, 2.5)	0.8	1113.6 (4, 66.7%)	1 (1369.6)	10 jr
			1	1120 (4, 66.7%)	1 (1408)	
MPEG	3	(3385.4, 2)	0.8	3322.5 (3, 50%)	1 (3483.5)	8 jr
			1	3567 (2, 25%)	1 (3672)	
VOPD	6	(3934.2, 4)	0.8	3700.2 (7, 87.5%)	1 (4290,8)	14 jr
			1	4103 (7, 87.5%)	1 (4354)	

Discussion :

Le but de ce test est de déterminer est ce qu'il existe d'autres pourcentages (configuration TSV) hors ceux déjà mentionner dans le tableau 5.1 (25, 50 et 100%), qui permettent de donner des performances optimums pour le réseau sur puce(NoC).

En effet on peut remarquer que dans le benchmark MWD le coût minimal est obtenu lors de l'utilisation de 66.67% de TSVs (4 TSVs) par contre d'après le tableau 5.1 ce coût est obtenu lors de l'utilisation de 100% de TSVs, donc notre application consomme moins de surface par rapport à la littérature, la même chose est observé dans le benchmark VOPD. Cependant on remarque que le temps CPU prend jusqu'à des jours afin de donner les solutions exactes d'un benchmark.

4 Etude et résultats des tests de la méthode MOPSO

4.1 Mono-objectif

La recherche [115] en implémenter une heuristique pour la génération de la population initiale dans les topologies 2D mesh et 3D mesh avec une population de taille 100, nombre d'itérations égales à 10, s2=0.5 et s3=0.25. On veut savoir si cette méthode donne toujours des résultats pertinents si le % de TSVs utilisé n'est pas 100%.

Le tableau 5.4 résume les résultats des tests dans différents benchmarks.

TAB. 5.4 : Résultats de PSO version mono objectif

	Topologie	α	Le coût de communication avec heuristique	Le coût communication sans heuristique	Le temps de CPU avec heuristique	Le temps de CPU sans heuristique
PIP	2D		640	640	0.105 s	0.07 s
	3D 25%	0.8	742.4	742.4	0.152 s	0.133 s
		1	768	768	0.149 s	0.125 s
	3D 50%	0.8	588.8	588.8	0.154 s	0.158 s
		1	640	640	0.205 s	0.186 s
	3D 100%	0.8	563.2	614.4	0.235 s	0.244 s
		1	640	640	0.211 s	0.174 s
MWD	2D		1248	1312	0.128 s	0.066 s
	3D 25%	0.8	1235.2	1420	0.273 s	0.261 s
		1	1312	1548	0.261 s	0.276 s
	3D 50%	0.8	1184	1318.4	0.235 s	0.226 s
		1	1248	1472	0.235 s	0.197 s
	3D 100%	0.8	1164	1280	0.326 s	0.326 s
		1	1216	1331	0.326 s	0.384 s
MPEG	2D		3772.5	4022.5	0.104 s	0.244 s
	3D 25%	0.8	3490.4	4066.7	0.47 s	0.248 s
		1	3617	4122	0.535 s	0.248 s
	3D 50%	0.8	3483.5	3749.85	0.311 s	0.233 s
		1	3584	3792	0.421 s	0.233 s
	3D 100%	0.8	3322.5	3401.412	0.427 s	0.309 s
		1	3567	3693	0.427 s	0.309 s
VOPD	2D		4141	4680	0.202 s	0.163 s
	3D 25%	0.8	4124	6248.4	0.779 s	0.356 s
		1	4235	6534	0.957 s	0.413 s
	3D 50%	0.8	3923.4	4735.599	0.702 s	0.504 s
		1	4289	4950	0.862 s	0.578 s
	3D 100%	0.8	3841.6	4704	1.063 s	0.669 s
		1	4141	4865	1.189 s	0.651 s
40IP	2D		3396	6494	4.731 s	0.338 s
	3D 25%	0.8	3316	6663.2	5.706 s	2.054 s
		1	3542	7766	6.582 s	1.848 s
	3D 50%	0.8	3327.6	6598	5.833 s	1.928 s
		1	3423	6770	6.152 s	1.593 s
	3D 100%	0.8	3085.2	4676.4	6.235 s	1.636 s
		1	3273	5552	6.632 s	1.983 s
80IP	2D		18108	20130	0.775 s	0.934
	3D 25%	0.8	7960	21026.4	197.82 s	5.651 s
		1	8217	21341.2	173.235 s	5.942 s
	3D 50%	0.8	7295.2	20845.203	210.083 s	6.69 s
		1	7521	21752	206.302 s	6.874 s
	3D 100%	0.8	6912.4	16523.998	161.204 s	10.245 s
		1	7210	18045.605	158.58 s	9.985 s

Discussion :

Le tableau 5.4 illustre que l'utilisation d'une heuristique pour la génération de la population initiale améliore considérablement le coût prenons par exemple le benchmark VOPD avec une affectation de 50% de TSV dans le cas où on utilise l'heuristique le coût égal 3923.4 par contre sans l'utilisation le coût égal 4735.6, si on néglige la différence de temps CPU on conclut que l'utilisation de l'heuristique est très intéressante.

On constate aussi que, quand on passe d'une topologie 3D 100% connectée verticalement vers une topologie 3D 25% connectée verticalement, le coût de communication se dégrade d'environ 20% en moyenne. Et si on passe d'une topologie 3D vers une topologie 2D la dégradation du coût de communication est plus importante surtout pour les benchmarks de grande taille, 60% pour le benchmark aléatoire de 80 IPs.

On a comparé nos résultats avec ceux de PSMAP [115] et de NMAP [117] pour les benchmarks MPEG et VOPD.On a remarqué que notre application donne de meilleur résultats que NMAP 3567 contre 3672 pour MPEG et 4199 contre 4141 pour VOPD , et des résultats assez proche de ceux de PSMAP 4119 contre 4141 pour VOPD et pour MPEG on a les mêmes résultats.

4.2 Multiobjectif

4.1.1 Effet de variation de s2 et s3 :

Le but de cette étude est de tester l'effet de variation des coefficients S2 et S3 sur le coût. Comme il y a une infinité de combinaisons possible, il n'est pas possible de tout les tester, de ce fait, nous avons choisi 10 combinaisons parmi les combinaisons possibles.

- $Nombre\ de\ TSV\ moyen = \frac{\sum TSV\ des\ solutions\ des\ Archives}{\sum TailleArchive\ Des\ Archives}$
- $Coût\ moyen = \frac{\sum Coût\ des\ solutions\ des\ Archives}{\sum Taille\ Des\ Archives}$
- $Nombre\ de\ solution\ optimal\ moyenn = \frac{\sum taille\ des\ archive}{Nombre\ d'exécution}$

Le tableau 5.5 représente les résultats obtenues aprés 10 exécution en variant s2, s3 avec une taille de population est égale à 100 ainsi que le nombre d'itération .

TAB. 5.5 : Tests fait pour fixer s2 et s3

	s2	s3	Nombre de solution	Le coût moyen (α =0.8)	Nombre de TSV Moyen
PIP	0.9	0.2	3	686.93	2
	0.75	0.15	4	678.4	2
	0.15	0.7	2	620.8	2
	0.2	0.1	3	624	2
	0.1	0.2	4	678.4	3
	0.4	0.2	4	674.6	3
	0.5	0.25	3	642.26	2
	0.4	0.9	4	617.6	3
	0.6	0.3	4	682.6	3
	0.32	0.98	4	620	3

	s2	s3	Nombre de solution	Le coût moyen (α =0.8)	Nombre de TSV Moyen
MWD	0.9	0.2	3	1425.06	2
	0.75	0.15	3	1331.2	3
	0.15	0.7	3	1380	2
	0.2	0.1	3	1321.6	4
	0.1	0.2	3	1329	2
	0.4	0.2	3	1280	3
	0.5	0.25	3	1320	3
	0.4	0.9	3	1384.53	2
	0.6	0.3	3	1359.06	3
MPEG	0.9	0.2	5	3668.87	3
	0.75	0.15	3	3804.03	3
	0.15	0.7	4	3831.63	3
	0.2	0.1	4	3583	2
	0.1	0.2	5	3626.76	3
	0.4	0.2	3	3593.7	2
	0.5	0.25	4	3622.21	3
	0.4	0.9	4	3834.09	3
	0.6	0.3	2	3984.3	2
	0.32	0.98	3	3981.53	2
VOPD	0.9	0.2	3	4535.77	2
	0.75	0.15	7	4435.77	5
	0.15	0.7	5	4215.77	4
	0.2	0.1	5	4451.84	3
	0.1	0.2	5	4350.6	3
	0.4	0.2	5	4249.03	4
	0.5	0.25	5	4215.46	3
	0.4	0.9	3	4085.68	4
	0.6	0.3	5	4428.03	3
	0.32	0.98	3	4105.33	4
40IP	0.9	0.2	6	3700.87	6
	0.75	0.15	5	3802.97	7
	0.15	0.7	7	3648.96	7
	0.2	0.1	6	3712.71	7
	0.1	0.2	9	3730.73	6
	0.4	0.2	6	3887.82	8
	0.5	0.25	7	3731.06	4
	0.4	0.9	6	3636.46	6
	0.6	0.3	9	3709.56	8
	0.32	0.98	10	3603.52	10
80IP	0.9	0.2	10	8261.53	15
	0.75	0.15	8	8455.85	17
	0.15	0.7	6	8374.75	9
	0.2	0.1	6	8100.85	14
	0.1	0.2	11	8201.16	13
	0.4	0.2	8	8045.4	12
	0.5	0.25	9	8411.29	16
	0.4	0.9	10	8686.2	14
	0.6	0.3	5	8612.2	6
	0.32	0.98	8	8564	14

Discussion

Pour les benchmarks PIP et VOPD on remarque que les meilleurs résultats, en terme de coût moyen réduit, ainsi que le nombre des interconnexions verticaux utilisés, sont obtenu lorsque les coefficients S2, S3 sont compris dans [0.1-0.5] et [0.6-1[respectivement, des résultats moins bons sont constaté lorsque S2 est comprise dans [0.6-1[et que S3 est comprise dans [0.1-0.5].

Pour le benchmark MWD on remarque que les meilleurs résultats, en terme coût moyenne réduit, ainsi le nombre des interconnexions verticaux utilisés, sont obtenu lorsque les coefficients S2, S3 sont compris dans [0.1-0.5] et que S2>S3, des résultats moins bons sont constaté lorsque S2 et S3 sont compris dans [0.6-1[ou quand S3>S2.

Pour le benchmark MPEG on remarque que les meilleurs résultats, en terme coût moyenne réduit, ainsi le nombre des interconnexions verticaux utilisés, sont obtenu lorsque les coefficients S2, S3 sont compris dans [0.1-0.4] et que S2>S3, des résultats moins bons sont constaté lorsque S2 et S3 sont compris dans [0.5-1[ou quand S3>S2.

Pour les benchmarks aléatoire on a retenus les valeurs {S2=0.32 ; S3=0.98} pour le benchmark de 40 IPs et {S2=0.4 ; S3=0.2} le benchmark de 80 IPs.

4.1.2 Effet de variation de la taille de population

Le tableau 5.6 représente les résultats obtenues aprés 10 exécution en variant la taille de la population avec un nombre d'itération égal à 100.

TAB. 5.6 : Résultats de variation de la taille de population sur les différents benchmarks

Benchmark	Taille de la population	Nombre de solution optimal moyenne	Coût de communication Moyenne (α =0.8)	Nombre de TSV Moyenne
PIP s2=0.4 s3=0.9	50	3	657.1	2
	100	3	639.2	2
	250	4	636.2	3
	500	4	624	3
	1000	4	624	3
MWD s2=0.4 s3=0.2	50	2	1376	3
	100	3	1345.6	3
	250	4	1331.6	3
	500	4	1325.55	2
	1000	4	1324.7	3
MPEG s2=0.2 s3=0.1	50	3	3934.6	3
	100	4	3716.5	3
	250	4	3618.55	4
	500	5	3602.5	2
	1000	6	3593.5	4
VOPD s2=0.4 s3=0.9	50	4	4514.6	4
	100	3	4433.45	4
	250	4	4291.2	3
	500	4	4105.2	4
	1000	5	3976.2	5
40IP s2=0.32 s3=0.98	50	5	3842.08	7
	100	7	3782.56	6
	250	7	3719.9	6
	500	7	3605.9	9
	1000	8	3549.5	8
80IP s2=0.4 s3=0.6	50	5	10553	15
	100	7	9554.9	13
	250	7	8674	17
	500	8	8205.65	18
	1000	7	8173.5	11

Discussion

Pour tous les benchmarks testés, on a constaté que plus la taille de la population augmente, plus le coût est réduit . L'effet de variation de la taille de population influence le nombre de solutions optimales trouvées, ainsi que le coût de communication. Plus la taille de la population est grande, plus on a la chance de tomber sur des résultats optimaux. Si la taille de la population est trop grande la convergence va être lente et l'inverse si la taille de la population est très petite y a des risques de tomber sur local optimum (retrouver les mêmes solutions à chaque fois).

Dans notre série de test la taille de population idéale est égale à 1000, du fait que sa donne plus de solutions optimales trouvées et que le coût de communication/nombre de TSV est le minimum avec cette valeur.

4.1.3 Effet de variation du nombre d'itération

Le tableau 5.7 représente les résultats obtenues aprés 10 exécution en variant le nombre d'itération avec une taille de population = 1000.

TAB. 5.7 : Résultats de variation du nombre d'itération sur les différents benchmarks

Benchmark	Nombre d'itération	Nombre de solution optimal moyen	Cout de communication moyen (α =0.8)	Nombre de TSV moyen
PIP s2=0.4 s3=0.9	100	3	624	3
	200	3	617.6	3
	500	4	624	3
	800	3	619.6	3
MWD s2=0.4 s3=0.2	100	4	1324.7	3
	200	3	1325.6	2
	500	4	1331.6	2
	800	4	1325.55	3
MPEG s2=0.2 s3=0.1	100	4	3593.5	3
	200	4	3588.2	4
	500	5	3595.7	2
	800	5	3592.3	3
VOPD s2=0.4 s3=0.9	100	5	3976.2	4
	200	5	4433.45	4
	500	6	4291.2	3
	800	5	4105.2	4
40IP s2=0.32 s3=0.98	100	6	3842.08	7
	200	6	3782.56	6
	500	7	3719.9	6
	800	7	3605.9	9
80IP s2=0.4 s3=0.2	100	7	10553	15
	200	7	9554.9	13
	500	8	8674	17
	800	8	8205.65	18

Discussion

Pour tous les benchmarks testés, on remarque que 100 itérations est suffisante pour avoir des bons résultats en termes de coût de communication et de nombres de TSVs. L'augmentation du nombre d'itérations n'a pas un impact notable sur le coût ou le nombre de TSV. Cependant, si on compare le nombre de solutions optimales obtenues, on voit qu'il augmente avec le nombre d'itérations. Dans notre série de test le nombre d'itérations idéale est 500 par rapport au nombre de solutions optimales trouvées. Plus le nombre de solutions est grand, plus un concepteur de circuit a le choix, ainsi il peut choisir entre une solution qui optimise le coût de communication, celle qui optimise le nombre de TSVs utilisés ou celle qui donne un compromis entre les deux.

5 Etude comparative :

Le tableau 5.8 illustre les meilleurs paramètres trouvés afin d'obtenir des résultats intéressantes en terme de coût de communication et nombre de TSV. Les benchmarks utilisés sont PIP, MWD, MPEG, VOPD, 40IP et 80IP.

TAB. 5.8 : Récapitulatif des résultats pour le choix des paramètres

Benchmark	s2	s3	Taille population	Nombre d'itération
PIP	0.4	0.9	500	500
MWD	0.4	0.2	1000	500
MPEG	0.2	0.1	1000	500
VOPD	0.4	0.9	1000	500
40IP	0.32	0.98	1000	500
80IP	0.4	0.2	1000	500

Le tableau suivant représente une comparaison entre la méthode exacte et la méthode MOPSO en fonction du coût de communication et nombre de TSVs (Pour consulter les archives de solutions voir Annexe B).

TAB. 5.9 : Comparaison entre les archives des deux méthodes

Benchmark	Méthode exacte			MOPSO		
	nombre de solution optimale	cout de comunication	nombre de TSVs	nombre de solution optimale	cout de comunication	nombre de TSVs
PIP	4	742.5	1	4	742.5	1
		588.8	2		588.8	2
		576	3		576	3
		563.2	4		563.2	4
MWD	4	1369.6	1	5	1369.6	1
		1177.6	2		1235.2	2
		1139.2	3		1203.2	3
		1113.6	4		1196.8	4
					1177.6	5
MPEG	3	3483.5	1	3	3483.5	1
		3342.4	2		3342.4	2
		3322.5	3		3322.5	3
VOPD	6	4290.8	1	7	4409,8	1
		3937,4	2		4124	2
		3881	3		4091,8	3
		3846.2	4		3951,8	4
		3802	5		3887,8	5
		3700.2	7		3868,6	7
					3841.6	8

Le résultat d'une optimisation multiobjectifs pour la méthode exacte et MOPSO est représenté sous forme de graphe ayant comme axes les deux objectifs à optimiser à savoir les coûts communication et en nombre de TSV (surface) (Voir Annexe C).

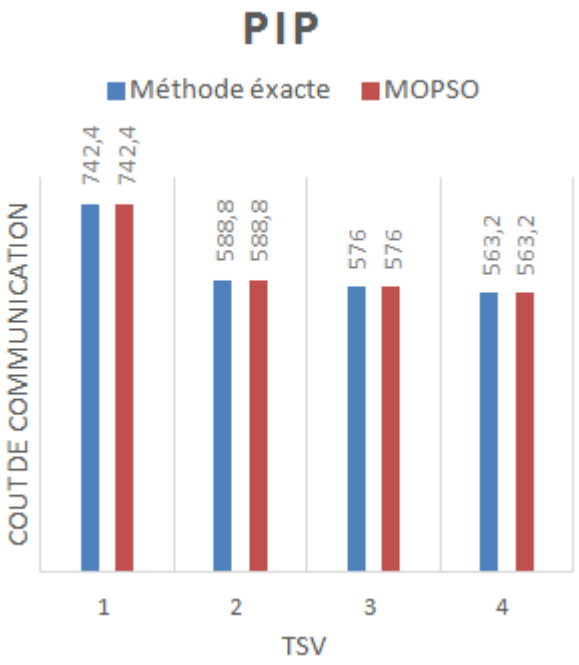


FIG. 5.5 : Comparaison des deux méthodes avec PIP

La figure 5.5 montre le front de pareto du benchmark PIP, on peut remarquer que les résultats des 2 algorithmes sont les mêmes.

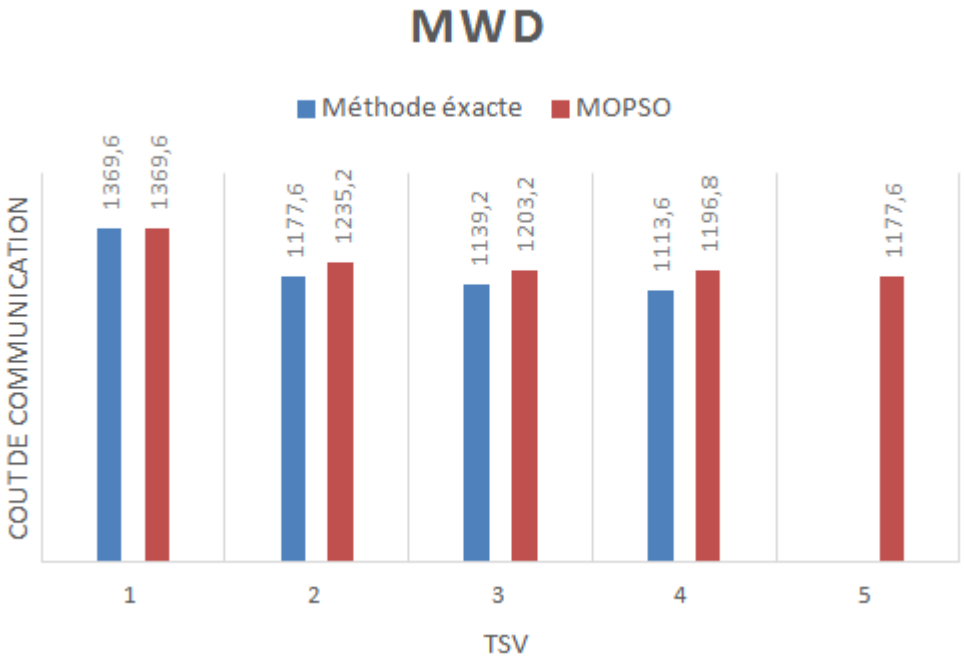


FIG. 5.6 : Comparaison des deux méthodes avec MWD

La figure illustre que pour le benchmark MWD avec la méthode exacte lors de l'utilisation de

4 TSV on obtient un coût minimal égal à 1113.6, en revanche avec MOPSO on obtient un coût égal à 1196.8 (7% de différence). La différence est presque négligeable en prenant en considération le temps CPU qui prend à peu près 10 jours (voir le tableau 5.2) pour la méthode exacte contre 5.3 secondes pour MOPSO.

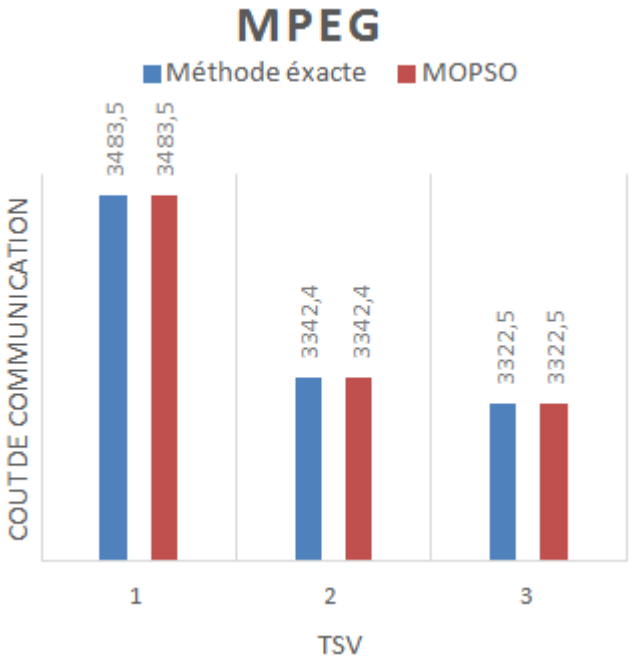


FIG. 5.7 : Comparaison des deux méthodes avec MPEG

D'après la figure on remarque que la méthode MOPSO nous donne les mêmes que ceux de la méthode exacte mais plus rapidement.

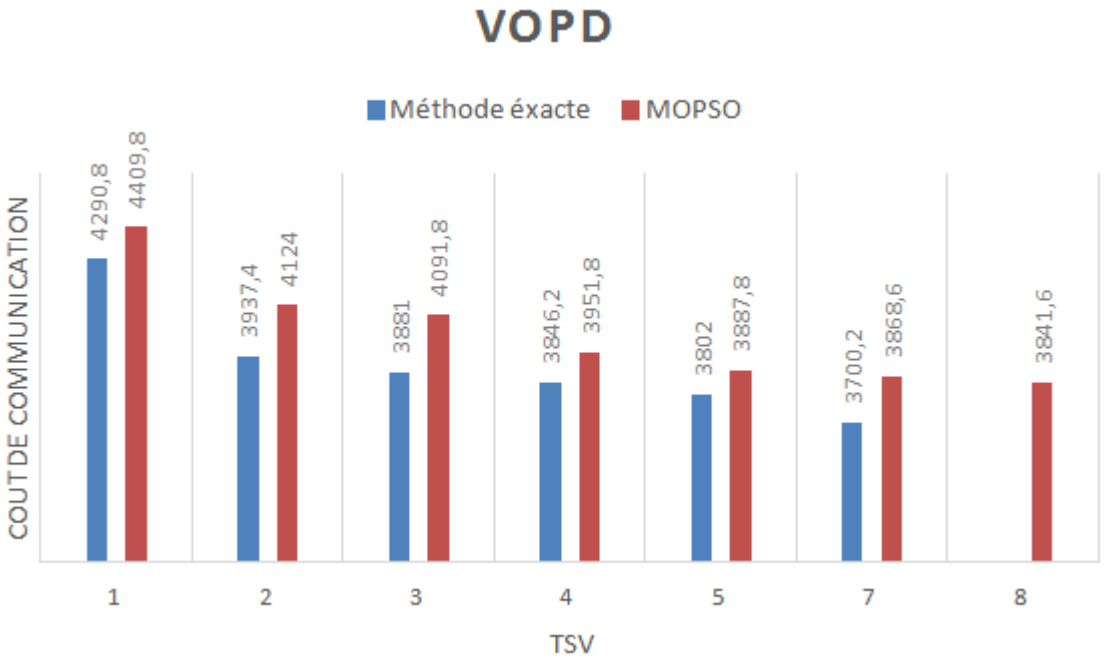


FIG. 5.8 : Comparaison des deux méthodes avec VOPD

Pour le dernier graphe du benchmark VOPD on remarque que la pire dégradation du coût de communication est de 5% citée pour les autres benchmark, donc MOPSO donne des résultats proches de l'optimum en un temps CPU raisonnable.

Discussion

D'après les graphes précédents et le tableau 5.9 on peut remarquer que l'algorithme proposé donne des résultats presque optimales en terme de coût de communication / nombre TSVs par rapport à la méthode exacte.

Pour quelque Benchmark on peut observer une différence où la méthode exacte est meilleure car elle donne des résultats optimums mais l'inconvénient principale de cette dernière est le temps CPU, par exemple pour les benchmarks 40IP et 80IP c'était impossible de les tester car elle peut prendre jusqu'à des mois.

6 Conclusion

Dans ce chapitre, nous avons commencé par présenter les benchmarks utilisées les benchmarks utilisées, puis on a effectué une série de tests de validation de notre proposition et une étude de paramétrage de la méthode MOPSO proposée, on a terminé avec une étude comparative entre la méthode exacte et la méthode MOPSO.

Notre solution prouve son intérêt avec ces valeurs en donnant des bonnes solutions de compromis dans un temps assez raisonnable.

Conclusion générale

L'intérêt d'utiliser les réseaux sur puce comme structure d'interconnexion dans les systèmes sur puce a été démontré par les travaux de recherche étudiés au cours de notre travail.

Lors de la conception des réseaux sur puce il faut suivre un processus en phases ou étapes, cependant chaque étape a ses défis. Citons le mapping, le choix de la topologie du réseau et le choix de la technologie d'intégration. Trouver une combinaison optimale de ces paramètres afin de maximiser les performances d'un NoC est un problème d'optimisation combinatoire. Après avoir passé en revue les différentes solutions proposées dans la littérature, nous avons cherché à emprunter une nouvelle voie pour la résolution du problème. Pour cela et d'après les recherches effectuées, on a proposé de traiter en parallèle la phase de mapping et celle du choix de topologie. C'est ce qui rend ce problème, un problème d'optimisation multiobjectifs.

Le but principal du travail présenté dans ce document est de proposer une technique qui traite en parallèle le mapping d'une application avec le choix de topologie en minimisant le coût de communication et le nombre de lien vertical (TSV) utilisé. Pour cela nous avons utilisé une méthode hybride basée sur deux algorithmes d'optimisation. Un algorithme MOPSO qui est une méta heuristique, et une heuristique pour la génération d'une partie de la population initiale.

Dans les expérimentations réalisées, nous avons pu observer l'efficacité de notre méthode en comparant ses résultats avec celle d'une méthode exacte et avec celle de la littérature, en prouvant l'intérêt d'utiliser des architectures 3D personnalisées. Néanmoins, la qualité des solutions trouvées par notre solution dépend du choix de paramétrage des métaheuristiques (taille de la population, les coefficients S2 et S3, nombre d'itération) et de l'équilibre entre le coût de communication et le coût de surface.

Pour le moment, en attendant des expérimentations et simulation plus poussées, nous considérons que notre méthode MOPSO est efficace parce qu'elle donne de bonnes solutions dans un temps raisonnable.

Dans nos perspectives, nous voulons d'abord approfondir notre expertise dans ce domaine et valider notre contribution par la communauté scientifique. Ensuite, améliorer notre contribution en :

- Procédant au reste des phases de conception des réseaux sur puce 3D, tel que le routage et l'ordonnancement.
- Améliorant notre solution algorithmique par exemple en intégrant une heuristique pour estimer le nombre de TSVs nécessaire.
- Explorant d'autres méthodes d'optimisation combinatoire multiobjectives liées aux problèmes de mapping, aux topologies personnalisées et l'utilisation de la technologie 3D.
- Explorant d'autres méthodes par exemple le deep learning pour résoudre ce problème.

Bibliographie

[1] G.J.Ritchie "Transistor Circuit Techniques discrete and integrated " Chapman & Hall 1993

[2] A.H KAMECHE, "Introduction sur les VLSI ", cours module VLSI, université Saad Dahleb Blida 2019

[3] Z.BELHADJ AMOR. "Validation de systèmes sur pucecomplexes du niveau transactionnelau niveau transfert de registres thèse de doctorat 2014

[4] H.MAYACHE "Mise en oeuvre d'une architecture multiprocesseur autour d'un reseau sur puce (NoC) de type Wishbone" thèse de magister, université de ANNABA 2018

[5] W WOLF et al "Multiprocessor System-on-Chip(MPSoC) Technology" IEEE transactions on computer-aided design of integrated circuits and systems, vol. 27, no. 10, october 2008

[6] N G.Einspruch ,Jeffrey L.Hilbert " Application Specific Integrated Circuit (ASIC) Technology" ACADEMIC PRESS 1991

[7] P Zhang "Advanced IndustrialControl Technology" Elsevier Inc 2010

[8] J Delorme. " Méthodologie de modélisation et d'exploration d'architecture de réseaux sur puce appliquée aux télécommunications ". domain_other. INSA de Rennes, 2007. Français

[9] W.J Dally . et B Towles ., "Route Packets, Not Wires : On-chip Interconnection Networks". Design Automation, USA, (May 2005)

[10] W Stallings "Computer Organization and Architecture : Designing for Performance"(8th. ed.) Prentice Hall Press USA 2009.

[11] M.F Chatmen " Conception d'un réseau sur puce optimisé en latence ". Electronique. thèse de doctorat. Université de Bretagne Sud, 2016.

[12] A Jantsch et H Tenhunen "Networks on Chip". Kluwer Academic Publishers New York, Boston, Dordrecht, London, Moscow 2004

[13] M Brière "Flot de conception hiérarchique d'un système hétérogène. Prototypage virtuel d'un réseau d'interconnexion optique intégré" thèse de doctorat , L'ECOLE CENTRALE DE LYON 2005.

[14] S Kundu et S Chattopadhyay " Network-on-Chip The Next Generation of System-of Chip Integration ". Taylor & Francis Group 2015

[15] SAMMAN, et al . "Micro architecture and Implementation of Networks-on-Chip with a Flexible Concept for Communication Media Sharing". Université technique de Darmstadt, Dissertation (thèse de doctorat), 2010.

[16] R LEMAIRE. "Conception et modélisation d'un système de contrôle d'applications de télécommunication avec une architecture de réseau sur puce (NoC) " thèse de doctorat. INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE 2006

[17] Y.A LARIBI. " Environnement de Mapping pour la Conception des Réseaux sur Puce (NoCs)" thèse d'Ingénieur d'Etat. ESI 2010

[18] W Stallings. " Computer Organization and Architecture : Designing for Performance (8th. ed.)". Prentice Hall Press USA 2009.

[19] S,Yehia" Conception des réseaux sur puce (NoCs) et génération des modèles ," , thèse de doctorat , FACULTE DES SCIENCES DE MONASTI 2012

[20] A Bouguettaya "Génération d'un réseau sur puce au format VHDL RTL à partir d'une modélisation de haut niveau UML par raffinement" thèse doctorat 2017

[21] A Ben Achballah " A Survey of Network-On-Chip Tools " (IJACSA) International Journal of Advanced Computer Science and Applications, Vol. 4, No. 9, 2013

[22] M. Ali, et al "Using the NS-2 Network Simulator for Evaluating Network on Chips (NoC)," in International Conference on Emerging Technologies, 2006.

[23] R. Lemaire, et al., "Performance evaluation of a NoC-based design for MC-CDMA telecommunications using NS-2," in The 16th IEEE International Workshop on Rapid System Prototyping, 2005.

[24] A. Pullini, et al., "NoC Design and Implementation in 65nm Technology," in ACM/IEEE International Symposium on Networks-onChip, 2007

[25] C. Seiculescu, et al., "SunFloor 3D : A Tool for Networks on Chip Topology Synthesis for 3D Systems on Chips," in Proceedings of the conference on Design, Automation and Test in Europe, 2009

[26] L. Benini and G. D. Micheli, "Networks on Chips : A New SoC Paradigm," Computer, 2002.

[27] H. Wang, et al., "Orion : A Power-Performance Simulator for Interconnection Networks," in 35th Annual IEEE/ACM International Symposium on Microarchitecture, 2002

[28] A. Kahng, et al., "ORION 2.0 : A Fast and Accurate NoC Power and Area Model for Early-Stage Design Space Exploration " in Proceedings of Design Automation and Test in Europe, 2009

[29] M. Lis, et al., "DARSIM : a parallel cycle-level NoC simulator " in 6th Annual Workshop on Modeling, Benchmarking and Simulation, 2010.

[30] Y Collette, P Siarry"Optimisation Multiobjectif", ÉDITIONS EYROLLES,2002.

[31] A Berro"Optimisation Multiobjectif et Stratégie d'évolution en environnement Dynamique"(thèse de Doctorat) ,l'université des sciences sociales Toulouse 2001

[32] F.Y. Edgeworth.Mathematical Physics. P. Keagan, London, 1881

[33] V. Pareto.Cours d'économie politique. Rouge, Lausanne, 1896.

[34] Ö, Johan" A Survey of Multiobjective Optimization in Engineering Design", 2000.

[35] J Andersson" Multiobjective optimization in Engineering Design", Applications to Fluid Power System (thèse de Doctorat), 2001

[36] M SAMIR"Optimisation Multiobjectif Par Un Nouveau Schéma De Coopération Méta/Exacte"(thèse de Magister),2006

[37] N Piegay "Optimisation multiobjectif et aide à la décision pour laconception robuste. : Application à une structureindustrielle sur fondations superficielles"(thèse de Doctorat) 2015

[38] I. Othmani ; Optimisation multicritère : Fondements et Concepts. thèse de PHD ; Université de Grenoble ; 1998.

[39] M. Ehrgott ; Multicriteria optimization ; In Lecture Notes in Economics and Mathematical Systems, volume 491 ; Springer ; 2000.

[40] O Mouelhi"Contribution À L'optimisation Multiobjectif En conception multidisciplinaire ", thèse de doctorat ,2013

[41] A ABDELKADER"Ordonnancement multiobjectifs d'application intensive sur architecture régulière" thèse de doctorat ,2012

[42] E. Talbi "Métaheuristique pour l'optimisation combinatoire multiobjectif : Etat d'art " December 2000

[43] F. IRISARRI "Stratégies de calcul pour l'optimisation multiobjectif des structures composites" thèse de doctorat 2009

[44] L ASLI, " Approche hybride pour les problèmes d'optimisation combinatoire multiobjectif : cas des problèmes de type sac-à-dos", thèse de magister USTHB 2010.

[45] T. Sen, M. E. Raiszadeh, and P. Dileepan. A branch and bound approach to the bicriterion scheduling problem involving total owtime and range of lateness. Management Science, 34(2) :254{260, 1988}.

[46] M. Vis ee, J. Teghem, M. Pirlot, and E. L. Ulungu. Two-phases method and branch and bound procedures to solve knapsack problem. Journal of Global Optimization, 12 :139{155, 1998}.

[47] B. S. Stewart and C. C. "White. Multiobjective A*". Journal of the ACM, 38(4) :775{814, 1991}.

[48] L. Mandow and E. Millan. Goal programming and heuristic search. In R. Caballero, F. Ruiz, and R. Steuer, editors, Second Int. Conf. on Multi-Objective Programming and Goal Programming MOPGP'96, pages 48{56, Torremolinos, Spain, May 1s996. Springer-Verlag.

[49] R. L. Carraway, T. L. Morin, and H. Moskowitz. Generalized dynamic programming for multicriteria optimization. European Journal of Operational Research, 44 :95{104, 1990.

[50] D. J. White. The set of e cient solutions for multiple-objectives shortest path problems. Computers and Operations Research, 9 :101{107, 1982.

[51] A HAMDANI,MADI.Z "Application des méthodes exactes pour l'optimisation multicritère de la gestion de spectre dans les réseaux de radio cognitive"(these master), 2018

[52] Ehrgott, Matthias et al "Approximate solution methods for multiobjective combinatorial optimization" TOP : An Official Journal of the Spanish Society of Statistics and Operations Research, 2004.

[53] S. M. Sait et H. Youssef"iterative computer algorithms with applications in engineering : solving combinatorial optimization problems" IEEE computers society, 1999.

[54] E. Bonomi et J.-L. Lutton ,"le recuit simulé pour la science", Juillet 1988.

[55] F. Glover,"Artificial Intelligence, heuristic frameworks and tabu search : Managerial and decision economics" volume 11, 1990.

[56] G. Di Caro and M. Dorigo "AntNet : Distributed stigmergetic control for communications networks".Journal of Artificial Intelligence Research,1998 9,317–365

[57] S. Kemmoé , et al "Des Essaims Particulaires Efficaces Pour l'optimisation Combinatoire" Congrès du GDR MACS, Pôle STP ; Octobre 2004.

[58] C Bontemps," Principes Mathématiques et Utilisations des Algorithmes Génétiques", Novembre 1995.

[59] M Yagoubi." Optimisation evolutionnaire multi-objectif parallele : application a la combustion Diesel". thèse de doctorat, Universite Paris Sud-Paris XI, 2012.

[60] A Spalanzani,"Algorithmes évolutionnaires pour l'étude de la robustesse des systèmes de reconnaissance automatique de la parole" , thèse de doctorat, Octobre 1999 ;

[61] K. E. Parsopoulos, D. K. Tasoulis et M. N. Vrahatis, "Multiobjective Optimization Using Parallel Vector Evaluated Particule Swarm Optimization", 2004

[62] J. Yen, J. Liao, B. Lee, et D. Randolph,"A hybrid approch to modeling metabolic systems using a genetic algorithm and simplex method ; IEEE Transaction on systems, man and cybernetics", vol. 28, 1998. .

[63] C.H. PAPADIMITRIOU, K. STEIGLITZ,"Combinatorial optimization - algorithms and complexity". Prentice Hall, 1982

[64] Agarwal, et al ."Survey of Network on Chip (NoC) Architectures & Contributions." Journal of Engineering, Computing and Architecture 2009

[65] T., Sigenza-Tortosa, et al J. 2004."Topology optimization for application-specific networks-on-chip". In International Workshop on System Level Interconnect Prediction (SLIP). ACM, 53--60

[66] E. Ofori-Attah and M. O. Agyeman, "A survey of recent contributions on low power NoC architectures," 2017 Computing Conference, London, 2017

[67] U. Ogras et R. Marculescu,"It's a small world after all : NoC performance optimization via long-range link insertion," Very Large Scale Integration (VLSI) Systems, IEEE Transactions on, vol. 14, n° 17, p. 693–706, 2006.

[68] EVAIN, Samuel"Uspider environnement de conception de réseau sur puce"institut nationale des sciences appliquées de rennes 2006

[69] L.Benini et al " Networks on Chips Technology and Tools" Morgan Kaufmann ; 1st Edition 2006

[70] Toubaline Nesrine et al "A Classiffcation and Evaluation Framework for NoC Mapping Strategies" Journal of Circuits, Systems, and ComputersVol. 26, No. 2 (2017)

[71] K.Manna, J.Mathew "Design and Test Strategies for 2D/3D Integration for NoC-based Multicore Architectures" Springer Nature Switzerland AG 2020

[72] A.ZIA et al"Highly-Scalable 3D CLOS NoC for Many-Core CMPs" Proceedings of the 8th IEEE International NEWCAS Conference 2010

[73] R. J. Gutmannet al.,"Three-dimensional (3D) ICs : A technology platform for integrated systems and opportunities for new polymeric adhesives,"inProc. Conf. Polymers Adhesives Microelectron. Photon.,2001, pp. 173–180.

[74] J. W. Joyner et al.,"Impact of three-dimensional architectures on interconnects in gigascale integration,"IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 9, no. 6, pp. 922–927, Dec. 2000.

[75] Vasilis F. Pavlidis et al "3-D Topologies for Networks-on-Chip"IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS, VOL. 15, NO. 10, OCTOBER 2007

[76] Brett Stanley Feero et al" Networks-on-Chip in a Three-Dimensional Environment : A Performance Evaluation",IEEE TRANSACTIONS ON COMPUTERS, VOL. 58, NO. 1, JANUARY 2009

[77] Hao Hua et al"Performance Trend in Three-Dimensional Integrated Circuits" IEEE International Interconnect Technology Conference ,2006

[78] B Feero"Performance Evaluation for Three-Dimensional Networks-On-Chip" IEEE Computer Society Annual Symposium on VLSI 2007

[79] M Brocard ”Caractérisation et analyse du couplage substrat entre le TSV et les transistors MOS dans les circuits intégrés 3D” thèse de doctorat l'école doctorale EEATS, 2013

[80] Majumder, ” at Signal integrity analysis in carbon nanotube based through-silicon via." Active and Passive Electronic Components 2014,

[81] D. Velenis, et al"Impact of 3d design choices on manufacturing cost," in IEEE 3DIC2009, Sept. 2009, pp. 1–5

[82] Mohit Pathak et al ”Through-Silicon-Via Management during 3DPhysical Design : When to Add and How Many ?” IEEE 2010

[83] Sung Kyu Lim“TSV-Aware 3D Physical Design Tool Needs for Faster Mainstream Acceptance of 3D ICs"Proc. DAC 2009

[84] G. Chen, F. Li et M. Kandemir, "Reducing Energy Consumption of On-Chip Networks Through a Hybrid Compiler-Runtime Approach," chez Parallel Architecture and Compilation Techniques. 16th International Conference on, 2007.

[85] U. Ogras et R. Marculescu, “It's a small world after all : Noc performance optimization via long-range link insertion,” Very Large Scale Integration (VLSI) Systems, IEEE Transactions on, vol. 14, n° 17, p. 693–706, 2006.

[86] C. Bobda, et al, "DyNoC : A dynamic infrastructure for communication in dynamically reconfugurable devices," chez Field Programmable Logic and Applications, 2005. International Conference on, 2005.

[87] G. Palermo, et al "Mapping and Topology Customization Approaches for Application-Specific STNoC Designs," chez Application-specific systems, Architectures and Processors, ASAP. IEEE International Conf. on, 2007.

[88] T. Pionteck, et al , "Applying partial reconfiguration to networks-on chips, " Chez Field Programmable Logic and Applications, 2006. FPL '06. International Conference on, 2006.

[89] M. Hubner, et al, "Scalable application dependent network on chip adaptivity for dynamicalrecon_gurable real-time systems," chez International Conference on Field Programmable Logic and Applications (FPL'04), 2004

[90] .A. Pinto, et al "Efficient synthesis of networks on chip," chez Proceedings of the 21st International Conference on Computer Design (ICCD), 2003.

[91] M. Stensgaard et J. Sparso, "ReNoC : A network-on-chip architecture with reconfigurable topology," chez Second ACM/IEEE International Symposium on Networks-on-Chip, 2008.

[92] G. Palermo et C. Silvano, "PIRATE : A Framework for Power/Performance Exploration of Network-On-Chip Architectures," chez PATMOS-04 : Proc. of International Workshop on Power and Timing Modeling, Optimization and Simulation, 2004.

[93] K. Srinivasan, et al , "ISIS : A genetic algorithm based technique for custom on-chip interconnection network synthesis," chez Proceedings of the 18th International Conference on VLSI Design, 4th International Conference on Embedded Systems Design (VLSID'05), 2005.

[94] K. Srinivasan,et al , "Linear programming based techniques for synthesis of network on chip architectures," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 14, n° 14, p. 407–420, 2006.

[95] J. Xu, et al "A design methodology for application specific networks-on chip," ACM Transactions on Embedded Computing Systems (TECS), vol. 5, n° 12, p. 263–280, 2006.

[96] R. Marculescu et P. Bogdan, "The chip is the network : Toward a science of network on-chip design," chez Foundations and Trends in Electronic Design Automation, 2007.

[97] Z. Guz, et al "Network delays and link capacities in application-specific wormhole NoCs," VLSI Design, vol. 2007, p. 15, 2007.

[98] N. Viswanathan. et al "Performance Analysis of Cluster based 3D Routing Algorithms for NoC". In IEEE Proc. of Recent Advances in Intelligent Computational Systems IEEE Transactions on Parallel and Distributed Systems. 2005 ; 16(9) :853-865.

[99] N. Viswanathan et al "Exploring Optimal Topology and Routing Algorithm for 3D Network on Chip". American Journal of Applied Sciences.2012 ; 9(3) : 300-308.

[100] R Sunkam Ramanujam and Bill Lin ." A layer multiplexed 3D On- Chip Network Architecture". IEEE Embedded Systems Letters.2009 ;1(2)

[101] N.Viswanathan, et al . "Performance comparison of 3D NoC topologies using network calculus". Life Science Journal. 10. 4379-4385.

[102] S.Suboh. et al ."An interconnection architecture for network-on chip systems". Telecommunication Systems. 2008 ;37 :137–144.

[103] Jung-Sheng Fu." Hamiltonicity of the WK Recursive Network with and without Faulty Nodes". IEEE Transactions on Parallel and Distributed Systems. 2005 ; 16(9) :853-865.

[104] K. Srinivasan et al " An automated technique for topology and route generation of application specific on chip interconnection networks", IEEE/ACM Int. Conf. Computer-Aided Design (2005), pp. 231–237.

[105] N. Choudhary, et al "GA based congestion aware topology generation for application specific", Sixth IEEE Int. Symp. Electronic Design, Test and Application (2011), pp. 93–98.

[106] G. Leary and K.S. Chatha, "Automated technique for design of NoC with minimal communication latency", Proc. 7th IEEE/ACM Int. Conf. Hardware/Software Codesign and System Synthesis (2009), pp. 471–480.

[107] U. Y. Ogras and R. Marculescu, ”Energy- and performance-driven NoC communication architecture synthesis using a decomposition approach”, Proc. Design, Automation and Test in Europe, Vol. 1 (2005), pp. 352–357.

[108] A.H KAMECHE "Approches basées sur la BBO pour le Data Clustering et le NoC Mapping"thèse de magistère 2013

[109] F.BOUMAAZA ”Mapping multi-objectifs d'applications intensives sur architecture MPSoC" thèse de magistère 2013

[110] P. K. Sahu and S. Chattopadhyay,"A survey on application mapping strategies for network-on-chip design", Journal of Systems Architecture, 2013

[111] F. M. Darbari et al ,"CGMAP : A new approach to network-on-chip mapping problem," IEICE Electronics Express, vol. 6, 2009.

[112] G. Fen and W. Ning,"Genetic algorithm based mapping and routing approach for network on chip architectures," Chinese Journal of Electron Devices, vol. 19, 2010

[113] M. Tavanpour, et al ,“GBMAP : An evolutionary approach to mapping cores onto a mesh based NoC architecture", International Journal of Computers Communications & Control, vol. 7,2010.

[114] J. Wang, et al "Bandwidth-aware application mapping for NoC-based MPSoCs," J. Comput. Inf. Syst., vol. 7, 2011.

[115] P. K. Sahu, et al ,"Application mapping onto mesh structured network-on-chip using particle swarm optimization," in Proc. IEEE Int. Symp. Very Large Scale Integr.Syst., 2011.

[116] I. Anagnostopoulos et al "Node resource management for DSP applications on 3D network-on-chip architecture," in Proc. IEEE Int. Conf. Digit. Signal Process, 2009.

[117] S. Murali and G. D. Micheli,"Bandwidth constrained mapping of cores onto NoC architectures", vol. 2. Feb. 2004.

[118] K. Siozios, I. et al "A high-level mapping algorithm targeting 3D NoC architectures with multiple vdd," in Proc. IEEE Int. Symp. Very Large Scale Integr. Syst, 2010.

[119] G.Sun "Energy-Aware Run-Time Mapping forHomogeneous NoC"IEEE International Symposium on System-on-Chip - SOC - Tampere, Finland 2010

[120] K. Bhardwaj and R.K. Jena, Energy and bandwidth aware mapping of IPs onto regular NoC architectures using Multi-Objective Genetic Algorithms, SOC 2009, pp. 027-031, 2009

[121] M.Kreutz et al "Energy and Latency Evaluation of NoC Topologies" International Symposium on Circuits and Systems (ISCAS 2005), 23-26 May 2005, Kobe, Japan

[122] S.Kaushik et al "Preprocessing-based Run-time Mapping of Applications on NoC-based MPSoCs" IEEE Computer Society Annual Symposium on VLSI 2011

[123] Yan cang Chen"An Energy-Aware eHeuristic Constructive Mapping Algorithm for Network on Chip"IEEE IEEE 8th International Conference on ASIC (ASICON) - Changsha, Hunan, China 2009

[124] N.Nedjah et al "Customized computer-aided application mapping on NoC infrastructure using multi-objective optimization" Journal of Systems Architecture2010

[125] W. Zhou et al " An application specific NoC mapping for optimized delay " IEEE International Conference on Design and Test of Integrated Systems in Nanoscale Technology, 2006.

[126] S. K. Pradip, M. Kanchan, S. Tapan and C. Santanu,"A Constructive Heuristic for Application Mapping onto Mesh Based Network-on-Chip", Journal of Circuits, Systems and Computers, Vol. 24, No. 08, 2015.

[127] H. Hu and M. Radu,"Energy- and Performance-Aware Mapping for Regular NoC Architectures", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 24, No. 4, pp. 551- 562, Apr. 2005

[128] L. Thiele, I. Bacivarov, W. Haid and K. Huang,"Mapping Applications to Tiled Multiprocessor Embedded Systems", In ACSD, pp. 29–40, 2007

[129] S. Khan, S. Anjum, U. A. Gulzari, M. K. Afzal, T. Umer and F. Ishmanov, "An Efficient Algorithm for Mapping Real Time Embedded Applications on NoC Architecture," in IEEE Access, vol. 6, pp. 2017

[130] S. Tosun "New heuristic algorithms for energy aware application mapping and routing on mesh-based NoCs"Journal of Systems Architecture2011 Vol. 57 ; Iss. 1 2011

[131] E Carvalho "Heuristics for Dynamic Task Mapping in NoC-based Heterogeneous MP-SoCs"18th IEEE/IFIP International Workshop on Rapid System 2007

[132] J. Fang‘’DI_GA : A Heuristic Mapping Algorithm for Heterogeneous Network-on-Chip” IOP Conf. Series : Materials Science and Engineering CHINA,2019

[133] C. ChiHsiang "Application mapping onto mesh-based network-on-chip using constructive heuristic algorithms" The Journal of Supercomputing / 11 Vol. 72 ; Iss. 11 2016

[134] C.Q Xu "Unified multi-objective mapping for network-on-chip using genetic-based hyper-heuristic algorithms"IET Computers & Digital Techniques Volume 12, Issue 4, July 2018, p. 158 – 166

[135] A. Alagarsamy, et al "A Self-Adaptive Mapping Approach for Network on Chip With Low Power Consumption," in IEEE Access, vol. 7, pp. 84066-84081, 2019, doi : 10.1109/ACCESS.2019.2925381

[136] I. Akturk and O. Ozturk, "ILP-Based Communication Reduction for Heterogeneous 3D Network-on-Chips," 2013 21st Euromicro International Conference on Parallel, Distributed, and Network-Based Processing, Belfast, 2013, pp. 514-518, doi : 10.1109/PDP.2013.83

[137] Buckler, M. et al. (2013). Low-power Networks-on-Chip : Progress and remaining challenges. IEEE International Symposium on Low Power Electronics and Design (ISLPED)

[138] Abbas El Dor "Perfectionnement des algorithmes d'optimisation par essaim particulaire : applications en segmentation d'images et en électronique" thèse de doctorat en informatique 2013

[139] Nadia Smairi "Optimisation par essaim particulaire : adaptation de tribes à l'optimisation multiobjectif" thèse de doctorat en informatique 2013

[140] Maria Zemzami et al "Application d'un modèle parallèle de la méthode PSO au problème de transport d'électricité" ISTE OpenScience ISTE Science Publishing, London, UK 2016

[141] ZIELINSKI K., LAUR R., Stopping Criteria for Differential Evolution in Constrained Single-Objective Optimization. In : Advanced in Differential Evolution, the series Studies in Computational Intelligence, vol. 143, pp. 111-138 Springer, Berlin Heidelberg. 2008

[142] Reyes-Sierra M, Coello Coello C a. (2006) Multi-Objective Particle Swarm Optimizers : A Survey of the State-of-the-Art. Int J Comput Intell Res 2 :287–308. doi : 10.5019/j.ijcir.2006.68

Annexe A : Benchmarks aléatoires

1 L'application 40 IPs

Cette application contient 40 IPs et 40 liens, le tableau suivant représente APG de l'application.

IPx	IPy	Coût
IP1	IP2	64
IP2	IP5	128
IP2	IP28	4
IP3	IP12	16
IP4	IP11	70
IP4	IP15	128
IP5	IP20	64
IP6	IP11	128
IP6	IP21	64
IP7	IP12	16
IP8	IP15	128
IP9	IP13	200
IP10	IP7	128
IP10	IP30	20
IP13	IP3	200
IP14	IP17	100
IP16	IP23	70
IP16	IP26	8
IP17	IP35	16
IP18	IP37	50
IP19	IP33	4
IP20	IP25	64
IP21	IP18	128
IP22	IP2	128
IP23	IP1	8
IP24	IP27	128
IP25	IP8	96
IP28	IP27	16
IP28	IP14	16
IP28	IP19	64
IP29	IP32	20
IP30	IP22	96
IP35	IP26	100
IP36	IP34	200
IP36	IP38	150
IP37	IP29	16
IP38	IP30	150
IP39	IP31	50
IP40	IP32	20
IP40	IP33	64

Application 80IPs

Cette application contient 80 IPs et 81 liens le tableau suivant représente APG de l'application.

IPx	IPy	Coût
IP1	IP2	64
IP2	IP5	128
IP2	IP28	4
IP3	IP12	16
IP4	IP11	70
IP4	IP15	128
IP5	IP20	64
IP6	IP11	128
IP6	IP21	64
IP7	IP12	16
IP8	IP15	128
IP9	IP13	200
IP10	IP7	128
IP10	IP30	20
IP13	IP3	200
IP14	IP17	100
IP16	IP23	70
IP16	IP26	8
IP17	IP35	16
IP18	IP37	50
IP19	IP33	4
IP20	IP25	64
IP21	IP18	128
IP22	IP2	128
IP23	IP1	8
IP24	IP27	128
IP25	IP8	96
IP28	IP27	16
IP28	IP14	16
IP28	IP19	64
IP29	IP32	20
IP30	IP22	96
IP35	IP26	100
IP36	IP34	200
IP36	IP38	150
IP37	IP29	16

IPx	IPy	Coût
IP38	IP30	150
IP39	IP31	50
IP40	IP32	20
IP40	IP33	64
IP40	IP39	2
IP40	IP80	2
IP41	IP42	64
IP42	IP68	4
IP42	IP45	128
IP43	IP52	16
IP44	IP51	70
IP44	IP55	128
IP45	IP60	64
IP46	IP51	128
IP46	IP61	64
IP47	IP52	16
IP48	IP55	128
IP49	IP53	200
IP50	IP70	20
IP50	IP47	128
IP53	IP43	200
IP54	IP57	100
IP56	IP66	8
IP56	IP63	70
IP57	IP75	16
IP58	IP77	50
IP59	IP73	4
IP60	IP65	64
IP61	IP58	128
IP62	IP42	128
IP63	IP41	8
IP64	IP67	128
IP65	IP48	96
IP68	IP54	16
IP68	IP59	64
IP68	IP67	16

IP68	IP54	16
IP68	IP59	64
IP68	IP67	16
IP69	IP72	20
IP70	IP62	96
IP75	IP66	100
IP76	IP74	200
IP76	IP78	150
IP77	IP69	16
IP78	IP70	150
IP79	IP71	50
IP80	IP79	2
IP80	IP72	20
IP80	IP73	64

Annexe B : Les résultats des archives

Les archives de MOPSO proposée

Les archives suivantes représentent les résultats obtenus pour les tests font sur la méthode MOPSO proposée avec un coût de TSV (α) égale à 0.8.

PIP

	Plan 1				Plan 2			
Tuile	1	2	3	4	5	6	7	8
IP	IP2	IP3	IP6	IP4	IP1	IP8	IP5	IP7
TSV	1	0	1	1	1	0	1	1

(576 ; 3 TSVs 75%)

	Plan 1				Plan 2			
Tuile	1	2	3	4	5	6	7	8
IP	IP8	IP3	IP6	IP2	IP7	IP4	IP5	IP1
TSV	1	1	1	1	1	1	1	1

(563.2 ; 4 TSVs 100%)

	Plan 1				Plan 2			
Tuile	1	2	3	4	5	6	7	8
IP	IP4	IP7	IP3	IP8	IP1	IP5	IP2	IP6
TSV	0	0	1	0	0	0	1	0

(742.4 ; 1 TSV 25%)

	Plan 1				Plan 2			
Tuile	1	2	3	4	5	6	7	8
IP	IP2	IP7	IP3	IP4	IP1	IP8	IP5	IP6
TSV	1	1	0	0	1	1	0	0

(588.8 ; 2 TSVs 50%)

MWD

	Plan1						Plan2					
Tuile	1	2	3	4	5	6	7	8	9	10	11	12
IP	IP3	IP2	IP8	IP4	IP1	IP5	IP11	IP9	IP10	IP12	IP6	IP7
TSV	0	1	0	0	0	0	0	1	0	0	0	0

(1369.6 ; 1 TSV 16.66%)

	Plan1						Plan2					
Tuile	1	2	3	4	5	6	7	8	9	10	11	12
IP	IP11	IP9	IP8	IP12	IP4	IP5	IP3	IP7	IP10	IP2	IP6	IP1
TSV	0	1	0	0	0	1	0	1	0	0	0	1

(1235.2 ; 2 TSVs 33.33%)

	Plan1						Plan2					
Tuile	1	2	3	4	5	6	7	8	9	10	11	12
IP	IP11	IP9	IP8	IP12	IP4	IP5	IP7	IP10	IP3	IP2	IP6	IP1
TSV	1	1	0	0	0	1	1	1	0	0	0	1

(1203.2 ; 3 TSVs 50%)

	Plan1						Plan2					
Tuile	1	2	3	4	5	6	7	8	9	10	11	12
IP	IP4	IP5	IP8	IP6	IP1	IP12	IP7	IP10	IP9	IP2	IP3	IP11
TSV	0	0	1	1	1	1	0	0	1	1	1	1

(1196.8 ; 4 TSVs 66.66%)

	Plan1						Plan2					
Tuile	1	2	3	4	5	6	7	8	9	10	11	12
IP	IP4	IP5	IP8	IP3	IP1	IP12	IP7	IP10	IP9	IP6	IP2	IP11
TSV	1	0	1	1	1	1	1	0	1	1	1	1

(1177.6 ; 5 TSVs 83.33%)

MPEG

	Plan1						Plan2					
Tuile	1	2	3	4	5	6	7	8	9	10	11	12
IP	IP6	IP3	IP2	IP4	IP5	IP1	IP8	IP7	IP12	IP11	IP10	IP9
TSV	0	0	0	0	1	0	0	0	0	0	1	0

(3483.5 ; 1 TSV 16.66%)

	Plan1						Plan2					
Tuile	1	2	3	4	5	6	7	8	9	10	11	12
IP	IP6	IP12	IP3	IP4	IP5	IP1	IP8	IP7	IP11	IP2	IP10	IP9
TSV	0	1	0	0	1	0	0	1	0	0	1	0

(3342.4 ; 2 TSVs 33.33%)

	Plan1						Plan2					
Tuile	1	2	3	4	5	6	7	8	9	10	11	12
IP	IP1	IP5	IP4	IP9	IP12	IP6	IP2	IP10	IP3	IP11	IP7	IP8
TSV	0	1	1	0	1	0	0	1	1	0	1	0

(3322.5 ; 3 TSVs 50%)

VOPD

	Plan1								Plan2							
Tuile	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
IP	IP6	IP5	IP7	IP1	IP8	IP4	IP3	IP2	IP11	IP10	IP13	IP14	IP12	IP9	IP16	IP15
TSV	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0

(4409.8 ; 1 TSV 12.5%)

	Plan1								Plan2							
Tuile	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
IP	IP12	IP13	IP14	IP6	IP11	IP10	IP9	IP8	IP16	IP15	IP7	IP5	IP1	IP2	IP3	IP4
TSV	0	1	0	1	0	0	0	0	0	1	0	1	0	0	0	0

(4124 ; 2 TSVs 25%)

	Plan1								Plan2							
Tuile	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
IP	IP9	IP8	IP6	IP3	IP12	IP16	IP15	IP2	IP10	IP7	IP5	IP4	IP11	IP13	IP14	IP1
TSV	1	0	1	1	0	0	0	0	1	0	1	1	0	0	0	0

(4091.8 ; 3 TSVs 37.5)

	Plan1								Plan2							
Tuile	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
IP	IP9	IP8	IP6	IP3	IP12	IP16	IP15	IP2	IP10	IP14	IP5	IP4	IP11	IP13	IP7	IP1
TSV	1	0	1	1	0	0	0	1	1	0	1	1	0	0	0	1

(3951.8 ; 4 TSVs 50%)

	Plan1								Plan2							
Tuile	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
IP	IP9	IP8	IP6	IP3	IP12	IP16	IP15	IP2	IP10	IP14	IP5	IP4	IP11	IP13	IP7	IP1
TSV	1	0	1	1	1	0	0	1	1	0	1	1	1	0	0	1

(3887.8 ; 5 TSVs 62.5%)

	Plan1								Plan2							
Tuile	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
IP	IP11	IP9	IP8	IP7	IP13	IP14	IP2	IP3	IP12	IP10	IP6	IP5	IP16	IP15	IP1	IP4
TSV	1	1	1	1	0	1	1	1	1	1	1	1	0	1	1	1

(3868.6 ; 7 TSVs 87.5%)

	Plan1								Plan2							
Tuile	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
IP	IP15	IP11	IP9	IP8	IP1	IP2	IP3	IP7	IP16	IP12	IP10	IP6	IP14	IP13	IP4	IP5
TSV	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

(3841.6 ; 8 TSVs 100%)

Les archives de la méthode exacte

Les archives suivantes représentent les résultats obtenus pour les tests font sur la méthode exacte avec un coût de TSV (α) égale à 0.8.

PIP

Nombre de solution 12

Le cout minimal est 742.400000

PLAN 1	PLAN 2
6 8	2 3
5 7	1 4

Nombre de TSVs est 1

5	1

Le cout minimal est 742.400000

PLAN 1	PLAN 2
6 8	3 5
4 7	2 1

Nombre de TSVs est 1

6	3

Le cout minimal est 563.200000

PLAN 1	PLAN 2
6 1	5 2
7 4	8 3

Nombre de TSVs est 4

6	5
1	2
7	8
4	3

Le cout minimal est 563.200000

PLAN 1	PLAN 2
6 8	5 7
2 3	1 4

Nombre de TSVs est 4

6	5
8	7
2	1
3	5

Le cout minimal est 588.800000

PLAN 1	PLAN 2
6 8	4 7
5 1	3 2

Nombre de TSVs est 2

8	7
1	2

Le temps d'execution= 11.100000

MWD

Nombre de solution 34

Le cout minimal est 1139.200000

PLAN 1	PLAN 2
1 2 3	5 6 7
4 11 12	8 9 10

Nombre de TSVs est 3

1	5
2	6
11	9

Le cout minimal est 1177.600000

PLAN 1	PLAN 2
1 2 3	7 6 12
5 8 4	10 9 11

Nombre de TSVs est 2

2	6
8	9

Le cout minimal est 1139.200000

PLAN 1	PLAN 2
4 11 12	8 9 10
1 2 3	5 6 7

Nombre de TSVs est 3

11	9
1	5
2	6

Le cout minimal est 1369.600000

PLAN 1	PLAN 2
3 2 8	11 9 10
4 1 5	12 6 7

Nombre de TSVs est 1

2	9

Le cout minimal est 576.000000

PLAN 1	PLAN 2
6 1	5 2
7 8	4 3

Nombre de TSVs est 3

6	5
1	2
7	4

Le cout minimal est 1113.600000

PLAN 1	PLAN 2
2 1 4	3 5 8
6 12 11	7 10 9

Nombre de TSVs est 4

2	3
1	5
6	7
11	9

Le cout minimal est 1113.600000

PLAN 1	PLAN 2
2 1 12	3 5 8
6 4 11	7 10 9

Nombre de TSVs est 4

2	3
1	5
6	7
11	9

Le cout minimal est 1113.600000

PLAN 1	PLAN 2
3 5 8	2 1 4
7 10 9	6 12 11

Nombre de TSVs est 4

3	2
5	1
7	6
9	11

Le cout minimal est 1113.600000

PLAN 1	PLAN 2
3 5 8	2 1 12
7 10 9	6 4 11

Nombre de TSVs est 4

3	2
5	1
8	12
9	11

Le cout minimal est 1369.600000

PLAN 1	PLAN 2
12 6 7	4 1 5
11 9 10	3 2 8

Nombre de TSVs est 1

9	2

Le cout minimal est 1113.600000

PLAN 1 PLAN 2

4 11 12 8 9 10

1 2 6 5 3 7

Nombre de TSVs est 4

11 9

1 5

2 3

6 7

Le cout minimal est 1113.600000

PLAN 1 PLAN 2

5 3 7 1 2 6

8 9 10 4 11 12

Nombre de TSVs est 4

5 1

3 2

6 9

11 7

Le cout minimal est 1139.200000

PLAN 1 PLAN 2

5 6 7 1 2 3

8 9 10 4 11 12

Nombre de TSVs est 3

5 1

6 2

9 11

Le cout minimal est 1177.600000

PLAN 1 PLAN 2

5 8 4 10 9 11

1 2 3 7 6 12

Nombre de TSVs est 2

8 9

2 6

Le cout minimal est 1113.600000

PLAN 1 PLAN 2

6 4 11 7 10 9

2 1 12 3 5 8

Nombre de TSVs est 4

6 7

11 9

2 3

1 5

Le cout minimal est 1113.600000

PLAN 1 PLAN 2

6 12 11 7 10 9

2 1 4 3 5 8

Nombre de TSVs est 4

6 7

11 9

2 3

1 5

Le cout minimal est 1369.600000

PLAN 1 PLAN 2

4 1 5 12 6 7

3 2 8 11 9 10

Nombre de TSVs est 1

2 9

Le cout minimal est 1177.600000

PLAN 1 PLAN 2

7 6 12 1 2 3

10 9 11 5 8 4

Nombre de TSVs est 2

6 2

9 8

Le cout minimal est 1113.600000

PLAN 1 PLAN 2

7 10 9 6 12 11

3 5 8 2 1 4

Nombre de TSVs est 4

7 6

9 11

3 2

5 1

Le cout minimal est 1113.600000

PLAN 1 PLAN 2

7 10 9 6 4 11

3 5 8 2 1 12

Nombre de TSVs est 4

7 6

9 11

3 2

5 1

Le cout minimal est 1139.200000

PLAN 1 PLAN 2

8 9 10 4 11 12

5 6 7 1 2 3

Nombre de TSVs est 3

9 11

5 1

6 2

Le cout minimal est 1113.600000

PLAN 1 PLAN 2

8 9 10 4 11 12

5 3 7 1 2 6

Nombre de TSVs est 4

9 11

5 1

3 2

7 6

Le cout minimal est 1113.600000

PLAN 1 PLAN 2

8 5 3 12 1 2

9 10 7 11 4 6

Nombre de TSVs est 4

5 1

3 2

9 11

7 6

Le cout minimal est 1113.600000

PLAN 1 PLAN 2

9 10 7 11 12 6

8 5 3 12 1 4

Nombre de TSVs est 4

9 11

7 6

5 1

3 2

Le cout minimal est 1177.600000

PLAN 1 PLAN 2

10 9 11 5 8 4

7 6 12 1 2 3

Nombre de TSVs est 2

9 8

6 2

Le cout minimal est 1113.600000

PLAN 1 PLAN 2

9 10 7 11 4 6

8 5 3 12 1 2

Nombre de TSVs est 4

9 11

7 6

5 1

3 2

Le cout minimal est 1113.600000

PLAN 1 PLAN 2

9 8 5 11 4 1

10 7 3 12 6 2

Nombre de TSVs est 4

9 11

5 1

7 6

3 2

Le cout minimal est 1369.600000

PLAN 1 PLAN 2

11 9 10 3 2 8

12 6 7 4 1 5

Nombre de TSVs est 1

9 2

Le cout minimal est 1113.600000

PLAN 1 PLAN 2

10 9 8 12 11 4

7 3 5 6 2 1

Nombre de TSVs est 4

9 11

7 6

3 2

5 1

Le cout minimal est 1113.600000

PLAN 1 PLAN 2

11 12 6 9 10 7

4 1 2 8 5 3

Nombre de TSVs est 4

11 9

6 7

1 5

2 3

Le cout minimal est 1113.600000

PLAN 1 PLAN 2

11 4 6 9 10 7

12 1 2 8 5 3

Nombre de TSVs est 4

11 9

6 7

1 5

2 3

Le cout minimal est 1113.600000

PLAN 1 PLAN 2

11 4 6 9 8 5

12 6 2 10 7 3

Nombre de TSVs est 4

11 9

1 5

6 7

2 3

Le cout minimal est 1113.600000

PLAN 1 PLAN 2

12 1 2 8 5 3

11 4 6 9 10 7

Nombre de TSVs est 4

1 5

2 3

11 9

6 7

Le cout minimal est 1113.600000

PLAN 1 PLAN 2

12 6 2 10 7 3

11 4 1 9 8 5

Nombre de TSVs est 4

6 7

2 3

11 9

1 5

Le cout minimal est 1113.600000

PLAN 1	PLAN 2
12 11 4	10 9 8
6 2 1	7 3 5

Nombre de TSVs est 4

11	9
6	7
2	3
1	5

le temps d'execution= 864000.900000

MPEG

Nombre de solution 12

Le cout minimal est 3483.500000

PLAN 1	PLAN 2
1 5 4	2 10 11
9 3 6	12 7 8

Nombre de TSVs est 1

5	10

Le cout minimal est 3342.40000

PLAN 1	PLAN 2
6 12 3	8 7 11
4 5 1	2 10 9

Nombre de TSVs est 2

12	7
5	10

Le cout minimal est 3322.500000

PLAN 1	PLAN 2
1 5 4	9 10 3
2 12 6	8 7 11

Nombre de TSVs est 3

5	10
4	3
12	7

Le cout minimal est 3483.500000

PLAN 1	PLAN 2
2 10 11	1 5 4
12 7 8	9 3 6

Nombre de TSVs est 1

10	5

Le cout minimal est 3342.40000

PLAN 1	PLAN 2
4 5 1	2 10 9
6 12 3	8 7 11

Nombre de TSVs est 2

5	10
12	7

Le cout minimal est 3322.500000

PLAN 1	PLAN 2
2 12 6	8 7 11
1 5 4	9 10 3

Nombre de TSVs est 3

12	7
5	10
4	3

Le cout minimal est 3342.40000

PLAN 1	PLAN 2
2 10 9	4 5 1
8 7 11	6 12 3

Nombre de TSVs est 2

10	5
7	12

Le cout minimal est 3342.40000

PLAN 1	PLAN 2
8 7 11	6 12 3
2 10 9	4 5 1

Nombre de TSVs est 2

7	12
10	5

Le cout minimal est 3322.500000

PLAN 1	PLAN 2
8 7 11	2 12 6
9 10 3	1 5 4

Nombre de TSVs est 3

7	12
10	5
3	4

Le cout minimal est 3483.500000

PLAN 1	PLAN 2
9 3 6	12 7 8
1 5 4	2 10 11

Nombre de TSVs est 1

5	10

Le cout minimal est 3322.500000

PLAN 1	PLAN 2
9 10 3	1 5 4
8 7 11	2 12 6

Nombre de TSVs est 3

10	5
3	4
7	12

Le cout minimal est 3483.500000

PLAN 1	PLAN 2
12 7 8	9 3 6
2 10 11	1 5 4

Nombre de TSVs est 1

10	5

Le temps d'execution= 699200.20000

VOPD

Nombre de solution 12

Le cout minimal est 3700.200000

PLAN 1	PLAN 2
1 4 5 8	2 3 6 7
14 16 11 10	13 15 12 9

Nombre de TSVs est 7

1	2
4	3
5	6
8	7
14	13
11	12
10	9

Le cout minimal est 3700.200000

PLAN 1	PLAN 2
2 3 6 7	1 4 5 8
13 15 12 9	14 16 11 10

Nombre de TSVs est 7

2	1
3	4
6	5
7	8
13	14
12	11
9	10

Le cout minimal est 3881.000000

PLAN 1	PLAN 2
2 1 11 15	3 4 5 16
10 9 12 14	8 7 6 13

Nombre de TSVs est 3

2	3
10	8
14	13

Le cout minimal est 3802.000000

PLAN 1	PLAN 2
2 5 6 7	3 4 12 9
1 16 14 8	15 11 13 10

Nombre de TSVs est 5

2	3
5	4
6	12
14	13
8	10

Le cout minimal est 4290,800000

PLAN 1	PLAN 2
12 16 13 14	5 6 4 3
11 9 8 15	7 10 1 2

Nombre de TSVs est 1

9	10

Le cout minimal est 3937.400000

PLAN 1	PLAN 2
3 7 10 9	2 13 11 12
4 5 6 8	1 14 15 16

Nombre de TSVs est 2

3	2
10	11

Le cout minimal est 3881.000000

PLAN 1	PLAN 2
3 4 5 16	2 1 11 15
8 7 6 13	10 9 12 14

Nombre de TSVs est 3

3	2
8	10
13	14

Le cout minimal est 3846.200000

PLAN 1	PLAN 2
2 5 6 7	3 4 12 9
1 16 11 8	14 13 15 10

Nombre de TSVs est 4

2	3
5	4
6	12
8	10

Le cout minimal est 3846.200000

PLAN 1	PLAN 2
3 4 12 9	2 5 6 7
14 13 15 10	1 16 11 8

Nombre de TSVs est 4

3	2
4	5
12	6
10	8

Le cout minimal est 3802.000000

PLAN 1	PLAN 2
3 4 12 9	2 5 6 7
15 11 13 10	1 16 14 8

Nombre de TSVs est 5

3	2
4	5
12	6
13	14
10	8

Le cout minimal est 4290,800000

PLAN 1	PLAN 2
11 9 8 15	7 10 1 2
12 16 13 14	5 6 4 3

Nombre de TSVs est 1

9	10

Le cout minimal est 3937.400000

PLAN 1	PLAN 2
4 5 6 8	1 14 15 16
3 7 10 9	2 13 11 12

Nombre de TSVs est 2

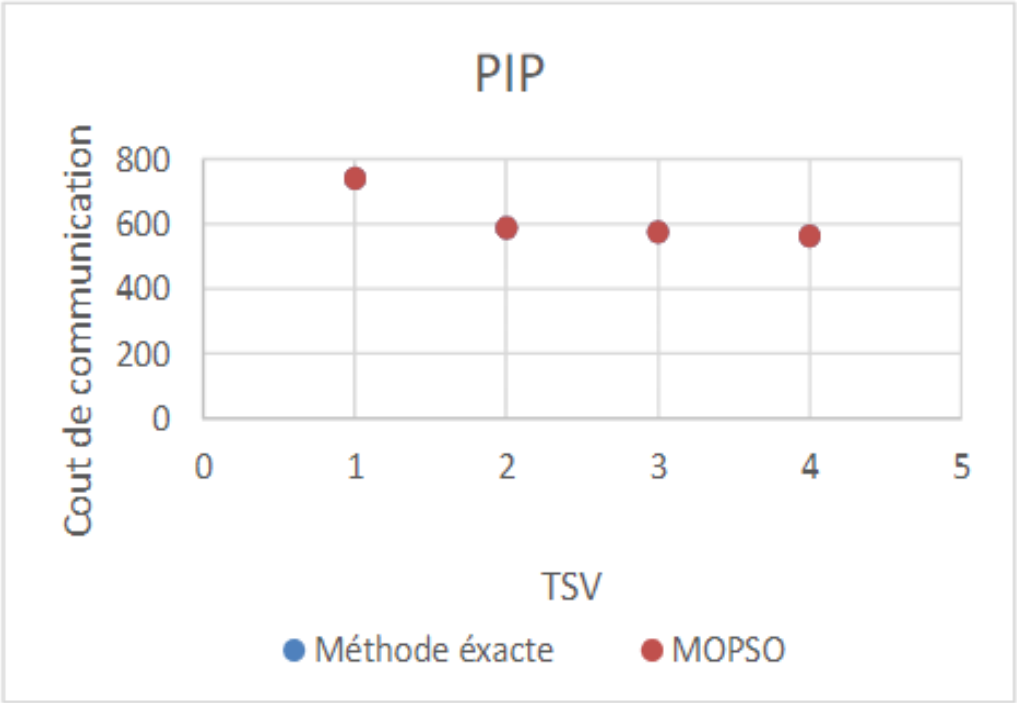
3	2
10	11

Le temps d'execution= 1489678.000000

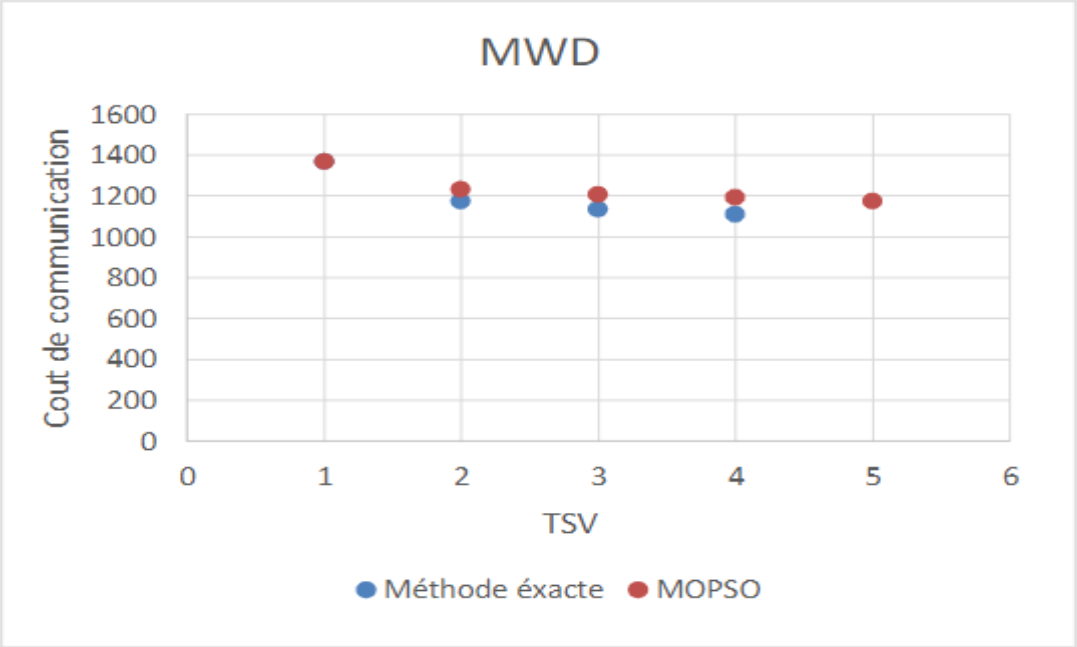
Annexe C : Front Pareto

Les graphes suivants représentent une comparaison entre les résultats obtenus dans la méthode exacte et MOPSO proposée et données en annexe B.

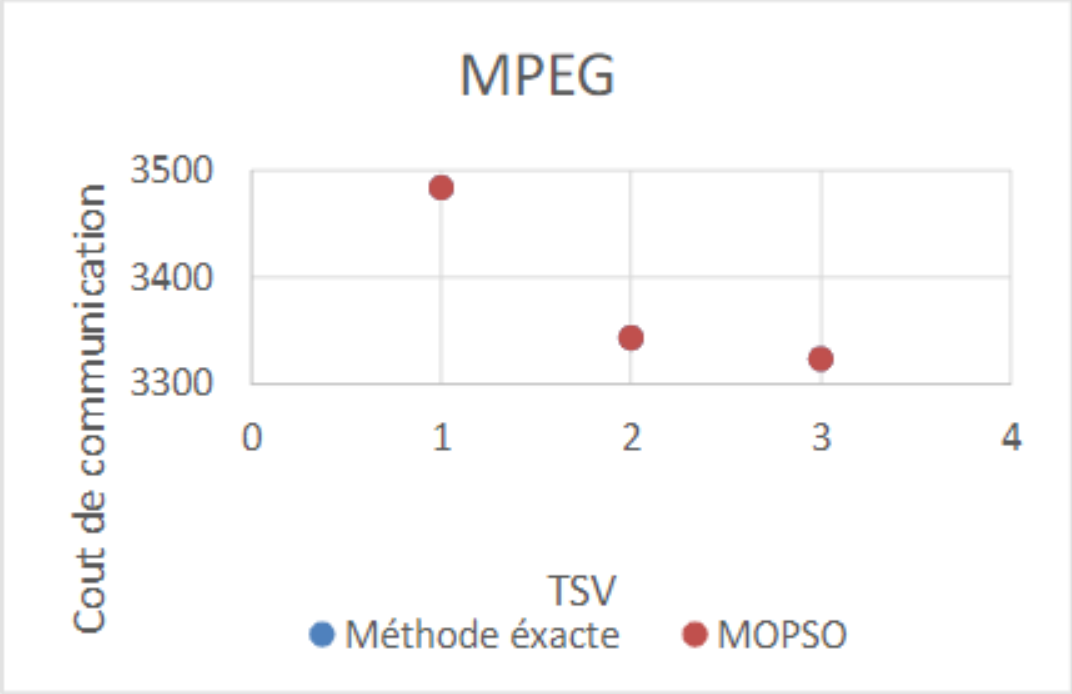
PIP



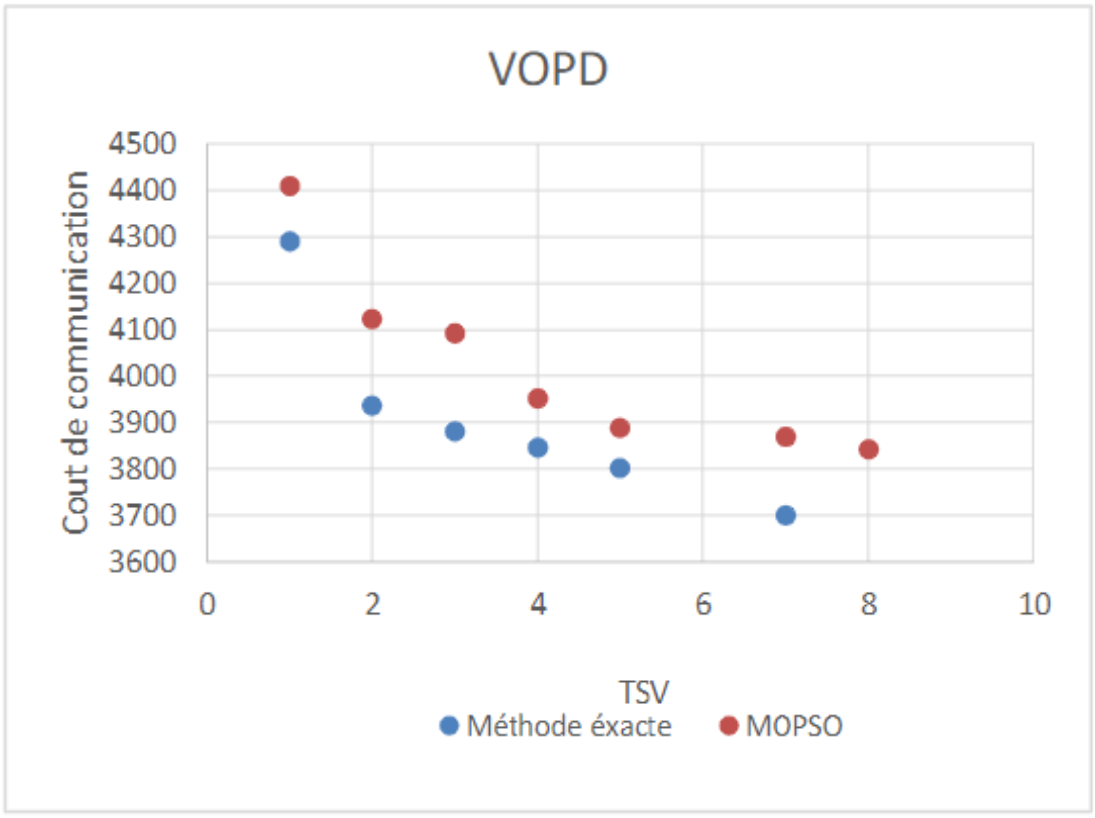
MWD


MWD
Cout de communication
1600
1400
1200
1000
800
600
400
200
0
0
1
2
3
4
5
6
TSV
Méthode éxacte
MOPSO

MPEG


MPEG
Cout de communication
3500
3400
3300
0
1
2
3
4
TSV
Méthode éxacte
MOPSO

VOPD


VOPD
Cout de communication
4500
4400
4300
4200
4100
4000
3900
3800
3700
3600
0
2
4
6
8
10
TSV
Méthode éxacte
MOPSO