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 :

Mapping multi-objectif dans les réseaux sur puce 3D

Réalisé par :

Elbey Abdelmoumene

Kada Issam

Soutenu devant un jury constitué de :

- Mr Riali I. Président
- Mr Kameche A. Examinateur
- Dr Toubaline N. Promotrice

Année universitaire : 2022/2023

Résumé

Le réseau sur puce est un nouveau concept d'interconnexions dans les systèmes sur puce. Cette architecture facilite l'intégration de plusieurs composants. Différents types d'interconnexion classiques ont été proposés pour la communication de ces composants (point à point, bus partagé et bus hiérarchique). Cependant, ces interconnexions rencontrent des limites lorsqu'un nombre important de composantes sont connectés dans le système. Le réseau sur puce (Network On Chip) est un paradigme pour résoudre ces problèmes et offre une amélioration des performances.

La conception d'un NoC nécessite plusieurs phases. On s'intéresse dans le cadre de notre projet, la phase du choix de topologie 3D du réseau et la phase de mapping qui consiste au placement des composants communicant travers le réseau. Traiter en parallèle ces deux phases permet d'optimiser plusieurs objectifs, le coût de communication et la surface, pour cela nous sommes orientés vers les méthodes d'optimisation multiobjectif.

La solution proposé est un outil d'aide qui permet au concepteur de NoC d'optimiser les performances de réseau (surface et coût de communication) pour une application spécifique.

Mots clés : Réseau sur puce, mapping, topologie, technologie 3D et optimisation multiobjectif.

Abstract

Network-on-a-chip is a new concept in system-on-a-chip interconnections. This architecture facilitates the integration of multiple components. Various conventional types of interconnection have been proposed for communicating these components (point-to-point, shared bus and hierarchical bus). However, these interconnections are limited when a large number of components are connected in the system. Network On Chip (NOC) is a paradigm for solving these problems, and offers improved performance.

The design of a NoC requires several phases. In the context of our project, we are interested in the network topology selection phase and the mapping phase, which consists in placing the components communicating across the network. Treat these two phases in parallel allows to optimize several objectives, the cost of communication and the surface, for this we are oriented towards methods of multi-objective optimization.

The proposed solution is a support tool that enables the NoC designer to optimize network performance (area and communication cost) for a specific application.

Keywords : Network-on-chip, mapping, topology, 3D technology and multi-objective optimization.

ملخص

الشبكة على شريحة هي مفهوم جديد للترابط في الأنظمة الذكية. يسهل هذا الهيكل تكامل العديد من المكونات. وقد اقترحت أنواع مختلفة من الترابط التقليدي لإيصال هذه المكونات (من نقطة إلى نقطة، حافلة مشتركة وحافلة هرمية). ومع ذلك، فإن هذه الروابط تواجه حدودا عندما يتم توصيل عدد كبير من المكونات في النظام. تعد الشبكة على شريحة نموذجا لحل هده المشاكل ولتقديم أداء أحسن.

يتطلب تصميم الشبكة على شريحة عدة مراحل. في إطار مشروعنا، نحن مهتمون بمرحلة اختيار طوبولوجيا الشبكة ومرحلة رسم الخرائط التي تتضمن وضع المكونات التي تتواصل فيما بينها من خلال الشبكة. معالجة هاتين المرحلتين بالتوازي يسمح بتحسين العديد من الأهداف، تكلفة الاتصال و المساحة، لهاذا فإننا موجهون نحو طرق التحسين متعددة الأهداف.

الحل المقترح هو أداة دعم تسمح لمصمم الشبكة على شريحة بتحسين أداءها (مساحة السطح و تكلفة الاتصال) من أجل تطبيق معين.

الكلمات المفتاحية الشبكات على شريحة ورسم الخرائط و توبولوجيا والتكنولوجيا ثلاثية الأبعاد وتحسين متعدد الأهداف.

Remerciement

À la fin de ce mémoire, nous tenons à exprimer tous nos remerciements et notre profonde gratitude. Tout d'abord, nous remercions sincèrement ***Dieu*** *de nous avoir donné la force et le courage nécessaires pour mener à bien ce travail modeste.*

Nous saisissons cette occasion pour exprimer notre reconnaissance profonde envers notre promotrice, Dr. ***TOUBALINE Nesrine****, pour son soutien et ses efforts tout au long de cette étude.*

Nous souhaitons également exprimer nos remerciements à tous ***les enseignants du département d'informatique de l'université Saad Dahleb Blida 1.***

Enfin, nous tenons à remercier toutes les personnes qui nous ont aidés de près ou de loin dans l'élaboration de ce travail.

Nous sommes reconnaissants envers tous ceux qui ont contribué à notre parcours académique et qui ont apporté leur soutien tout au long de ce mémoire.

Nos remerciements leur sont adressés avec une profonde gratitude.

Dédicace

Avec une affection immense, je souhaite dédier ce travail modeste :

À celle qui m'a donné naissance, qui m'a entouré de tendresse et d'espoir avec un amour infini, Ma chère mère bien-aimée.

À mon soutien inébranlable dans la vie, celui qui a toujours été présent à mes côtés et m'a soutenu dès mes premiers pas, Mon cher et bienveillant père. À ma famille.

À mon binôme, Abdelmoumen, qui a partagé ce parcours avec moi.

Issam

Dédicace

C'est avec un amour immense que je dédie ce modeste travail :

À celle qui m'a donné la vie, qui m'a comblé de tendresse et d'espoir, une source inépuisable d'amour, Ma mère bien-aimée.

À mon soutien inconditionnel dans la vie, celui qui a toujours été à mes côtés et qui m'a soutenu dès le début, Mon père chéri.

À ceux qui m'ont entouré de leur affection et de leur bienveillance, ceux qui m'ont toujours aimé.

Et enfin, à mon binôme, Issam, qui a partagé ce parcours avec moi.

C'est à toutes ces personnes que je dédie ce travail avec une profonde gratitude et un amour infini.

Abdelmoumene

Table des matières

Table des figures

Liste des tableaux

Liste des acronymes

A :

ACA : Andean Condor Algorithm.

ACO : Ant Colony Optimization.

APG : Application Graph.

ARG : Architecture Graph.

ASIC : Application Specific Integrated Circuit.

B :

BA : Bat Algorithm.

C :

CAO : Conception Assistée par Ordinateur.

CHA : Genetic Hyperheuristique Algorithm.

CHMAP : Constructive Heuristique Mapping.

CPU : Central Processing Unit.

CSO : Chicken Swarm Optimization.

G :

GA : Genetic Algorithm.

I :

IC : Integrated Circuit.

ILP : Integer Linear Programmation.

IP : Intellectual Property.

K :

KLMap : Kerniran_Lin Mapping.

M :

MILP : Mixed Integer Linear Programmation.

MMT : Modified Mesh Topology.

MPSOC : Multiprocessor System On Chip.

MSP : Modified Shortest Path.

N :

NI : Network Interface.

NMAP : Network Mapper.

NOC : Network On Chip.

NP : Non determinist Polynomial.

P :

PMAP : Physical Mapping.

PSO : Particle Swarm Optimization.

R :

RNT : Recursive Network Topology.

S :

SFOA : SailFish Optimization Algorithm.

SOC : System On Chip.

T :

TSV : Throught Silicon Via.

TTOT : Tabou Topology Optimization.

Introduction générale

L'importante évolution de la technologie de fabrication des systèmes électroniques lors de ces dernières années a permis l'apparition d'une nouvelle génération de systèmes appelée les systèmes sur puces (SoC : Système On Chip). Le système sur puce (SoC) est un paradigme pour la conception des circuits intégrés d'aujourd'hui (IC), composé d'un grand nombre de blocs IP (Intellectual Property) sur le même substrat de silicium.

Les interconnexions classiques (point à point, bus partagé, bus hiérarchique) sont les structures d'interconnexion dominantes pour les systèmes SoC. Cependant ils trouvent leurs limites face à la consommation d'énergie, la bande passante et la surface lorsqu'un nombre important de composants sont connectés. C'est dans ce contexte que les réseaux sur puce (NoC : Network On Chip) sont apparus.

Le NoC est un paradigme d'interconnexion inspiré des réseaux informatique classiques, permettant principalement un parallélisme de communication au niveau des circuits.

Les critères d'un réseau sur puce en termes de la bande passante, de surface, coût de communication et de consommation d'énergie sont fortement dépendantes de l'emplacement des composant (mapping) et de choix de la topologie.

Le problème du mapping est un problème d'optimisation combinatoire NP difficile. Placer N IPs dans M nœuds du réseau ($N \leq M$) implique $M ! / (M-N) !$ arrangements d'IPs possibles dans les nœuds du NoC. Il est nécessaire de développer une heuristique ou métaheuristique afin de déterminer une solution optimale ou pré-optimale dans un temps CPU raisonnable.

Les applications modernes nécessitent l'intégration de plus de cœurs d'IP afin de répondre aux différents exigences, donc des réseaux de grande tailles doivent être mis en place pour communiquer ces différents cœurs. Cependant le fait d'agrandir la taille de réseau a montré des mauvaises conséquences. L'apparition de la technologie des circuits intégrés à 3 dimensions (3D IC) a encouragé les chercheurs à implémenter des réseaux de grand taille en utilisant cette nouvelle technologie pour en exploiter les avantages par rapport aux circuits à 2 dimensions.

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 permet de décrire les compromis optimaux entre plusieurs mesures de performance du réseau.

Le but de ce travail est de proposer un outil qui maximise les performances des réseaux sur puce en terme de coût de communication et de surface dans un cadre multiobjectif.

Plan de document

Le premier chapitre présente le fonctionnement, les avantages et les applications des réseaux sur puces, des généralités sur les réseaux sur puces ainsi que l'intégration de la technologie 3D et ses avantages par rapport au 2D.

Le deuxième chapitre se concentre sur la personnalisation des topologies et le mapping, en explorant différentes méthodes pour adapter les réseaux sur puces.

Le troisième chapitre traite de l'optimisation multiobjectif, en mettant l'accent sur les approches permettant de trouver des solutions optimales dans le cas de plusieurs objectifs.

Le quatrième chapitre propose une solution spécifique pour résoudre les problèmes de personnalisation et de mapping.

Enfin, le cinquième chapitre présente les tests et les résultats obtenus, évaluant ainsi l'efficacité de la solution dans des scénarios réels.

Cette structure globale du document offre une compréhension approfondie des réseaux sur puces, de leur personnalisation de topologies et de l'optimisation multiobjectif, tout en proposant une solution et en évaluant ses performances.

Chapitre 1 : Les réseaux sur puce (NoCs)

1 Introduction

Durant les dernières décennies, on est passé de la conception de circuits composés de milliers de portes à des systèmes sur puce(SoC) intégrant un nombre important de cœurs pour répondre aux exigences de nouvelles applications.

Cette évolution entraine une augmentation des besoins de communication à l'intérieur de la puce pour les échanges entre les différents cœurs(IPs) qui compose le système sur puce. Les interconnexions classiques ne répondent pas aux exigences de flexibilité, consommation d'énergie et de surface réduit.

Afin de surmonter ces problèmes, un nouveau paradigme de communication est apparu, C'est le réseau sur puce (NoC : Network On Chip). Ce type d'architecture inspiré des réseaux informatiques classiques, qui peut répondre aux exigences citées auparavant.

Dans ce chapitre nous allons introduire le principe des réseaux sur puce, décrivant les types d'interconnexions classiques utilisés dans les systèmes sur puce. Ensuite nous présenterons les composants et les caractéristiques des réseaux sur puce.

2 Les systèmes sur puce

Selon la loi de Moore[1], le nombre de transistor par circuit est multiplié par deux tous les dix-huit à vingt-quatre mois. Cette loi s'applique aux puces des circuits intégrés.

Un circuit intégré est une petite puce de silicium contenant des centaines de milliers de composants et capables d'effectuer des tâches allant de la simple combinaison logique à des fonctions très complexes telles que celles requises dans les microprocesseurs[2].

Un Système sur puce ou SoC (System-on-Chip) présent dans les systèmes embarqués est un ensemble de composants matériels conçus et intégrés dans une seule puce électronique pour réaliser les fonctionnalités demandées[3].

2.1 Multiprocesseur système sur puce (MPSoC)

Un MPSoC 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.

2.2 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 tels 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 préspécifiées 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.

3 Les types d'interconnexions dans les systèmes sur puce

Différents systèmes d'interconnexions existent avec leurs avantages et leurs inconvénients pour interconnecter les composant d'un SoC dont les principaux sont : le point à point, le bus (partagé ou hiérarchique) et le réseau sur puce.

3.1 Le point à point

C'est la solution la plus simple et la première interconnexion qui est apparue pour interconnecter les différents composants dans un SoC. Elle consiste à interconnecter deux éléments communicant via un lien dédié sans aucun protocole de gestion de communication[4][5] (Figure 1).

Lors de l'utilisation d'un nombre important de liens, cette solution offre de meilleures performances en terme de parallélisme, la bande passante et de la latence. Cependant, le taux d'utilisation des liens est très bas (environ 10% selon une recherche de Dally et al[6]). Ce mode de connexion ne présente aucune flexibilité car elle n'est adaptée qu'à une architecture donnée et donc difficilement réutilisable.

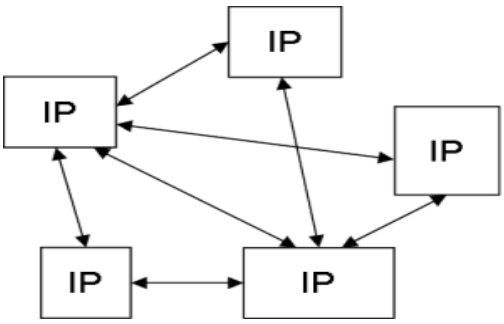


Figure 1 : point à point [7]

3.2 Le bus partagé

C'est la connexion la plus utilisée dans les SoCs modernes. Le bus est un moyen de communication sert à interconnecter un ensemble de modules via un seul support partagé [8].

L'avantage de cette architecture est la simplicité et l'extensibilité. Cependant, le bus a aussi ces limites, puisqu'il s'agit d'une ressource partagée. Il ne permet pas le parallélisme des communications (un seul transfert à la fois).

L'augmentation de nombre de composants (IPs) conduit à l'augmentation de longueur de bus, ce qui affect négativement le délai de communication et la consommation d'énergie (Figure 2).

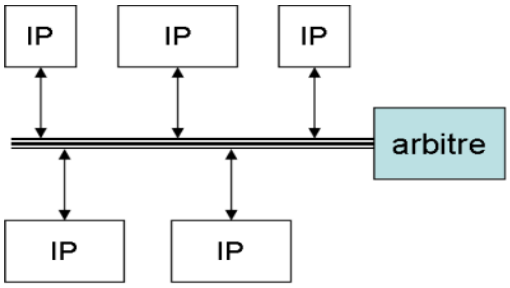


Figure 2 : bus partagé [7]

3.3 Le bus hiérarchique

Pour remédier certains problèmes rencontrés dans la structure d'interconnexion par bus, une nouvelle structure d'interconnexion qui est le bus hiérarchique a été proposée et utilisée dans l'industrie (IBM Core Connect, ARM AMBA [9]). Le principe de cette solution est de relier plusieurs bus entre eux par une passerelle et chaque bus possède son propre arbitre (Figure 3).

Cette approche présente l'avantage de répartir les propriétés intellectuelles (IP) sur plusieurs bus distincts, ce qui permet d'équilibrer la charge et de réduire la longueur des bus. Cela a pour conséquence une amélioration des performances en termes de consommation d'énergie et de latence. Cependant, cette solution pose le défi complexe de la gestion de l'adressage lors du passage d'un bus à l'autre par le biais de passerelles. [5][7].

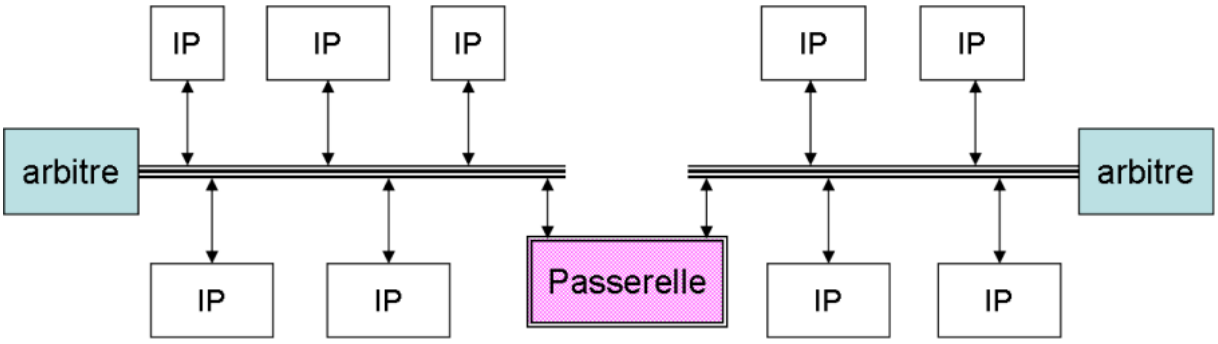


Figure 3 : bus hiérarchique[7]

3.4 Le réseau sur puce

Le réseau sur puce (NoC) consiste à adapter les notions des réseaux informatiques dans le contexte des SoCs afin de surmonter les problèmes d'interconnexions classiques. Dans les NoCs, un cœur est relié à un routeur via un réseau d'interface (NI), le transfert des données d'une source vers une destination se fait à travers des routeurs (switches) et des canaux d'interconnexion [10] (Figure 4).

Le réseau sur puce offre une meilleure performance que le bus au niveau de la bande passante, la latence, et de la consommation d'énergie. En plus cette architecture est flexible et facilement extensible, mais malheureusement cette solution a un coût de conception important.

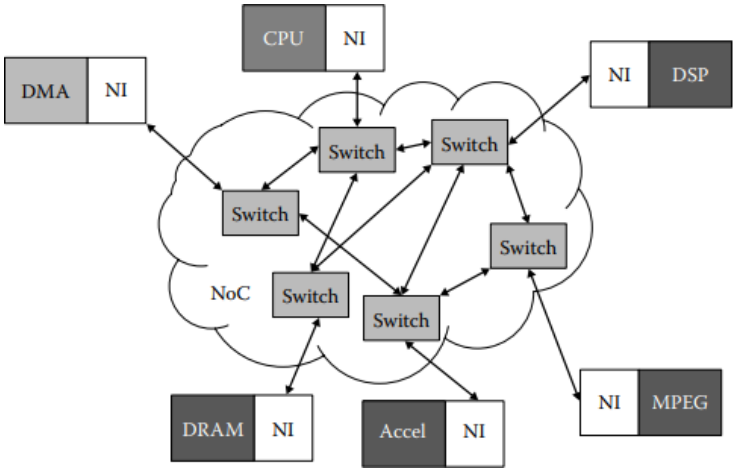


Figure 4 : Réseau sur Puce [11]

Le tableau suivant illustre une comparaison entre les différents types d'interconnexion qui peuvent être utilisées dans un système sur puce.

Tableau 1 : Comparaison entre les types d'interconnexion [12]

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

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

Le réseau sur puce et le point à point offrent une meilleure performance en terme de parallélisme par rapport au bus (partagé ou hiérarchique). Le point à point n'est pas réutilisable à cause de ses liens dédiés, par contre les réseaux sur puce et le bus sont facilement réutilisables. Pour la consommation d'énergie et la scalabilité il est claire que le réseau sur puce a des résultats plus forts que les autres communications. De plus le NoC offre l'avantage de mise à l'échelle (possibilité d'ajouter des nouveaux IPs) sans altérer le système global.

4 Les composants d'un réseau sur puce

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

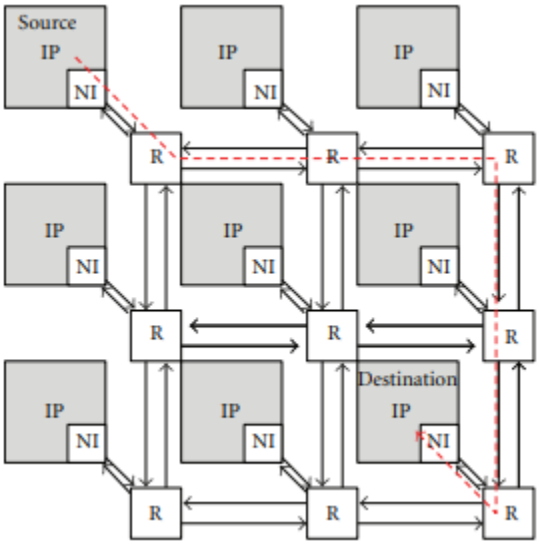


Figure 5 : Les composants d'un NoC [13]

4.1 Les interfaces réseau (NI)

L'interface réseau relie une ressource (IP) à un routeur réseau. Par conséquent, elle permet à la ressource d'envoyer des données au routeur [14]. Le rôle de NI dans les NoCs est le même rôle que celui de la carte réseau pour l'Internet qui agit comme une interface entre l'ordinateur et le réseau, en permettant l'échange de données entre les deux. L'interface réseau se compose de deux parties : une partie frontale qui est reliée à un composant fonctionnel du circuit, une partie arrière qui est relié au réseau sur puce (Figure 6). Ce bloc est important car il permet la séparation entre le calcul et la communication.

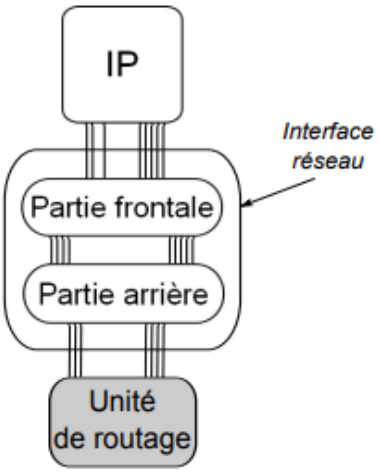


Figure 6 : Architecture d'une interface réseau [14]

4.2 Les routeurs

Leur rôle principal est d'acheminer les données d'une source à une destination en fonction de la topologie réseau et de la stratégie de routage. Un routeur est composé d'un certain nombre de ports d'entrée/sortie contenant ou pas des files d'attentes, d'une matrice de commutation, et un port local pour accéder aux unités de traitement connectées à ce routeur (Voir Figure 7). En plus de cette infrastructure de connexion physique, le routeur contient également une unité (logique) de routage et d'arbitrage, qui met en œuvre les algorithmes de routage (sélectionne la liaison de sortie pour un message entrant).

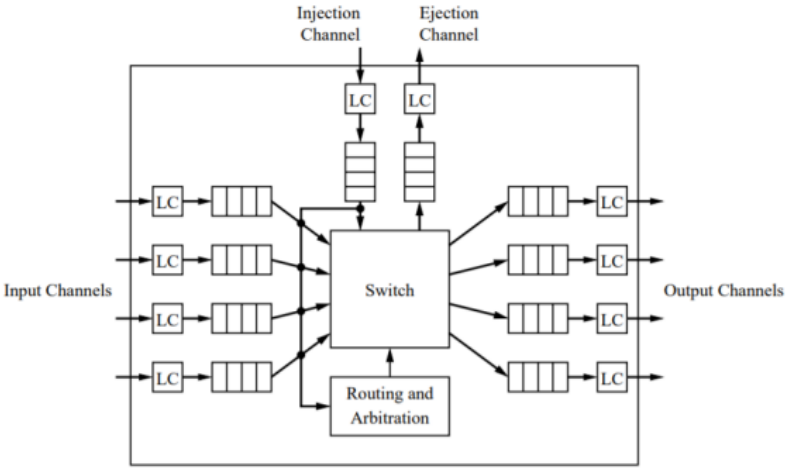


Figure 7: Architecture d'un routeur [15]

Les files d'attentes : sont utilisées pour stocker les données et permettent également de maintenir le trafic en cours sans bloquer les émetteurs tant que les files ne sont pas pleines.

Le commutateur ou matrice de commutation : qui connecte les files d'attente d'entrée aux files d'attente de sortie. Plusieurs multiplexeurs en cascade sont utilisés afin de multiplexer les ports d'entrés vers les ports de sorties.

L'unité de routage et d'arbitrage : définit comment doit être configuré le commutateur pour que les données contenues dans les tampons soient correctement aiguillées. De plus il doit gérer les conflits d'accès lorsque plusieurs files d'attente d'entrée doivent être connectées à la même file de sortie.

4.3 Les liens physiques

Ils relient les routeurs entre-eux ou les routeurs aux NI pour assurer le transfert de données. Ce sont eux qui offrent la bande passante. Un lien peut consister en plusieurs canaux virtuels et peut être monodirectionnel ou bidirectionnel [16].

5 Les Performances des réseaux sur puce

Lorsque l'on souhaite évaluer les caractéristiques d'une architecture de réseau sur puce, de nombreuses performances peuvent être prises 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 [17].

5.1 Le débit

C'est une métrique caractérisant les performances d'un réseau [18]. Le débit est le nombre total de paquets (une partie de message composé de plusieurs flits) qui atteignent leur destination par unité de temps. Chaque élément du réseau (liens, nœuds) peut alors être caractérisé par sa bande passante qui est couramment exprimée en bit par seconde (bps).

5.2 La latence de transfert

Un critère de performance important dans le cadre des réseaux sur puce est l'estimation de la latence des transferts. Elle peut se définir à plusieurs niveaux : flit, paquet (ensemble de flits) ou message (ensemble de paquets). Pour un flit, il s'agit du temps écoulé entre l'introduction du flit dans le réseau et sa réception par la ressource destinatrice. Pour un paquet, la latence correspond au temps entre l'envoi du premier flit et la réception du dernier. Dans le cas d'un message, c'est le temps entre l'envoi du premier paquet et la réception du dernier[17].

En général la latence est le nombre moyen de cycles entre le moment de génération du paquet au nœud source et de livraison du paquet au nœud destination.

5.3 La consommation d'énergie

La consommation d'énergie dans les NoC est souvent une préoccupation majeure. L'énergie est définie comme l'énergie moyenne consommée par les composants de NoC tel que les IP, les routeurs et les liens pour transmettre les données. Le contrôle du trafic réseau (par exemple l'analyse de la bande passante, c'est-à-dire la limitation de la bande passante consommée par les IPs qu'ils communiquent peu) peut aider à mieux gérer l'énergie consommée par ressources de calcul en réseau.

5.4 La surface

C'est l'espace occupée par l'ensemble des composants qui constituent le réseau sur puce.

Ces critères (le débit, la latence, la consommation d'énergie, la surface) sont directement reliés, aux choix de topologie.

6 Topologies

Comme pour les réseaux informatiques, de nombreuses topologies sont envisageables pour construire un NoC. La topologie définit comment les ressources sont reliées entre eux. Chaque

disposition obtient des caractéristiques particulières, que ce soit en cas de panne ou pour sa technique de routage. De plus chaque topologie définit certains critères physiques à savoir : le degré du nœud, le diamètre, nombre de liens [10], [11], [19]. Les topologies peuvent être classifier en deux classes : topologies directes et topologie indirectes.

6.1 Topologies directes

Dans cette configuration, chaque nœud est connecté directement à un nombre spécifique de nœuds voisins, et un message entre deux nœuds passe par un ou plusieurs nœuds intermédiaires. Les avantages de ce type de topologie résident dans leur extensibilité et leur structure réutilisable Les topologies directes les plus courantes sont :

6.1.1 La topologie maillé (2D Mesh)

La topologie en maille est l'une des plus simples et des plus étudiées grâce à son évolutivité et sa flexibilité. Elle se compose de m colonnes et n lignes. Les adresses des routeurs peuvent être facilement définies par des coordonnées XY dans la maille. Cette topologie est facilement implacable sur une technologie silicium à cause de ces liens qui ont la même longueur (Figure 8.a).

6.1.2 La topologie torus :

C'est une topologie dérivée de la structure maillé possédant une particularité dans lequel les routeurs des extrémités sont connectés les uns aux autres d'une manière symétrique (Figure 8.b). Elle offre ainsi une bande passant légèrement supérieure à celle de 2D maillée mais elle est plus complexe à implémenter.

6.1.3 La topologie anneau

La topologie en anneau relie les nœuds d'un réseau entre eux afin de former une boucle (Voir Figure8.c). Cette architecture est facilement intégrable sur silicium. Mais elle n'est pas extensible car ces performances se dégradent à mesure que le nombre d'IPs connectées augmente.

6.1.4 La topologie octogone

La topologie octogone est une version améliorée de la topologie anneau pour remédier aux problèmes de performance de cette dernière. Elle permet de réduire la latence, car la communication traverse au maximum deux routeurs au sein de réseau (Figure8.d).

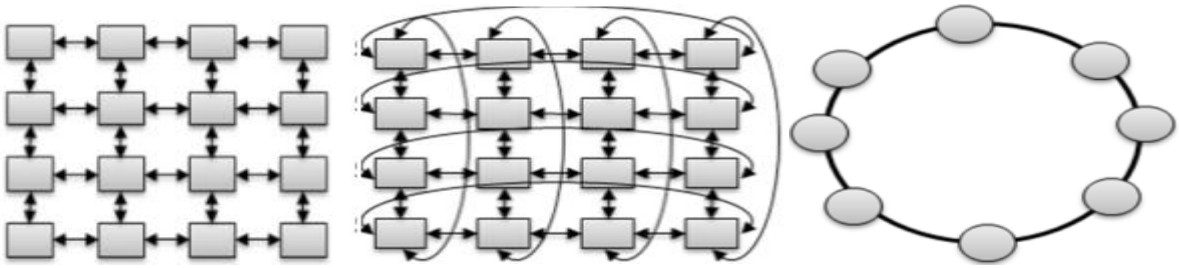
6.2 Topologies indirectes

Dans une topologie indirecte, tous les routeurs ne sont pas connectés à des unités de traitement comme dans la topologie directe. Au lieu de cela, certains routeurs ne sont utilisés que pour propager les messages à travers le réseau (envoyer les messages à des commutateurs intermédiaires). Les topologies indirectes sur mesure offrent une meilleure flexibilité pour répondre aux exigences souhaitées. D'autre part, l'un des principaux problèmes dont souffrent les topologies indirectes est le temps de conception nécessaire pour profiler l'application et décider de la meilleure disposition topologique qui satisfait ces exigences de conception [20].

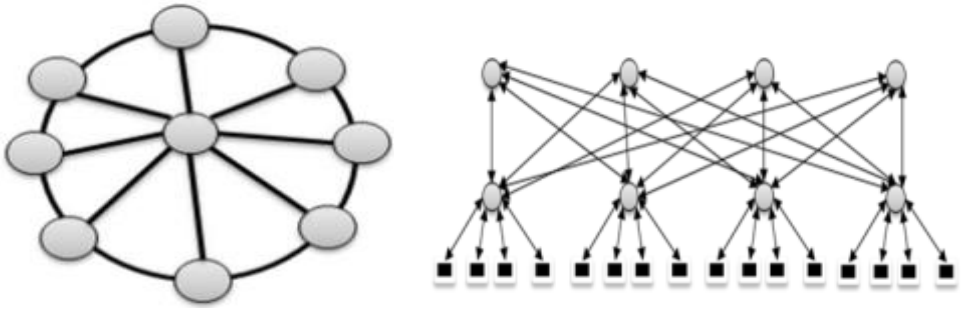
La topologie indirecte la plus courante est :

6.2.1 La topologie arbre

Dans le réseau arbre les nœuds sont des routeurs et les feuilles sont des cœurs connectés au réseau (Figure 8.e). Les routeurs qui sont au-dessus des feuilles sont appelés ses ancêtres et respectivement les feuilles qui sont au-dessous de leur ancêtre sont appelées ses descendants. Dans la topologie arbre chaque nœud a des multiples ancêtres qui signifient qu'il y a beaucoup de chemins alternatifs entre les nœuds.



Topologie maillé [a] topologie torus [b] topologie anneau[c]



Topologie octogone [d] topologie arbre [e]

Figure 8: exemples de topologies [12]

6.3 Topologie 3D

Une topologie 3D est un ensemble des plans empilés et connectés verticalement à travers des liens verticaux dites TSV, chaque plan contient des tuiles, chaque tuile peut contenir un composant (IP), un routeur, un lien vertical (Figure 9).

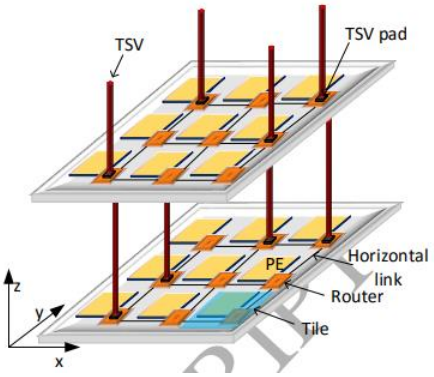


Figure 9 : Exemple d'un Noc 3D [21]

6.4 Topologie personnalisée

L'approche de personnalisation vise à générer une topologie sur mesure de type irrégulière (les nœuds ont un nombre de voisins différent) pour une application spécifique.

Une topologie personnalisée est une topologie irrégulière, elle peut être issue d'une topologie régulière (standard) qui a été modifiée au plus juste de façon à enlever ou ajouter des éléments (l'addition ou la suppression des liens et ressources d'une ou plusieurs topologies régulières).

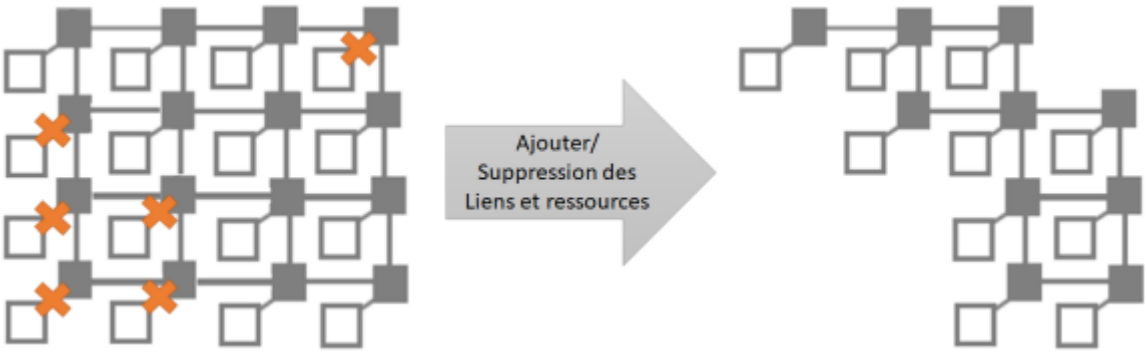


Figure 10 : exemple d'une topologie personnalisée

Les topologies 3D personnalisées dans les réseaux sur puce utilisent une disposition verticale des composants, offrant des avantages en termes de densité, de performances et d'efficacité énergétique. En empilant les composants verticalement, la longueur des interconnexions est réduite, améliorant la vitesse de communication, réduisant la latence et la consommation d'énergie, tout en augmentant la capacité de traitement et la bande passante. Ces topologies sont conçues spécifiquement pour répondre aux besoins d'applications particulières et peuvent impliquer des conceptions d'empilement stratégiques, des interconnexions avancées et des techniques de gestion thermique. Les topologies 3D personnalisées offrent une solution prometteuse pour les applications nécessitant de hautes performances dans des espaces restreints.

6.5 Topologie Standard

Une topologie est dite standard ou régulière lorsque tous les nœuds ont le même degré (tous les nœuds ont le même nombre de voisins)[22]. Les topologies de réseau régulières sont généralement adaptées pour la majorité des NoCs. Elles sont hautement réutilisables et simples à mettre en œuvre car elles ne sont pas conçues pour une application spécifique. Cependant, dans les topologies régulières l'utilisation non optimale des interconnexions entraînent une consommation d'énergie accrue.

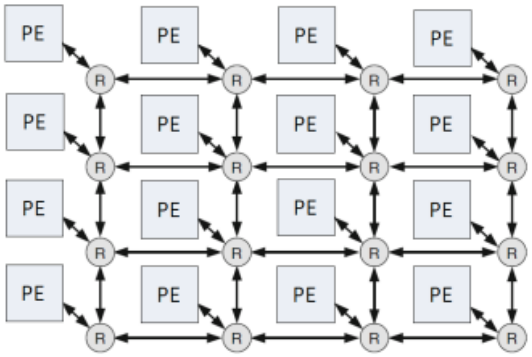


Figure 11 : Un exemple d'une topologie régulière [23]

L'intégration des circuits intégrés 3D et les liens de communication vertical améliore d'une manière remarquable les performances des réseaux sur puce. La section qui suit introduit l'intégration de la technologie 3D dans les réseaux sur puce.

7 La technologie 3D

La technologie des circuits intégrés 3D offre la promesse d'être une nouvelle façon d'augmenter les performances du système (longueur de fil réduite et donc un délai d'interconnexion réduit). Dans cette approche, plusieurs matrices (puces) en silicium sont empilées et connectés électriquement en utilisant des liens d'interconnections verticaux appellent TSV (Through Silicon Via) formant une seule puce [24], [25]. Ces TSV peuvent être placés très densément et leur longueur verticale peut être aussi petite que quelques dizaines de micromètres [26]. Grâce à ces TSV les IPs des différents plans peuvent maintenant communiquer plus rapidement, tout en consommant moins d'énergie [27], [28] (40% d'énergie économisé après l'application de cette technologie sur des mémoires puce interconnecter par Samsung [29]).

Cependant, la surface occupée par les TSVs est très importante, dans Xiangyu et al [30] la surface totale occupé par les TSVs augmente chaque fois que la taille du réseau augmente. De ce fait, le nombre total de TSVs utilisés dans la configuration affecte de manière significative le coût global des circuits intégrés 3D [31], [32].

Une recherche dans [33] a montré qu'une amélioration des performances de 40% et 36% et une diminution de 62% et 58% de la consommation d'énergie est démontrée pour un NoC 3D par rapport à une topologie NoC 2-D traditionnelle pour une taille du réseau de N = 128 et N = 256 nœuds, respectivement.

De plus, la technologie 3D offre un débit très élevé avec une faible latence par rapport à celle de 2D [34].

En général l'utilisation des circuits 3D offre des meilleurs résultats en termes de Fonctionnalité, Puissance et de performances par rapport aux circuits 2D.

8 Algorithme de routage

L'algorithme de routage détermine le chemin que doit suivre un paquet pour atteindre sa destination [35].

On peut classer les algorithmes de routage en deux catégories : algorithme déterministes et algorithme adaptif. Dans le routage déterministe le chemin entre la source et la destination est unique et prédéfinie, par exemple l'algorithme XY avec la topologie 2D Mesh est un algorithme de routage déterministe. Par contre, pour l'algorithme adaptif plusieurs chemins peuvent être considérés lors de la transmission du paquet, et le chemin sélectionné est celui qui montre le minimum possible de congestions.

8.1 L'algorithme XY

C'est un algorithme déterministe qui achemine d'abord les paquets dans la direction X ou horizontale vers la bonne colonne, puis dans la direction Y ou verticale vers la colonne appropriée. Le routage XY convient bien à un réseau utilisant une topologie maillée ou torus. Il ne se heurte jamais en situation de blocage [36], car il utilise un schéma de routage déterministe qui garantit que chaque paquet de données suit toujours le chemin le plus court vers sa destination.

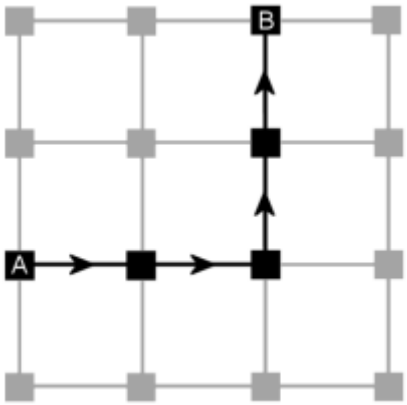


Figure 12 : Routage XY de routeur A vers routeur B [36]

9 Conclusion

Dans ce chapitre, nous avons présenté le nouveau paradigme réseau sur puce, qui est venue pour résoudre les problèmes d'interconnexions classiques tel que le parallélisme la consommation d'énergie, la surface. On a défini le réseau sur puce, on a vu ces composants qui sont des routeurs communicants entre eux via des liens physiques, et des interfaces réseaux qui permettent d'adapter les protocoles de communications des routeurs avec ceux des IPs. De plus on a présenté quelques métriques de performances qui permettent l'évaluation des NoCs.

La personnalisation des topologies et le mapping des IPs dans un réseau sur puce impliquent d'adapter la configuration du réseau en fonction des besoins de l'application. Cela inclut la conception d'une structure spécifique du réseau en tenant compte des contraintes et des exigences.

Dans le chapitre suivant on s'intéresse aux travaux de la littérature traitant la phase de choix de la topologie et au placement de composants (mapping) communicants.

Chapitre 2 : Personnalisation des topologies et phase de mapping

1 Introduction

Le problème de mapping constitue un des principaux domaines d'étude de recherche. Plusieurs algorithmes et techniques sont proposés pour avoir un maximum de performances d'une application donnée.

Le mapping est une étape importante du processus de conception du NoC. Il consiste au placement des composants communiquant sur la topologie du réseau de sorte à maximiser les performances. De plus, le choix de topologie du réseau influence ces performances.

Ce chapitre est consacré à l'étude des techniques de la littérature qui s'intéressent aux choix de la topologie et la phase de mapping. De ce fait notre étude sera divisé en deux parties, Dans la première partie, notre étude se focalise sur les travaux dédies aux topologies personnalisées. La deuxième partie étudie les méthodes traitantes la phase de mapping.

2 Phases de conception des réseaux sur puce

La conception d'un NoC passe généralement par les étapes suivantes [10], [37] :

- Choix de la topologie.
- L'ordonnancement et l'assignation des tâches aux IPs.
- Le mapping de ces IPs dans la topologie.
- Définir la stratégie de routage.
- Vérification des performances.
- Simulation physique de réseau.

Notre travail se situe dans le mapping des Ips, et le choix de topologie. Le choix de topologie permet de spécifie la façon dont les nœuds du réseau et ses liens sont interconnectes. Par contre la phase de mapping consiste au placement des composants communicants (IPs) d'une application spécifique sur la topologie du réseau afin de maximiser les performances.

3 Topologies personnalisées (état de l'art)

Les NoCs de la littérature utilisent généralement des topologies régulières car elle présente l'intérêt d'être une topologie de structure mathématique simple, ce qui permet d'utiliser des règles de routage simples. Une topologie personnalisée permet plus de liberté et ainsi de tailler précisément le réseau ce qui permet d'avoir une énergie basse. Cependant, elle nécessite en revanche une plus grande attention pour le routage.

Les topologies régulières et personnalisées ont des caractéristiques qui peuvent être résumés dans le tableau de comparaison suivant :

Tableau 2 : comparaison entre topologie régulière et personnalisée [38]

Topologie régulière	**Topologie personnalisée**
Maillée, Torus , . . .	Topologie irrégulière
Soc utilité générale	Soc application spécifique
Routeurs homogènes	Routeurs hétérogènes
Topologie réutilisées	Conception des routeurs réutilisées
Peu de temps de conception	Long temps de conception
Performances basses	Performances élevées
Energie élevées	Energie basse

Plusieurs solutions sont proposées afin de générer des topologies irrégulières pour les applications spécifiques.

Dans [39], un algorithme génétique a été proposé dont l'objectif principal est de minimiser la consommation d'énergie et la surface silicium tout en maximisant les performances. Cet algorithme itératif permet de générer des configurations (architectures) aléatoirement et affecter les IPs d'une application aléatoirement, puis invoquant un algorithme appelé MSP (Modified Shortest Path) afin de générer le chemin le plus court entre deux extrémités de communication source et destination.

Dans [40], une méthode utilisant l'algorithme de "branch and bound" a été proposée qui explore toutes les configurations possibles afin de choisir la configuration qui minimise la consommation d'énergie.

Une autre méthode utilisant les techniques de la programmation linéaire a été proposée dans [41]. L'objective est de minimiser la consommation d'énergie et la surface du silicium tout en minimisant le nombre des routeurs.

Une approche commence avec une architecture d'interconnexion régulière générique telle que 2D Mesh et y insèrent des liens supplémentaires. Par exemple l'approche qui a été proposée dans [42], Elle a été appliquée à une architecture 2D Mesh en insérant quelques liens spécifiques entre des routeurs en fonction du flux de données.

L'approche proposée par Chatha et al [43], est une approche de suppressions de liens basée sur une topologie régulière 2D-mesh pour minimiser la surface. Elle consiste à partir du placement des IPs allouer à chaque IP un routeur parmi les routeurs de ces coins. Ensuite, éliminer les routeurs non utilisés, cette dernière a subi un changement en nombre des liens associées à ces routeurs.

Pour les topologies 3D, Vertically-Partially-Connected 3D-NoC est une solution pour réduire le nombre des liens TSV afin de minimiser la surface de système [44], [45]. Parmi les topologies 3D partiellement connectées, La topologie MMT 3D on trouve, et la topologie RNT (RNT : Recursive Network Topology) 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 [46].

Deux méthodes de personnalisation sont présentées dans [47], Le premier type de personnalisation est basé l'insertion de liens croisés génériques (qui sont les connexions entre

n'importe quel couple de nœuds) tandis que le second type est basé sur l'insertion de liens croisés spidergon (qui sont des connexions entre un nœud et son homologue diagonal dans le réseau).

Dans [48], plusieurs mappings sont générés en partitionnant le graphe d'application, Pour chaque mapping, une topologie est générée et évaluée afin de générer une seule topologie personnalisée à la fin.

Le tableau suivant résume les caractéristiques des différentes approches de personnalisations :

Tableau 3 : Synthèse sur les approches de personnalisation

Référence	Méthode	Topologie	Objectif	Année
[43]	Elimination des liens et des routeurs	2D Mesh	Energie + surface	2008
[49]	Long-Range Link Insertion	2D Mesh	Latence	2006
[50]	Cross-by-pass link (insertion)	2D Mesh	Latence + débit	2017
[51]	Suppression des liens	3D Mesh	Surface + énergie	2012
[52]	Sparse Hamming Graphe	3D Mesh	Surface	2022
[53]	GA	2D Personnalisée	Surface + énergie	2005
[54]	Recuit Simulé	personnalisée	latence	2009
[48]	Génération des topologies par partitionnement	2D Mesh	Energie	2006
[55]	TTOT	2D	Consommation d'énergie + latence	2016

Voici une analyse comparative entre les topologies 2D et 3D, ainsi que les topologies personnalisées et non personnalisées.

Topologie 2D : Dans une topologie 2D, les nœuds sont organisés en une grille bidimensionnelle. Cette topologie offre une structure régulière et facile à concevoir, mais elle peut souffrir de problèmes de latence et de congestion, en particulier pour les grandes échelles.

Topologie 3D : Dans une topologie 3D, les nœuds sont empilés en couches verticales pour former une structure tridimensionnelle. Cette topologie permet d'augmenter la densité des nœuds et d'améliorer la bande passante et la latence en réduisant les distances physiques entre les nœuds. Cependant, la conception et la gestion d'un réseau 3D peuvent être plus complexes.

Topologie personnalisée : Une topologie personnalisée est conçue spécifiquement pour répondre aux besoins d'une application spécifique. Elle peut être optimisée la latence, la bande passante accrue et la consommation d'énergie. Cependant, le développement d'une topologie personnalisée peut être coûteux.

Topologie non personnalisée : Une topologie non personnalisée, est généralement utilisée comme une solution générique pour différentes applications. Elle peut être plus facile à mettre en œuvre et à intégrer, mais peut ne pas offrir les meilleures performances pour des applications spécifiques.

D'après ces travaux (Tableau 3), une topologie peut être personnalisé dans le contexte d'application spécifique (dont les caractéristiques particulièrement la communication entre composants sont connus au moment de la conception du circuit).

Il est possible d'envisager des améliorations et des combinaisons de topologies existantes. Par exemple, nous pouvons envisager d'utiliser une topologie 3D pour augmenter la densité des nœuds et améliorer les performances dans notre domaine d'application spécifique. Nous pouvons également explorer des topologies personnalisées pour optimiser les performances en fonction de nos besoins spécifiques.

Après le choix de la topologie du réseau sur puce, nous abordons dans la section suivante la phase de mapping qui consiste au placement des composants sur cette topologie.

4 Phase de mapping

La phase de mapping est une phase centrale dans la conception des NoCs, dont le résultat a un impact direct sur les performances du système. Elle consiste à placer chaque élément sur l'architecture du réseau sur puce (affecter chaque IP de l'application à une tuile de l'architecture) tout en respectant les contraintes imposées (minimisation de cout de communication, la surface, la consommation d'énergie ...) [56].

4.1 Types de mapping

Deux types de mapping sont distingués, mapping statique et mapping dynamique.

4.1.1 Mapping statique

Dans ce type de mapping, toutes les IPs sont affectées aux tuiles avant l'exécution de l'application (le placement se fait lors de la conception de l'architecture). L'avantage principale de ce type est la vue globale du système [57].

4.1.2 Mapping dynamique

Dans le mapping dynamique, les tâches sont affectées durant l'exécution de l'application. Une tâche peut être ajouter, modifier ou supprimer. Il peut offrir de meilleures performances et apporter des avantages aux systèmes tels que la réduction de la consommation d'énergie [57].

5 Le problème de mapping

Le problème du mapping est un problème d'optimisation combinatoire NP difficile. Théoriquement, placer N IPs dans M nœuds du réseau ($N \leq M$) implique $M ! / (M-N) !$ arrangements d'IPs possibles dans les nœuds 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 [58].

6 Résolution d'un problème de mapping

Pour résoudre ce problème, deux classes de méthodes sont utilisées, les méthodes exactes et les méthodes approchées.

6.1 Méthodes exactes

Les méthodes exactes garanties de trouver solutions optimales, mais le temps d'exécution est très important si la taille de problème est très grande [59].

6.2 Méthodes approchées

Les méthodes approchées offrent l'alternative de disposer d'une solution optimale ou proche de l'optimale en un temps réduit même lorsque la taille du problème a traité est grand.

Parmi les méthodes approchées en trouve les méthodes heuristiques et métaheuristiques (figure 13). Les méthodes métaheuristiques basent sur des idées générales inspirées de la nature, alors que les heuristiques se sont des méthodes développées pour résoudre un problème particulier. [57], [60].

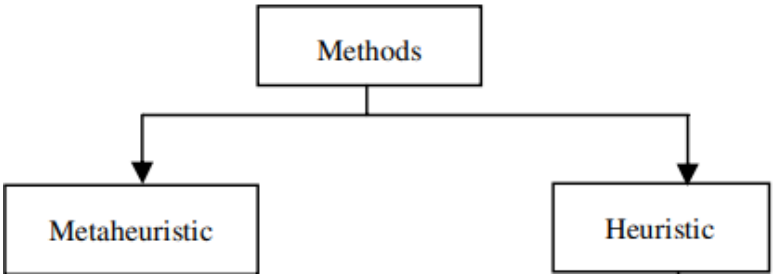


Figure 13 : Classification selon la méthode choisie - heuristique ou métaheuristique [58]

Parmi les techniques de mapping existants dans la littérature on trouve :

Technique NMAP (network mapper) [61], il permet de placer les IPs sur une topologie 2D maillé, cette technique permet de minimiser le coût de communication, elle procède en 3 étapes :

- Placer les IPs qui ont une communication maximale avec leurs voisins.
- Placer les IPs qui communiquent le plus avec les IPs de l'étape 1.
- Placer le reste des IPs avec les IPs de l'étape 2 déjà placés.

Technique PMAP [62], cet algorithme se fait en deux phases, la première consiste à placer les IPs qui communiquent le plus entre eux sur des tuiles adjacentes, la deuxième phase consiste à placer le reste des IPs en fonction des IPs déjà placés.

La technique GBMAP [63], basé sur l'algorithme génétique, elle est appliquée sur une topologie maillé, dont l'objectif et de réduire la consommation d'énergie et les besoins en bande passante.

Un algorithme à base de cluster appelé Cluster-based Simulated Annealing est présentée dans [64] pour améliorer la technique métaheuristique le recuit simulé. Il regroupe les IPs qui communiquent le plus dans un même cluster et pour chaque cluster une région de réseau est allouée. L'objectif de cet algorithme et la minimisation de coût de communication.

La technique SPIRAL [65], son but et de minimiser la consommation d'énergie. Elle affecte les tâches de l'application au ressources en démarrant du centre de l'architecture jusqu'à l'arriver aux tuiles frontières (figure 14).

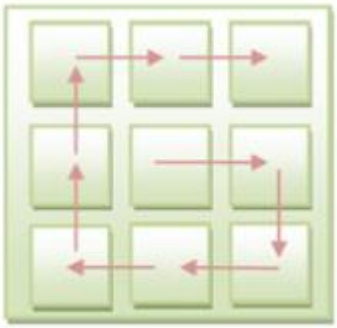


Figure 14 : Exemple de la technique SPIRAL [65]

Le travail présente dans [66] utilise une technique d'optimisation des essaims de poulets (CSO) comme une solution pour mapper une application a une topologie Noc 3D et visait à réduire le coût de communication, et pour gérer le mapping initial cet algorithme utilise la technique de regroupement des k plus proches voisins.

Un algorithme d'optimisation sailfish (SFOA) inspire de la nature est utilisée pour le mapping optimisé du graphe de tâches de l'application sur un Noc bidimensionnel avec une topologie maillée, via l'utilisation de l'approche de clustering partagé K-plus proche voisin dans le but de minimiser le temps de calcul, la consommation d'énergie et le coût de communication [67].

Dans [68], une technique de mapping pour un NoC 2D est présentée. L'objectif principal est de construire une chaîne de cœurs liés qui peut être utilisée pour construire un nouveau système de mappage.

Le tableau suivant résume les caractéristiques des travaux de mapping existants dans la littérature.

Tableau 4 : Synthèse sur les approches de mapping

Référence	Méthode	Benchmarks	Topologie	Performances	Année
[69]	MILP	PIP, HDTV, MWA, VOPD	2D Mesh	Consommation d'énergie	2004
[70]	ILP	MPEG4, PIP	2D Mesh	Coût de communication	2016
[71]	GA	VOPD, MPEG4	2D Mesh	Energie + latence + coût	2009

[72]	PSO	VOPD, MPEG4, PIP	2D/3D Mesh	Latence + débit + coût de communication	2011
[73]	ACO	VOPD, MPEG4	Mesh	Bande passante + énergie	2011
[74]	CHMAP	VOPD, MPEG4, MWD, PIP	Mesh	Coût de communication	2016
[75]	CastNet	VOPD, MPEG4, MWD, G25	Mesh	Coût de communication	2011
[76]	Branch and Bound	VOPD, MPEG4	2D/3D Mesh	Consommation d'énergie	2005
[77]	GHA	VOPD, MPEG4, MDWD, PIP	2D Mesh	Latence + consommation d'énergie	2018
[78]	BA	VOPD, MWD	3D Mesh	Consommation d'énergie	2017
[67]	SFO	VOPD, MPEG	2D	Consommation d'énergie	2021

Pour résoudre le problème de mapping, les algorithmes basés sur les méthodes exactes ne sont pas pratiques pour les NoCs ayant un grand nombre de cœurs. Cependant les méthodes approchées basés sur des heuristiques et méta heuristiques comme AG, PSO, SFO, BA permettent des meilleures performances en un temps raisonnable.

La majorité des travaux déjà cités utilisent la topologie 2D-Mesh pour le but d'améliorer leur performance. Ceci est dû à la facilité de son implantation sur la technologie silicium et sa capacité de s'adapter à une taille de réseau variable. L'augmentation du nombre de cœurs dans les topologies 2D a conduit à l'augmentation des dimensions des circuits et à l'allongement des fils de communication. La technologie de fabrication de circuits intégrés 3D a été récemment développée.

Traiter en parallèle la phase de mapping et la personnalisation de la topologie nous permettra d'optimiser plusieurs objectifs antagonistes : 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 partiellement connectée.

7 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 méthodes de personnalisation. L'utilisation des liens verticaux (TSV) doit être contrôler afin de maintenir l'équilibre entre les performances et le coût en surface dans le système.

Pour la deuxième partie de ce chapitre, on a abordé le processus de mapping, leurs types ainsi que les travaux existants dans la littérature. On conclut qu'il existe une relation étroite entre le mapping et le choix de topologie, cela nous a permis d'optimiser plusieurs objectifs. De ce fait une étude approfondie sur l'optimisation multiobjectif sera détaillée dans le chapitre suivant.

Chapitre 3 : L'optimisation multiobjectif

1 Introduction

Dans le domaine industriel, généralement les problèmes d'optimisation sont de nature multiobjectif puisque plusieurs critères permettent de caractériser une solution. Pour les réseaux sur puce les critères les plus répondus sont le coût de communication, la consommation d'énergie, la surface et la latence.

Trouver une combinaison optimale de ces critères afin de maximiser les performances d'un NoC est un problème d'optimisation multiobjectif. De ce fait, ce chapitre sera consacré à l'étude des méthodes d'optimisation multiobjective.

2 Optimisation multi-objectif

Dans un problème d'optimisation multiobectif, il n'y a pas une solution optimale unique, mais un ensemble de solutions, car en général aucune solution n'est la meilleure vis-à-vis de tous les critères simultanément [79]. Donc Le caractère de problème multiobjectif est donné par l'existence d'objectifs contradictoires signifiants que l'amélioration d'un objectif implique la détérioration d'un autre.

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 [80] [81].

La plupart des problèmes d'optimisation réels sont décrits à l'aide de plusieurs objectifs souvent contradictoires devant être optimisés simultanément. La solution optimale correspond alors aux meilleurs compromis possibles permettant de résoudre le problème. Il s'agit de l'optimisation mutiobjectif, qui fournit aux décideurs un ensemble de solutions optimales dit Front de Pareto [82].

Les problèmes d'optimisation ont été définis une ou plusieurs fonction(s) objective(s) et un ensemble de contraintes, tel qu'une fonction objective représente le but à atteindre et définit un espace de solutions au problème [83]. On peut formaliser un problème d'optimisation multiobjectif comme suit :

$$\mathbf{min}\, \boldsymbol{F}(\boldsymbol{x}) = (\boldsymbol{f_1}(\boldsymbol{x}), \boldsymbol{f_2}(\boldsymbol{x}), \dots, \boldsymbol{f_m}(\boldsymbol{x})) \qquad (1)$$

$$\text{sous la contrainte } \boldsymbol{x} \in \boldsymbol{C}$$

Où $\boldsymbol{x} = (\boldsymbol{x1}, \dots, \boldsymbol{xn})$ est le vecteur représentant les variables de décision ; C représente l'ensemble des solutions réalisables associé à des contraintes d'égalité, d'inégalité et des bornes explicites (espace de décision), et $\boldsymbol{F}(\boldsymbol{x}) = (\boldsymbol{f_1}(\boldsymbol{x}), \boldsymbol{f_2}(\boldsymbol{x}), \dots, \boldsymbol{f_m}(\boldsymbol{x}))$ est le vecteur de fonctions objectifs à optimiser (ou critères de décision) avec $\boldsymbol{m} \geq \boldsymbol{2}$ le nombre de fonctions objectifs.

La résolution de ce problème consiste à minimiser aux mieux m fonctions objectif. Donc le but de l'optimisation multi-objectif est de minimiser ce groupe de fonctions.

2.1 La dominance

La résolution d'un problème d'optimisation multiobjectif ne donne pas une solution unique mais plusieurs solutions possibles. Dans ce cas, la résolution donne souvent un ensemble de solutions non optimales car elles ne minimisent pas toutes les fonctions objectives. Ceci est de fait que des objectifs peuvent être contradictoires (l'amélioration d'un objectif entraine la dégradation d'un autre objectif).

Lorsque on obtient des solutions d'un problème multiobjectif ces solutions sont appelées des solutions de compromis. Pour trouver des meilleurs compromis une relation d'ordre a été identifier, cette relation appelée la dominance.

Parmi les relations de dominance les plus connues on trouve la dominance au sens de Pareto [84].

On dit qu'un solution $\boldsymbol{X}$ domine $\boldsymbol{Y}$ au sens de Pareto si :

- $\boldsymbol{X}$ est strictement meilleur que $\boldsymbol{Y}$ dans tous les objectifs.
- Ou $\boldsymbol{X}$ est strictement meilleur que $\boldsymbol{Y}$ dans au moins un objectif, et $\boldsymbol{X}$ égale à $\boldsymbol{Y}$ dans les restes des objectifs.

La formule mathématique de dominance au sens de Pareto pour un problème d'optimisation de deux objectifs est :

$$\boldsymbol{F_1(x) \leq F_1(y)\ et\ F_2(x) < F_2(y), ou\ F_1(x) < F_1(y)\ et\ F_2(x) \leq F_2(y).}$$

3 Les méthodes de résolution des problèmes multiobjectif

Les méthodes d'optimisation multicritères s'intéressent aux problèmes essayant d'optimiser simultanément plusieurs objectifs. Elles peuvent être classées selon le schéma suivant [85] :

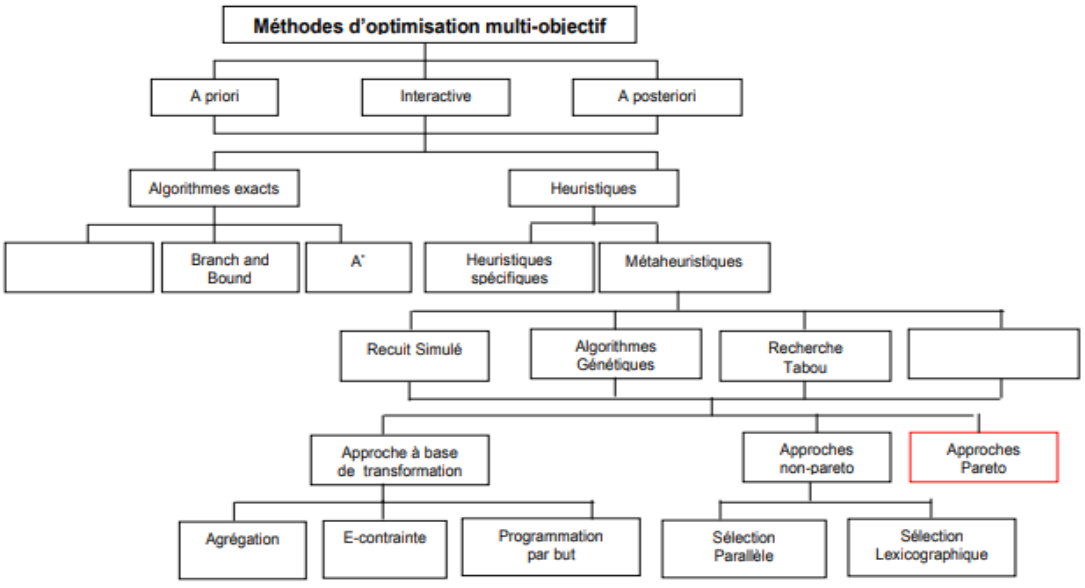


Figure 15 : Classification des Méthodes d'optimisation multiobjectif [82]

Dans la littérature, nous rencontrons deux classifications différentes des méthodes de résolution des problèmes. La première classification divise ces méthodes en trois familles suivant la coopération entre la méthode d'optimisation et le décideur :

3.1 Les méthodes à priori

Le décideur définit le compromis qu'il désire réaliser avant de lancer la méthode d'optimisation. Dans cette famille l'utilisateur présente l'importance de chaque objectif voulu atteindre afin de pouvoir transformer le problème multi-objectif en mono-objectif.

3.2 Les méthodes progressives

Également appelées les méthodes interactives. L'utilisateur affine son choix de compromis au fur et à mesure du déroulement de l'optimisation. Il peut intervenir itérativement en modifiant certaines valeurs ou contraintes pour diriger le processus de conception vers l'optimum.

3.3 Les méthodes à posteriori

Dans cette famille le processus d'optimisation détermine un ensemble de solutions et la sélection de la meilleure solution ne se fait qu'à la fin du processus d'optimisation. Les méthodes à posteriori les plus utilisées sont celles basées sur les algorithmes génétiques, appelés aussi algorithmes évolutionnaires [86].

La deuxième classification divise les méthodes suivant leur façon de traiter les fonctions objectives. On distingue les méthodes de transformation, les méthodes basées sur l'équilibre de Pareto et les méthodes non Pareto, que nous détaillons dans la suite.

3.4 Les méthodes de transformation

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

3.4.1 Les méthodes d'agrégation

Les méthodes agrégatives fusionnent les différentes fonctions objectives pour se ramener à un problème d'optimisation mono-objectif, en regroupant les critères à optimiser dans une unique fonction objective. La méthode d'agrégation la plus connue et la plus employée est la moyenne pondérée. Cette méthode consiste à additionner tous les objectifs en affectant un cœfficient de poids (équation 2) [87].

$$\boldsymbol{F(x) = \sum_{i=1}^{m} \omega_i f_i(x)} \qquad (2)$$

où les poids $\boldsymbol{\omega_i \geq 0}$ et $\boldsymbol{\sum_{i=1}^{m} \omega_i = 1}$.

Quoi que largement utilisée, cette méthode présente toutefois quelques problèmes. D'un côté, le décideur doit déterminer a priori les poids de chaque critère. Ceci repose sur la connaissance

du problème par le décideur. D'un autre côté, il doit pouvoir exprimer l'interaction entre les différents critères [87].

3.4.2 Les méthode ε-contrainte

Elle est aussi dite méthode du compromis [88]. Elle consiste à choisir une seule fonction objective à optimiser et faire la transformation du reste des objectifs en contraintes.

3.4.3 Programmation par but

On définit un ensemble de buts qu'on espère atteindre pour chaque fonction objective. L'algorithme tente de minimiser l'écart entre la solution courante et ses buts. Cette méthode est efficace et facile à implémenter mais la définition des buts à atteindre est souvent coûteuse [82].

3.5 Approches non Pareto

Ces approches possèdent un processus qui traite séparément les différents objectifs. Elles sont efficaces et faciles à implémenter seulement pour les problèmes multiobjectifs avec un nombre réduit d'objectifs [82].

3.6 Approches Pareto

La notion de solution optimale unique dans l'optimisation monoobjectif disparait pour les problèmes d'optimisation multiobjectif au profit de la notion d'ensemble de solutions Pareto optimales, selon le critère de dominance au sens de Pareto. On appellera front de Pareto (ou surface de compromis) d'un problème, l'ensemble des points de l'espace de recherche tel qu'il n'existe aucun point qui est strictement meilleur que les autres sur tous les critères simultanément. Il s'agit de l'ensemble des meilleurs compromis réalisables entre les objectifs contradictoires, et le décideur qui aura pour rôle de choisir la solution à retenir.

Ces approches utilisent la notion de dominance lors de la sélection des solutions. De plus, elles ne transforment pas les objectifs du problème. Dans ce type de méthode Il existe un équilibre tel que l'on ne peut pas améliorer un critère sans détériorer au moins un des autres critères.

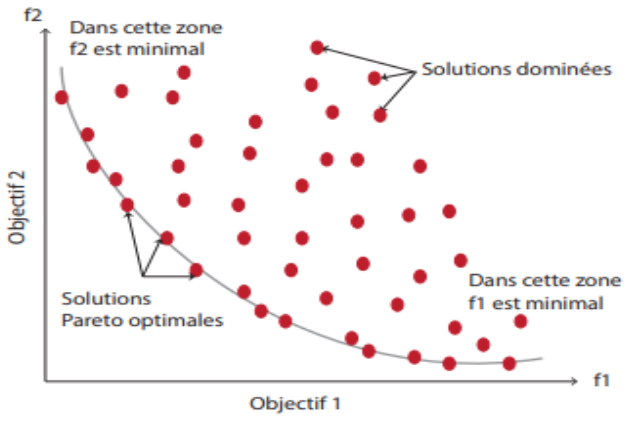


Figure 16 : Front de Pareto pour un problème de minimisation de deux objectifs [87]

4 Les algorithmes de résolution des problèmes multiobjectifs

Les algorithmes de résolutions des problème multiobjectifs se répartissent en deux grandes catégories :

4.1 Les méthodes exactes

Les méthodes exactes garantissent de trouver l'ensemble des solutions optimales. Pour ce faire ces méthodes explorent l'espace de tous les combinaisons possibles. Cependant, leurs temps d'exécution accroit de manière exponentielle en fonction de la taille du problème, ce qui fait qu'elles ne sont appropriées que lorsque la taille du problème est raisonnable, Ce qui ne poussent à s'intéresser aux méthodes approchées (heuristiques et métaheuristiques). Parmi les méthodes de résolution exacte on trouve : la programmation dynamique [89], [90].

4.2 Les méthodes approchées

On distingue dans cette catégorie deux classes, les heuristiques et les métaheuristiques.

Une méthode approché heuristique pour un problème d'optimisation est un algorithme qui a pour but de trouver une solution réalisable pour un problème difficile en un temps raisonnable, tenant compte des critères et des contraintes mais sans garantie l'optimalité.

Les métaheuristiques sont des méthodes générales de recherche dédiées aux problèmes d'optimisation difficiles [91], elles peuvent atteindre des solutions en un temps raisonnable. Elles reprennent des idées qui l'on trouve dans la vie courante, et elles sont en générale présenter sous forme d'inspiration. Ces méthodes nécessitent quelque transformation avant de pouvoir être appliquée à la résolution d'un problème donné.

On oppose toujours les méthodes approchées aux méthodes exactes, qui trouve toujours l'optimum, mais leur inconvénient est le temps de résolution.

Les méthodes approchées métaheuristiques incluent notamment le recuit simulé [92], qui se base sur solution unique, les algorithmes génétiques [93], et les algorithmes de recherche d'essaim comme PSO [94], [95] qui sont basées sur population des solutions.

4.2.1 Algorithmes basées sur solution unique

En général, les métaheuristiques à base de solution unique sont plutôt axées sur l'exploitation de l'espace de recherche, on n'est donc jamais sûr d'obtenir l'optimum. Les métaheuristiques à solution unique commencent avec une seule solution initiale et s'en éloignent progressivement, en construisant une trajectoire dans l'espace de recherche, pour cela elles sont appelées les méthodes de trajectoire [87].

4.2.2 Les algorithmes basés sur population des solutions

Contrairement aux algorithmes partant d'une solution singulière, les métaheuristiques base de population de solutions sont plus utilisées pour résoudre des problèmes d'optimisation

multibjective. Parmi les algorithmes basés sur population de solutions en trouve l'algorithme Condor des Andes (ACA).

5 Optimisation par condor des Andes

Le condor des Andes est un oiseau volant appartenant à la famille des Cathartidae. Ils sont répartis dans les Andes en Amérique du Sud le long de la côte Pacifique. Pavez a présenté un algorithme méta-heuristique dans [96]. L'algorithme a été développé à partir des observations faites sur le comportement de vol des condors des Andes pour la recherche de nourriture à un moment différent de l'année. La distance parcourue par un condor des andes pour chercher de la nourriture dépend de la saison en cours. Un condor des Andes voyage souvent plus loin de son nid au printemps et en été qu'en hiver et en automne. Ce comportement traversé donne lieu au taux d'exploration et d'intensification basé sur la valeur de fitness moyenne. Si cette valeur est bonne, le nombre d'explorations diminue et le nombre d'intensifications augmente. Par conséquent, la valeur de fitness moyenne contrôle le bon équilibre entre le taux d'exploration et le taux d'intensification [97].

Notre méthode est basée sur l'optimisation par Condor des Andes. C'est une méthode basée sur population de solution, ce qui permet de manipuler plusieurs solutions (de compromis) à chaque itération de l'algorithme.

Les méthodes basées sur l'AG, PSO, Recuit Simulé sont beaucoup utilisées dans la littérature, pour cela nous avons choisi la méthode ACA nouvellement proposée et qu'elle a prouvé s'efficacité en terme des résultats obtenus dans la littérature selon [97].

5.1 Algorithme général ACA

Entrés : AC, N_C, DP, PC

Sortie : AC_b *// la meilleur solution*

Début

Initialiser_Population () ;

Calculer_Coût () ;

Calculer_Fitness_Moyen () ;

Trier_solutions_du_meileur_ Coût _au_pire () ;

Calculer_Quantité_d'intensification(QI)_et_Quantité_d'exploration(QE) () ;

Mettre_Ajour_Etat_Condors () ;

Tantque (itr < MaxItr) faire

Pour chaque solution AC_i ***dans population*** AC ***faire***

Si Etat de (AC_i) ***== Exploration alors***

Exploration_Movement (AC_i)***;***

Sinon

Intensification_Movement(AC_i)***;***

Fin Si

Fin Pour

Calculer_Coût () ;

Trier_solutions_du_meileur_au_pire () ;

Mettre_Ajour_Etat_Condors () ;

Trouver_la meillere_Solution() ;

Mettre à jour AC_b***;***

Fin Tantque ;

Fin ;

Les pseudo code de chaque procédure utilisée dans l'algorithme ACA sont définis dans les sections suivants.

Procédure Initialiser_Population ()

```
Entrées :  Nc // la taille de Population
Sortie AC // Population de condors (Solutions)
  Début
    Pour i allant de 1 à Nc  faire
      Générer Aléatoirement les positions et le nombre de TSVs ;
      Générer Aléatoirement les placements des IPs ;
    Fin Pour
   Retourner AC ;
  Fin
```

Procédure Calculer_Coût ()

```
Entrées : AC, Nc
Sortie : coût
  Début
    Pour i allant de 1 à Nc faire
      Calculer le coût de AC_i selon l'équation (2) (voir chapitre 4);
    Fin Pour
    Retourner coût ;
 Fin
```

Procédure Trier_solutions () ;

```
Entrées :  AC, Nc
  Début
       Tant que la population est non triée faire
          Pour i allant de 1 à Nc-2 faire
            Si (coût de AC_i > coût de AC_{i+1}) alors
               permuter (AC_i , AC_{i+1})
            Fin Si
          Fin Pour
       Fin Tant que
Fin
```

Procédure Mettre_Ajour_Etat_Condors () ;

```
Entrées : AC, Qi, Qe, Nc
  Début
    Pour i allant de 0 à Qi-1 faire
      Etat de AC_i = Intensification;
    Fin Pour
    Pour i allant de Qi à Nc-1 faire
      Etat de AC_i = Exploration;
    Fin Pour
 Fin
```

Procédure CalculerQi_Qe () ;

```
Entrées : DP, PC, AFnew, AF
Sorties : QE, QI
Début
  Si AFnew ≥ AF alors
    DP = DP + PC
    Si DP > 1 alors
     DP = 1
    Fin si
  Sinon
     DP = DP - PC
     Si DP < 0 alors
       DP = 0
    Fin Si
Fin Si
QE = Nc * DP
QI = Nc - QE
 Retourner QE et QI
Fin
```

Procédure Calculer_Fitness_Moyen () ;

```
Entrées : AC, Nc
Sortie : AF
  Début
    AF =0 ;
    Pour i allant de 1 à Nc faire
      AF += coût de AC_i ;
    Fin Pour
    AF = AF / Nc;
    Retourner AF ;
 Fin
```

Procédure Exploration_Movement ()

```
Entrées : AC, Nc
Début
Pour i allant de 1 à Nc faire
  Changer aléatoirement la position de chaque Ip de  AC_i ;
   Changer aléatoirement les position et les nombres des TSVs de AC_i ;
Fin pour
Fin
```

Procédure intensification_Movement ()

```
Entrées : AC, Nc
Début
Pour i allant de 0 à Nc-1 faire
  Sélectionner aléatoirement deux IPs de AC_i ;
  Permuter les positions de ces deux IPs ;
   Fin pour
Fin
```

La méthode de mapping proposée basée sur l'ACA prend deux graphes, à savoir **APG** et **ARG** (voir chapitre 4), en entrée et réalise adéquatement un processus de mapping d'une application à la plateforme NoC. Les paramètres utilisés dans la méthode basée sur l'ACA pour le mapping des applications sont répertoriés dans la section suivant :

$\boldsymbol{AC}$: Population de solutions.

$\boldsymbol{AC_i}$: Condor des Andes, Une solution i de la population.

$\boldsymbol{AC_b}$: Meilleur condor des Andes (meilleur solution).

$\boldsymbol{N_C}$ **:** Nombre de condors (solutions). Tels que $\boldsymbol{N_C > 2}$. C'est la taille de la population.

$\boldsymbol{AF}$: Fitness moyen de la population calculé avec la formule suivante :

$$\boldsymbol{AF} = \sum_{\boldsymbol{i=1}}^{\boldsymbol{N_C}} \frac{\boldsymbol{Fonction\ Objective(AC_i)}}{\boldsymbol{N_C}}$$

$\boldsymbol{AF_{new}}$ **:** Nouveau fitness moyen. Fitness moyen de la nouvelle population également calculé en utilisant la formule précédente.

$\boldsymbol{DP}$: Paramètre de distribution tel que $\boldsymbol{DP} \in [\boldsymbol{0}, \boldsymbol{1}]$. Répartir la population en exploration et en intensification.

$\boldsymbol{PC}$ **:** Pourcentage de changement. Ajustement du niveau d'exploration dans la population, où $\boldsymbol{PC} \in [\boldsymbol{0}, \boldsymbol{1}]$**.**

$\boldsymbol{QE}$ **:** Quantité d'exploration, où $\boldsymbol{QE} \in [\boldsymbol{0}, \boldsymbol{N_C}]$, elle peut être calculée par $\boldsymbol{QE} = \boldsymbol{N_C} * \boldsymbol{DP}$**.**

$\boldsymbol{QI}$ **:** Quantité d'intensification, où $\boldsymbol{QI} \in [\boldsymbol{0}, \boldsymbol{N_C}]$, elle peut être calculée par $\boldsymbol{QI} = \boldsymbol{N_C} - \boldsymbol{QE}$**.**

L'ACA est un algorithme basé sur une population. La population initiale de $\boldsymbol{N_C}$ solutions (condors des Andes) est représentée par : $Population$, $\boldsymbol{AC} = \boldsymbol{AC_1}, \boldsymbol{AC_2}, \boldsymbol{AC_2}, \dots, \boldsymbol{AC_{Nc}}$**.**

Le mapping utilisant l'ACA nécessite l'initialisation de certains paramètres pour contrôler la procédure itérative de l'algorithme. En tant qu'entrée de l'ACA, on a la population initiale $\boldsymbol{AC}$, $\boldsymbol{N_C}$ le nombre de condors (taille de la population), $\boldsymbol{DP}$ le paramètre de distribution pour l'exploration et l'intensification, et le pourcentage de changement $\boldsymbol{PC}$.

Le coût de chaque $\boldsymbol{AC_i}$ est calculé, et en fonction de cette valeur la population de condors est triée de manière **croissante**. Ensuite, le fitness moyen (le coût moyen) $\boldsymbol{AF}$ de la population est calculé en utilisant l'équation. Cette valeur $\boldsymbol{AF}$ sera utilisée comme indicateur de performance pour déterminer s'il y a une amélioration de la population ou non. Les valeurs des quantités $\boldsymbol{QE}$ et $\boldsymbol{QI}$ sont déterminées en utilisant la valeur initiale du paramètre $\boldsymbol{DP}$. Sur la base de ces quantités, le nombre de condors dans la population initiale est déterminé pour effectuer l'exploration ou l'intensification.

La valeur $\boldsymbol{AF_{new}}$ est calculée et comparée à la fitness moyenne de la population précédente $\boldsymbol{AF}$. Si $\boldsymbol{AF_{new}}$ est supérieure à $\boldsymbol{AF}$, alors le taux de distribution $\boldsymbol{DP}$ est augmenté pour favoriser l'exploration en augmentant la quantité d'exploration $\boldsymbol{QE}$. Cela est réalisé en augmentant le paramètre de distribution $\boldsymbol{DP}$ de $\boldsymbol{PC}$, ce qui augmente la proportion de solutions en statut d'exploration dans la population totale, tandis que la proportion de solutions en statut d'intensification diminue.

D'autre part, si $\boldsymbol{AF_{new}}$ est inférieure à $\boldsymbol{AF}$, alors la quantité d'exploration $\boldsymbol{QE}$ est réduite. Le paramètre de distribution $\boldsymbol{DP}$ est diminué en soustrayant $\boldsymbol{PC}$ ce qui réduit la proportion de solutions en statut d'exploration et augmente la proportion de solutions en statut d'intensification.

Ce processus de génération de nouvelle population et de mise à jour des états est répété jusqu'à atteindre le nombre maximum d'itérations. à chaque itération la meilleur solution $\boldsymbol{AC_b}$ est calculer afin de la comparée avec l'ancien $\boldsymbol{AC_b}$, si la nouvelle $\boldsymbol{AC_b}$ meilleure que l'ancienne en garde cette nouvelle solution, sinon en garde l'ancienne.

6 Conclusion

Ce chapitre a été consacré aux différents approches pour résoudre un problème d'optimisation. Nous avons présenté donc un état de l'art sur des méthodes d'optimisation multiobjectifs avec des concepts tels que la dominance et le front de Pareto. On trouve que 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. De plus, on a présenté une nouvelle technique d'optimisation qui s'appelle optimisation par condor des Andes.

Dans le chapitre suivant, nous allons présenter en détail l'implémentation et l'adaptation de l'algorithme ACA au problème de Mapping et au choix de topologie. Pour cela nous avons choisi d'utiliser la méthode d'optimisation Pareto qui nous a permettra de trouver l'ensemble de solutions de compromis.

Chapitre 4 : Solution Proposée

1 Introduction

Le réseau sur puce (Noc) est largement considéré comme un modèle pour établir une communication efficace au sein des systèmes implémentés sur puce (SoC).

La topologie la plus utilisée par les travaux antérieurs est la topologie Mesh 2 dimension, par conséquent cette topologie a montré des limites dans le cas de l'intégration d'un nombre important des cœurs. Ce problème à encourager les chercheurs à concevoir une nouvelle technologie appelée la technologie 3D. Cependant cette technologie a des limitations concernant la mise en place des TSV.

Le traitement de mapping et la personnalisation des topologies en parallèle est un problème d'optimisation multi- objective. De ce fait, nous avons explorer et adapter l'algorithme Condor des Andes (ACA) qui vise à trouver des solutions non dominées pour résoudre efficacement ce problème (optimisation de coût et de surface).

2 Formulation de problème

Le problème de mapping pour un NoC peut être formulé dans les sections suivants.

2.1 Graphe d'application (APG)

Le graphe d'application (APG) est un graphe orienté $\boldsymbol{G}$ $(\boldsymbol{C}, \boldsymbol{E})$ où $\boldsymbol{C} = \{\boldsymbol{C_1}, \boldsymbol{C_2}, \boldsymbol{C_3}, \dots, \boldsymbol{C_n}\}$ est un ensemble de sommets, représentant les cœurs d'une application et $\boldsymbol{E} = \{\boldsymbol{e_{i,j}} \mid \boldsymbol{i,j} = \boldsymbol{1,2,3,\dots,n}\}$ désigne l'ensemble des bords dirigés, représentant le poids de communication entre deux cœurs adjacents $\boldsymbol{C_i}$ et $\boldsymbol{C_j}$ [97].

2.2 Graphe d'architecture (ARG)

Un graphe d'architecture NoC (ARG) est un graphe de topologie $\boldsymbol{H}$ $(\boldsymbol{U}, \boldsymbol{L})$ où $\boldsymbol{U} = \{\boldsymbol{U_1}, \boldsymbol{U_2}, \boldsymbol{U_3}, \dots, \boldsymbol{U_n}\}$ désigne l'ensemble des nœuds, qui représente les tuiles dans l'architecture NoC connectées aux routeurs locaux, et $\boldsymbol{L} = \{\boldsymbol{l_{i,j}} \mid \boldsymbol{i,j} = \boldsymbol{1,2,3,\dots,n}\}$ est un ensemble de liens physiques (chemin) entre les tuiles $\boldsymbol{U_i}$ et $\boldsymbol{U_j}$, où n est le nombre de routeurs avec des éléments de traitement dans l'architecture NoC [97].

Le mapping d'application est défini par une fonction $\mathbf{F : APG \rightarrow ARG}$ dans laquelle chaque nœud (ou cœur) ($\boldsymbol{C_i} \in \boldsymbol{C}$) d'APG correspond à une tuile appropriée ($\boldsymbol{U_i} \in \boldsymbol{U}$) d'ARG. Le mapping est effectué si et seulement si, le nombre total de cœurs dans APG est inférieur ou égal au nombre total de tuiles dans ARG, c'est-à-dire $|\boldsymbol{C}| \leq |\boldsymbol{U}|$ [97].

La figure suivante représente la façon de mapping d'un graphe d'application sur un graphe d'architecture.

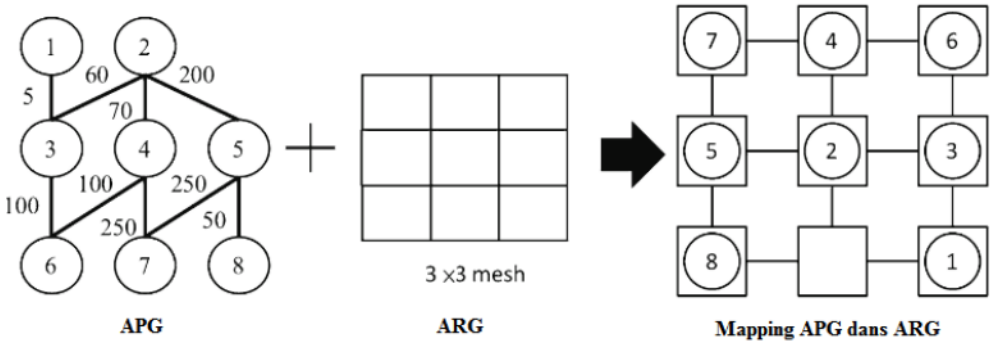


Figure 17 : Exemple d'un mapping

3 Définition de notre fonctions objectives

Notre but consiste à minimiser le coût de communication $\boldsymbol{F(x)}$ entre les différentes Ips d'une application (APG) placé sur une architecture (ARG), et la surface $\boldsymbol{s(x)}$ (minimiser le nombre de TSV) dans une topologie 3D.

Tels que le nombre possible de TSVs est calculé de la manière suivante :

$$\boldsymbol{Max\ (TSV) = \frac{Nombre\ totale\ des\ tuiles}{2}}$$

D'où le nombre de $\boldsymbol{Tsv = [1, Max(Tsv)]}$.

- **La surface $s(x)$:**

  Cette fonction assure l'utilisation d'un nombre minimum de liens verticaux (nombre TSV). Donc on s'intéresse seulement au nombre de TSV (liens, l'espace entre les couches empilées, la taille des routeurs ..., sont ignorées).

$$\boldsymbol{S(x) = min\ \{nombre\ des\ liens\ verticaux\}} \qquad \textit{(1)}$$

- **Le coût de communication $F(x)$:**

$$\boldsymbol{F(x) = min\{\ Coût\ entre\ les\ IPs\}}$$

$$\boldsymbol{Coût = \sum_{i=0}^{E} e_{C_{i,}C_j} * M_{dist}\ (u_i, u_j)} \qquad \textit{(2)}$$

Où $\boldsymbol{E}$: représente le nombre total des arcs dans le graphe d'application.

$\boldsymbol{e_{C_{i,}C_j}}$: représente le volume de communication entre les cœurs $\boldsymbol{C_{i,}C_j}$ dans le graphe d'application.
$\boldsymbol{M_{dist}}$: Représente la distance Manhattan entre deux tuiles $\boldsymbol{u_i, u_j}$ qui correspondent aux cœurs $\boldsymbol{C_{i,}C_j}$ respectivement dans le graphe d'architecture.

$\boldsymbol{M_{dist}(u_i, u_j) = |x_i - x_j| + |y_i - y_j|}$ Si les deux tuiles se trouvent dans le même plan.

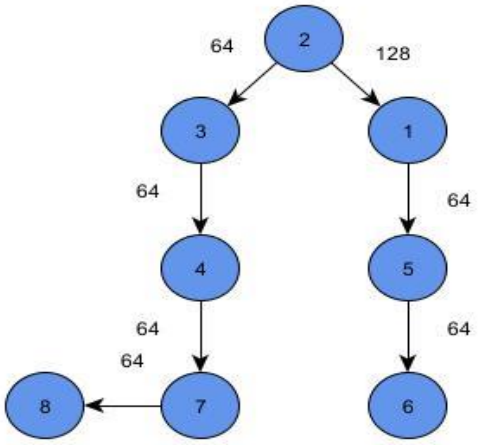
$M_{dist}(u_i, u_j) = M_{dist}(u_i, u_k) + M_{dist}(u_k, u_j) + \alpha$ Si les deux tuiles se trouvent dans différents plans.

Où , u_k : représente un tuile d'un lien vertical (TSV), et α représente le coût d'un lien vertical. S'il existe plusieurs liens verticaux, on va choisi le lien qui donne le minimum nombre de sauts (minimum distance) entre la tuile source et celle de destination.

4 Représentation d'une application

Nous avons représenté une application par une matrice, où chaque case $[i][j]$ représente le volume de communication entre IP_i et IP_j.

Exemple :



	1	2	3	4	5	6	7	8
1					64			
2	128		64					
3				64				
4							64	
5						64		
6								
7								64
8								

Figure 18 : exemple d'un APG

5 Représentation d'une solution

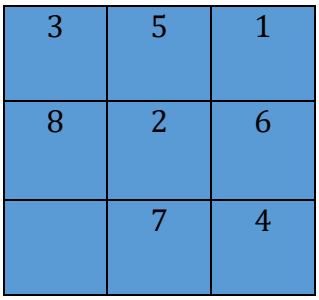
Comme nous avons dit auparavant, la topologie la plus simple et la plus facile de mise en œuvre est la topologie Mesh.

5.1 Cas 2D Mesh

Dans le cas d'un NoC 2D Mesh, une solution peut être représenter par une matrice de taille $m * n$, où m représente le nombre de lignes, et n représente le nombre de colonnes, tel que $m * n \geq$ au nombre des IP_s.

Exemple

Cet exemple représente un mapping de **8** IP_s dans un NoC a 9 tuiles.



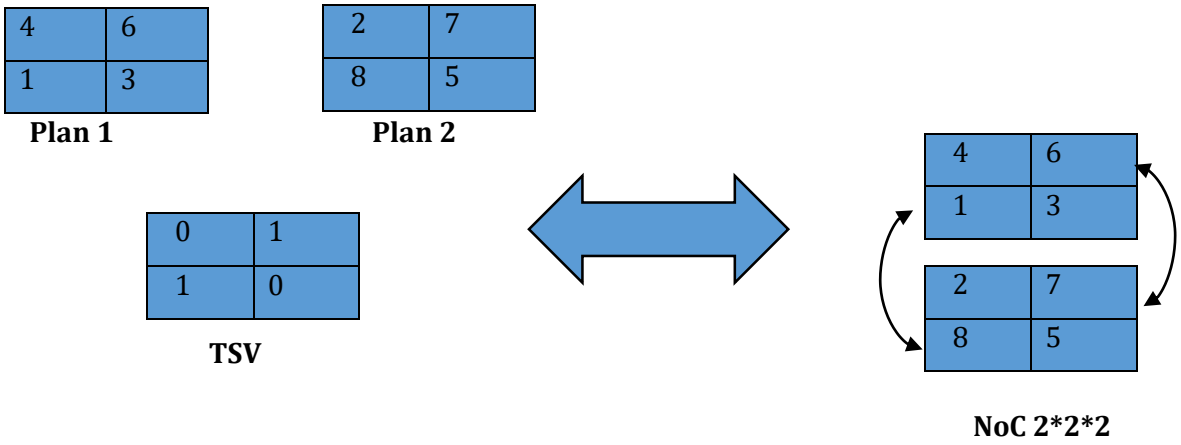
NoC 3 ∗ 3

5.2 Cas 3D Mesh

Nous avons représenté un NoC 3D Mesh par deux plans liées par un ou plusieurs liens verticaux, chaque plan contient la moitié de nombre total d'$\boldsymbol{IP_s}$ d'une application. Un plan peut être représenter par une matrice de taille $\boldsymbol{m * n}$, tel que $\boldsymbol{m * n} \geq$ nombre de $\boldsymbol{IP_s/2}$. Les positions des liens verticaux peuvent être aussi représenter par une matrice de taille $\boldsymbol{m * n}$, les cases qui contient **1** représentent les positions de ces liens et les cases contient **0** indiquent que la tuile n'a pas un lien vertical.

Exemple

Cet exemple représente un mapping de **8** $\boldsymbol{IP_s}$ dans deux plans qui sont liés par deux TSVs.



6 Mapping multi-objectif Utilisant ACA

Comme on a déjà dit, dans la version multiobjectif on a plusieurs critères à optimiser (dans notre cas le coût de communication et le nombre de TSVs), donc on a la possibilité d'avoir un ensemble de solutions de compromis et pas une seule solution.

L'algorithme ACA proposée dans [97] est un algorithme qui a le but de résoudre le problème de mapping mono-objectif. Pour cela nous avons proposé une nouvelle version d'ACA qui permet de résoudre le problème de mapping multi-objectif en optimisant en parallèle le coût de communication et le nombre des liens verticaux.

6.1 Algorithme ACA multi-objectif

Entrés : AC, N_C
Sortie : AC_m ***// meilleur archive***
Début
Initialiser_Population () ;
Trouver_Front_Pareto() ;
Mettre_Ajour_Etat_Condors () ;
Tantque (itr < MaxItr) faire
Pour chaque solution AC_i ***dans population*** AC ***faire***
Si Etat de (AC_i) ***== Exploration alors***
Exploration_Movement (AC_i)***;***
Sinon
Intensification_Movement(AC_i)***;***
Fin Si
Fin Pour
Trouver_Front_Pareto() ;
Mettre_Ajour_Etat_Condors () ;
Mettre à jour l'archive AC_m ***;***
Fin Tantque ;
Fin ;

La procédure **Initialiser_Population** est la même que celle de l'ACA mono-objectif (voir chapitre3).

À chaque itération un ensemble de solutions de compromis est calculer en utilisant la notion de dominance au sens de Pareto selon la procédure **Trouver Front de Pareto**.

Procédure Trouver_Front_Pareto()

Entrée : AC, Nc
Sortie : AC_b // archive Front de Pareto
Début
Pour i allant de 1 à Nc faire
Pour j allant de 1 à Nc faire
Si (coût de AC_j < coût de AC_i ET nombre des TSVs de AC_j <= nombre des TSVs de AC_i)
OU (coût de AC_j <= coût de AC_i ET nombre des TSVs de AC_j < nombre des TSVs de AC_i)
Alors
Supprimer AC_i ;
Fin si
Fin pour
Fin pour
Ajouter les solutions non dominées dans l'archive AC_b ;
Retourner AC_b ;
Fin

Procédure Mettre_Ajour_Etat_Condors () ;

Entrées : AC, N_c , AC_b
Début
Pour i allant de 1 à N_c
Si (AC_i appartient à l'archive AC_b) alors
Etat de AC_i = Intensification ;
Sinon
Etat de AC_i = Exploration;
Fin pour
Fin

Les Procédures **Exploration_Movement** et **Intensification_Movement** sont les mêmes que celles de l'ACA mono-objectif (voir chapitre 3).

6.1.1 Exemple d'un mouvement d'exploration

La figure suivante montre un mouvement d'exploration pour une solution aléatoire.

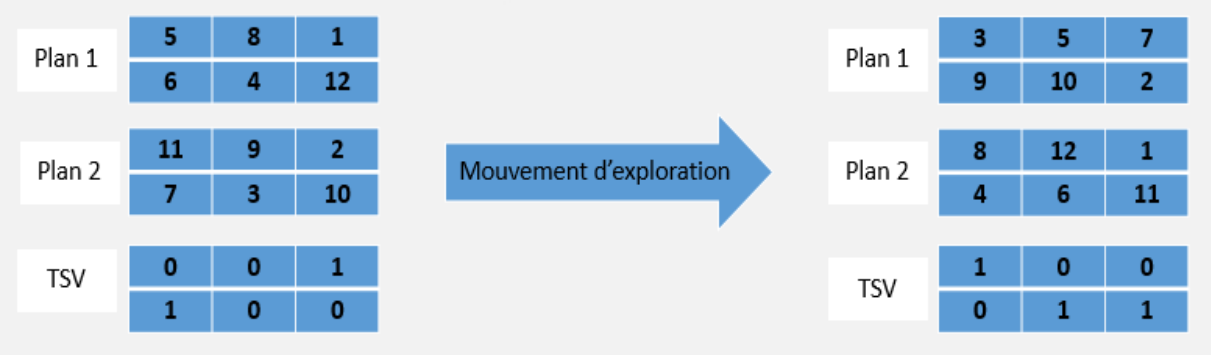


Figure 19 : Mouvement d'exploration

6.1.2 Exemple d'un mouvement d'intensification

La figure suivante montre un mouvement d'intensification pour une solution aléatoire.

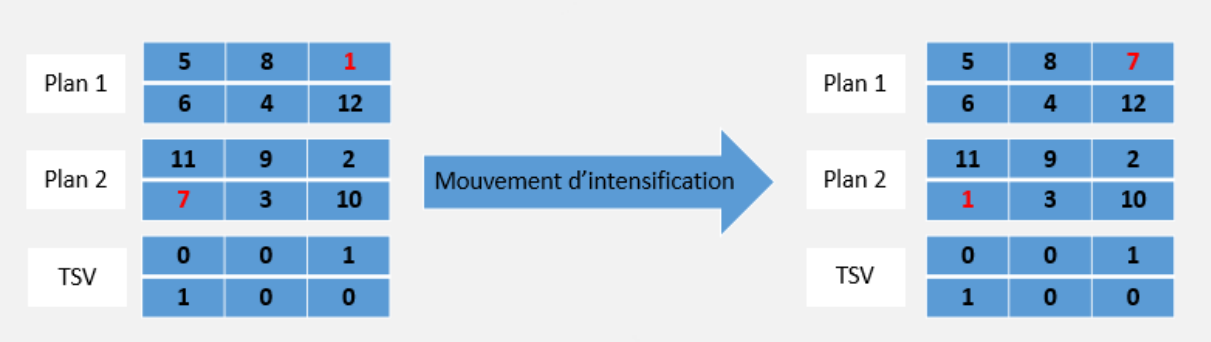


Figure 20 : mouvement d'intensification

À chaque itération on va vérifier s'il existe une solution (S1) dans l'archive $\boldsymbol{AC_b}$ **(front de Pareto)** domine une solution (S2) de la meilleur archive $\boldsymbol{AC_m}$ selon la procédure ***Mettre à jour*** $\boldsymbol{AC_m}$, si oui, on remplace S2 par S1.

Procédure Mettre à jour AC_m ()

```
Entrée : AC_b , AC_m
Sortie : AC_m // Meilleur archive
Début
  Pour chaque solution S1 dans l'archive AC_b
    Pour Chaque solution S2 dans l'archive AC_m
      Si (coût de S1 < coût de S2 ET nombre des TSVs de S1 <= nombre des TSVs de S2)
      OU (coût de S1 <= coût de S2 ET nombre des TSVs de S1 < nombre des TSVs de S2)
      alors
          Remplacer S2 par S1 ;
      Fin si ;
    Fin pour ;
  Fin pour ;
Retourner AC_m ;
Fin
```

Après tous les itérations on obtient une archive $\boldsymbol{AC_m}$ contient que les meilleurs solution trouver par l'algorithme.

7 Conclusion

Ce chapitre a montré la manière que nous avons adaptée à l'algorithme Condor des Andes (ACA) pour résoudre le problème de mapping et la topologie 3D personnalisée. Nous avons commencé par introduire notre problème en formule mathématique, puis on a présenté notre solution.

Dans le chapitre suivant, après l'implémentation de notre algorithme, nous allons présenter l'ensemble des graphes d'application, les tests ainsi que les résultats obtenus pour tester notre solution.

Chapitre 5 : Tests et résultats

1 Introduction

Après avoir présenté notre solution dans le chapitre précédent, nous présentons les différents résultats obtenus lors de l'expérimentation. Nous commençons par présenter les différents benchmarks sur lesquels nous avons effectués nos tests, puis nous étudions les résultats obtenus sur ces benchmarks.

Pour l'implémentation de l'algorithme ACA, Nous avons choisi d'utiliser le langage JAVA sur Eclipse. Les tests sont réalisés sur une machine équipée d'un processeur Intel Core i3 2 GHz et d'une RAM de 4 GO sous le système d'exploitation Windows (10) de 64 bits.

2 Présentation des benchmarks

Les benchmarks sont utilisés pour tester les différentes techniques de mapping implémentées. Dans notre expérimentation on a utilisé 7 benchmarks qui sont PIP, MWD, MPEG, MPEG_4, VOPD et deux aléatoires.

2.1 Benchmark PIP

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

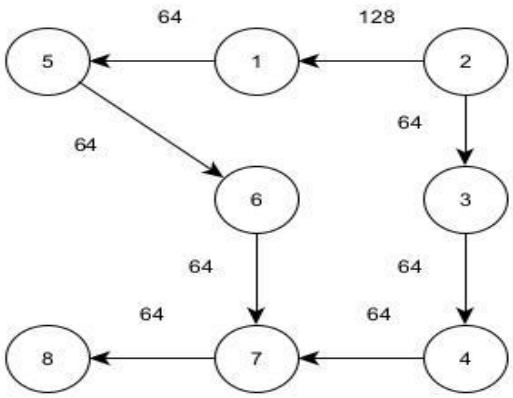


Figure 21 : Benchmark PIP [58]

2.2 Benchmark MWD

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

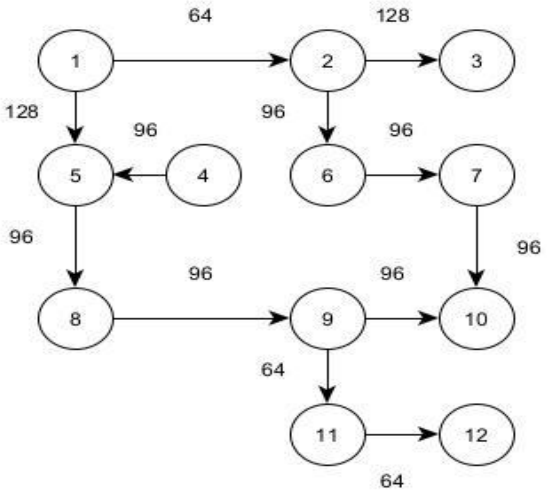


Figure 22 : Benchmark MWD [58]

2.3 Benchmark MPEG

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

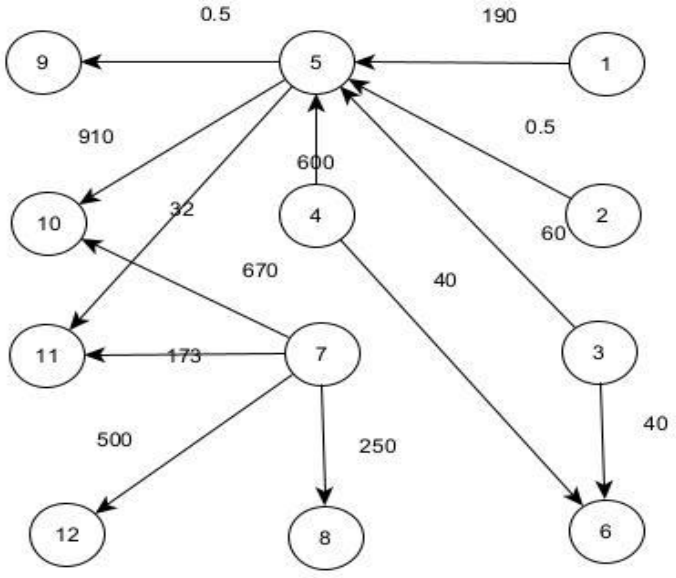


Figure 23 : Benchmark MPEG [58]

2.4 Benchmark MPEG_4

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

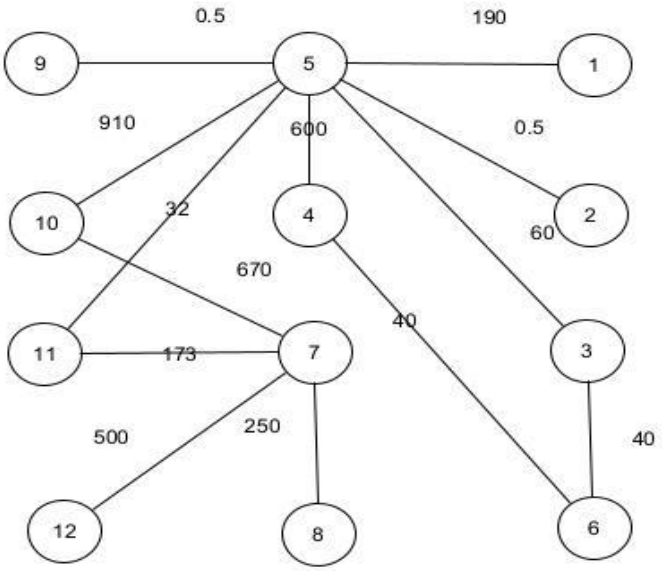


Figure 24 : Benchmark MPEG_4 [58]

2.5 Benchmark VOPD

Le benchmark VOPD possède 16 nœuds et 21 liens, la figure suivante montre le graphe d'application de ce benchmark.

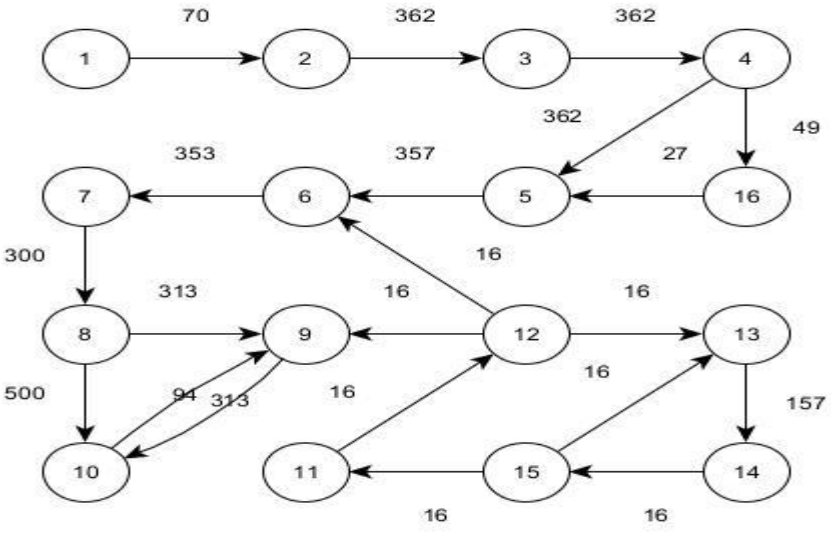


Figure 25 : Benchmark VOPD [58]

2.6 Benchmarks aléatoires

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 (vous trouvez les détails de ces deux benchmarks dans l'annexe A) [101].

3 Architecture des benchmarks

Les applications sont mappées sur des structures de mapping 2D et 3D avec des tailles indiquées dans le tableau 6.

Tableau 5 : Différentes applications et tailles des topologies

Benchmark	***2D MESH***	***3D MESH***
PIP	*2 x 4*	*2 x 2 x 2*
MWD	*3 x 4*	*2 x 3 x 2*
MPEG	*3 x 4*	*2 x 3 x 2*
MPEG_4	*3 x 4*	*2 x 3 x 2*
VOPD	*4 x 4*	*2 x 4 x 2*
40 IP	*5 x 8*	*4 x 5 x 2*
80 IP	*8 x 10*	*5 x 8 x 2*

4 Tests et résultats de l'algorithme ACA

4.1 Résultats de la version mono-objectif

La population initiale a été générée aléatoirement avec une taille égale à 100, nombre d'itérations aussi égal à 100, paramètre de distribution DP = 0.5, le pourcentage de changement PC = 0.1. Le tableau suivant résume les résultats des benchmarks après 10 exécutions avec des topologies 2D MESH et 3D MESH complétement connectée (100% TSVs) et partiellement connectée (25%, 50% de TSVs). Pour un topologies 3D nous avons fixés le coefficient d'un lien vertical à 1 ($\alpha = 1$). Nous avons fixé les valeurs de DP et PC et les pourcentages des TSVs selon la littérature. (Pour consulter les résultats voire Annexe B).

Tableau 6 : Résultats de ACA mono-objectif

	Topologie	***Le coût de communication***	***Temps d'exécution***
PIP	*2D*	*640*	*1.12 s*
	3D 25%	*768*	*2.401 s*
	3D 50%	*640*	*2.451 s*
	3D 100%	*640*	*2.332 s*
MWD	*2D*	*1344*	*1.414 s*
	3D 25%	*1344*	*3.049 s*
	3D 50%	*1312*	*3.904 s*
	3D 100%	*1280*	*3.164 s*
MPEG	*2D*	*3962*	*1.389 s*
	3D 25%	*3897.5*	*2.829 s*
	3D 50%	*3771.5*	*3.429 s*
	3D 100%	*3757*	*2.844 s*
MPEG_4	*2D*	*7516*	*1.486 s*
	3D 25%	*7802*	*3.003 s*
	3D 50%	*7544*	*3.504 s*
	3D 100%	*7093*	*2.925 s*
VOPD	*2D*	*4641*	*1.699 s*
	3D 25%	*5300*	*3.267 s*
	3D 50%	*5124*	*3.641 s*
	3D 100%	*4614*	*3.251 s*
40 IP	*2D*	*8554*	*3.707 s*
	3D 25%	*7628*	*6.831 s*
	3D 50%	*7276*	*6.602 s*
	3D 100%	*7076*	*6.028 s*
80 IP	*2D*	*28146*	*6.84 s*
	3D 25%	*22846*	*12.019 s*
	3D 50%	*22334*	*11.478 s*
	3D 100%	*22256*	*10.488 s*

Discussion

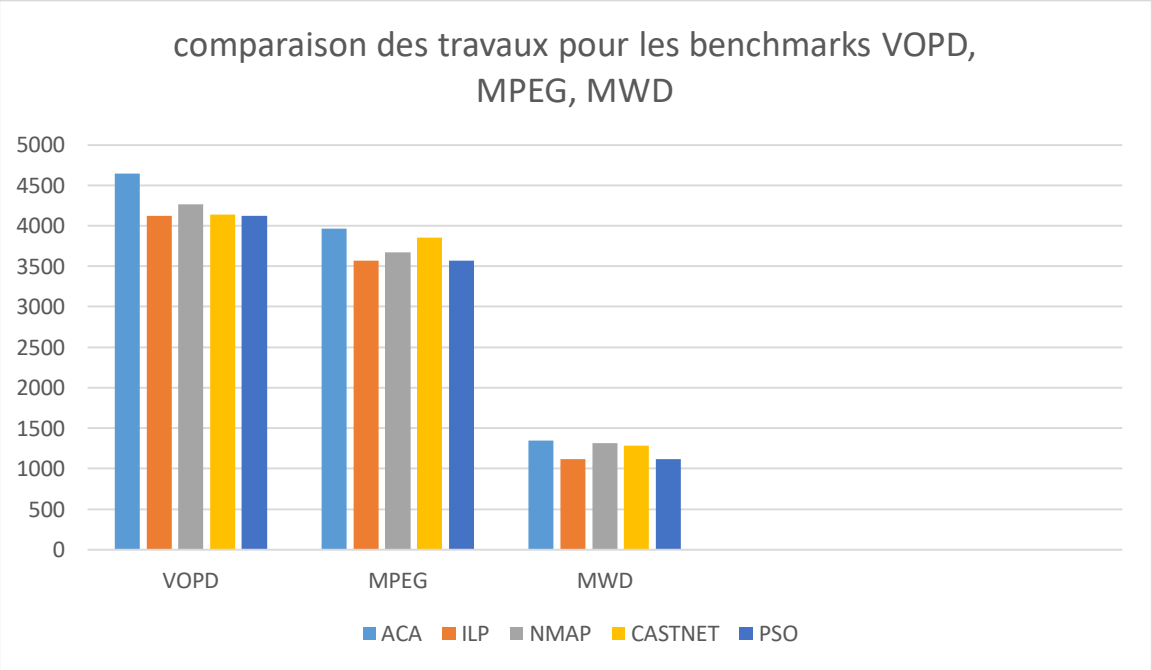
Nous remarquons dans le tableau précédent que le nombre de TSVs influence directement le coût de communication, c'est-à-dire le coût de communication s'améliore quand on passe d'une topologie 3D avec 25% de TSVs vers une topologie 3D avec 100% de TSVs pour tous les benchmarks.

On remarque aussi qu'une topologie 3D complétement connecté et partiellement connecté (50% TSVs) offrent le même coût de communication que la topologie 2D pour le benchmark PIP. Pour les benchmarks MWD, MPEG, MPEG_4 et VOPD, 40 IP et 80 IP une topologie 3D complétement connectée offre toujours un meilleur coût qu'une topologie 2D.

Pour le temps d'exécution, on conclut que plus la taille de l'application est grand, plus le temps de CPU augmente.

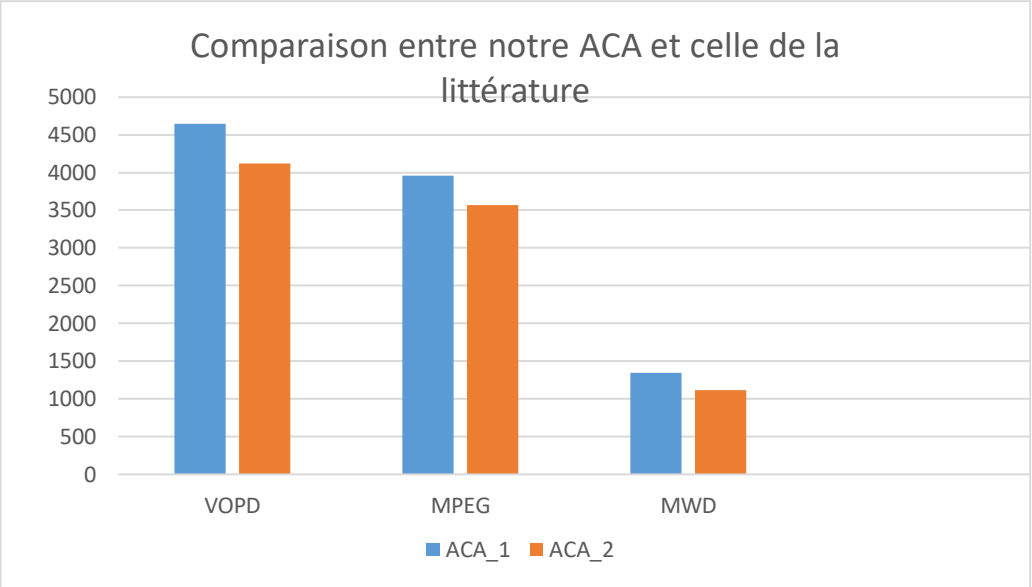
4.1.1 Comparaison avec la littérature

L'histogramme suivant montre une comparaison entre nos résultats (coût de communication) et les résultats d'autres travaux existants dans la littérature pour une topologie 2D Mesh.



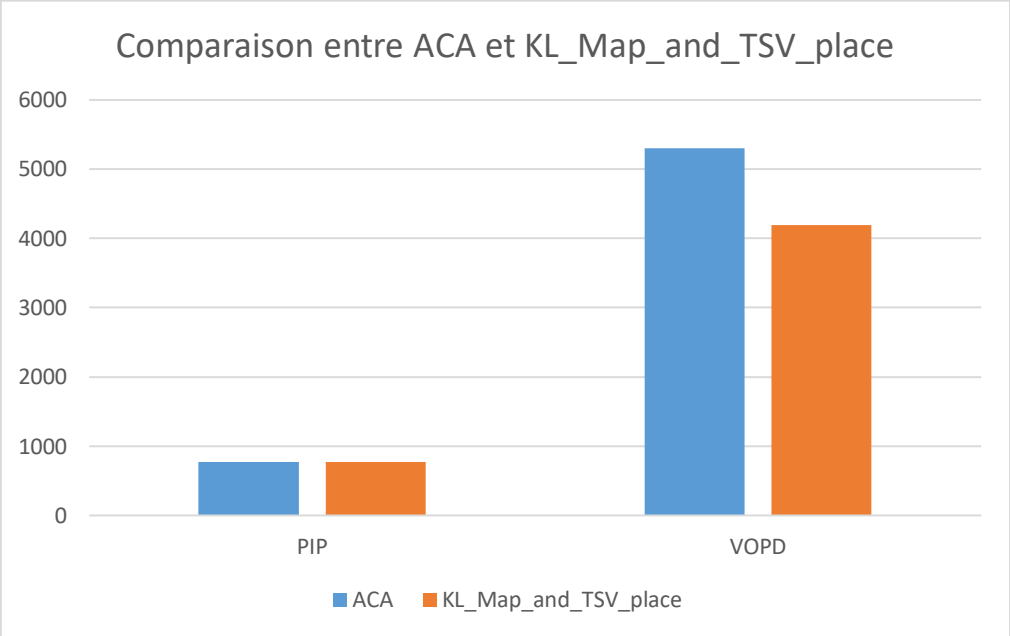
On remarque que nos résultats sont assez faibles par rapport aux résultats des travaux présentées dans [98].

L'histogramme suivant montre une comparaison entre le coût de communication trouver par notre ACA mono-objectif (ACA_1) et le coût de communication trouver dans [97] (ACA_2) pour une topologie 2D Mesh.



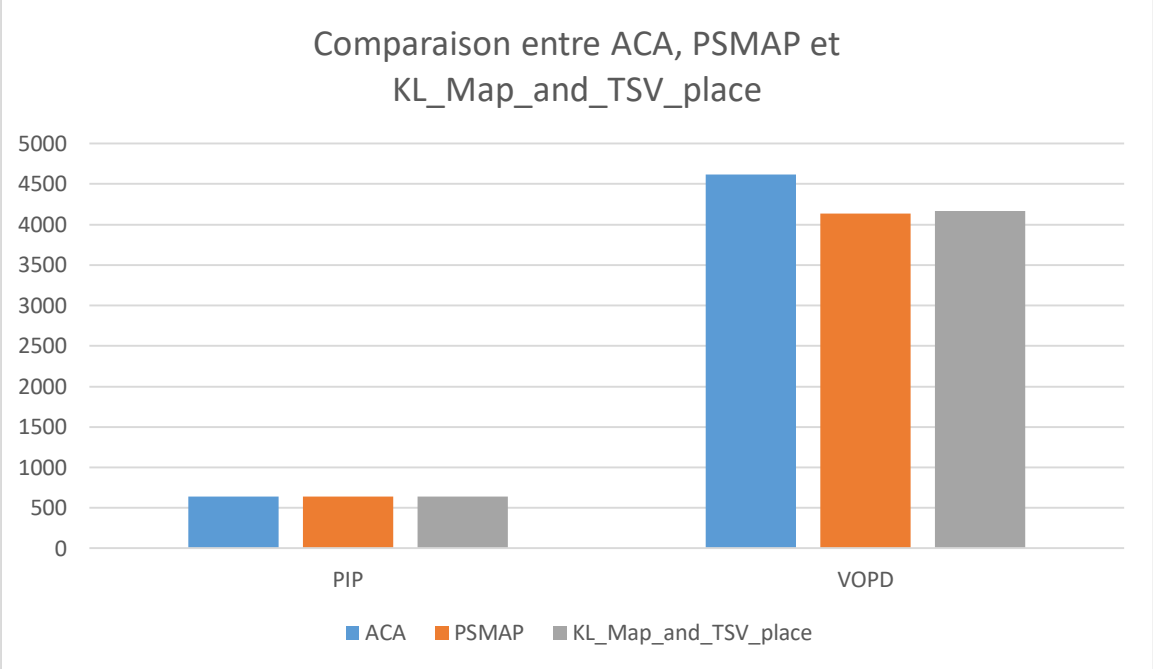
Nos résultats sont faibles que celles d'ACA [97], on remarque que pour le benchmark VOPD, ACA_2 est meilleur que notre ACA (ACA_1), 4119 contre 4641, pour les benchmarks MWD, MPEG ACA_2 est encore meilleur (1120 contre 1344 pour le benchmark MWD, et 3567 contre 3962 pour le benchmark MPEG).

L'histogramme suivant résume une comparaison en terme de coût de communication entre notre algorithme ACA avec KL_Map_and_TSV_place [99] pour une topologie 3D Mesh partiellement connecté (25% de TSVs).



On remarque que nos résultats sont égaux avec les résultats de KL_Map_and_TSV_place pour le benchmark PIP, par contre pour le benchmark VOPD le résultat obtenu par notre ACA est plus faible que celle de KL_Map_and_TSV_place (5300 contre 4189).

L'histogramme suivant résume une comparaison en terme de coût de communication entre ACA et KL_Map_and_TSV_place et PSMAP [100] pour une topologie 3D Mesh complètement connecté.



D'après l'histogramme précédent, on remarque que les résultats des trois travaux (ACA, PSMAP, KL_Map_and_TSV_place) sont égaux pour le benchmark PIP, cependant la méthode PSMAP est meilleur qu'ACA et KL_Map_and_TSV_place (4135 contre 4614 et 4167 respectivement).

4.2 Résultats de la version multi-objectif

Dans le cas multi-objectif, nous avons calculés le coût moyen et le nombre de TSVs moyen des solutions du l'archive. Le tableau suivant résume les résultats obtenus après 10 exécutions en variant la taille de population avec une nombre d'itération égale à 100.

Tableau 7 : Résultats obtenus en variant la taille de population initiale

	Taille de population	***Nombre de solutions***	***Coût moyen*** $(\alpha = 1)$	***Nombre de TSVs moyen***
PIP	*100*	*2*	*704*	*1.5*
	250	*2*	*704*	*1.5*
	500	*2*	*704*	*1.5*
	1000	*2*	*704*	*1.5*
MWD	*100*	*2*	*1744*	*1.5*
	250	*4*	*1560*	*2.75*
	500	*2*	*1504*	*2*
	1000	*3*	*1493.3*	*2.6*
MPEG	*100*	*2*	*4547.75*	*1.5*
	250	*2*	*4042.25*	*1.5*
	500	*3*	*3935.3*	*2.3*
	1000	*2*	*3892*	*2*
MPEG_4	*100*	*2*	*8815*	*2*
	250	*2*	*8295*	*2.5*
	500	*2*	*8015*	*1.5*
	1000	*2*	*7865.5*	*2.5*
VOPD	*100*	*2*	*5864*	*3*
	250	*2*	*5801*	*2.5*
	500	*3*	*5744.3*	*2*
	1000	*2*	*5252*	*2*
40 IP	*100*	*2*	*8847*	*2.5*
	250	*5*	*8718.8*	*3.8*
	500	*5*	*8690.4*	*3.8*
	1000	*3*	*8336.6*	*3.6*
80 IP	*100*	*4*	*27786*	*4.75*
	250	*4*	*27403*	*2.75*

	500	*4*	*26417.5*	*5.5*
	1000	*5*	*26222.8*	*6.2*

Nous remarquons dans le tableau précédent que plus la taille de la population augmente, plus le coût moyen est réduit pour les benchmarks MWD, MPEG, MPEG_4, VOPD, 40IP et 80IP. Autrement dit, plus la taille de la population est grande, plus la chance de trouver des solutions optimales est grande. Cependant, pour le benchmark PIP la taille de population n'influence pas les performances. Dans notre série de test, on prend la taille de population qui égale à 1000 comme une taille idéale.

4.2.1 Effet de variation du nombre d'itération

Le tableau suivant résume les résultats obtenus après 10 exécutions en variant le nombre d'itération avec une taille de population = 1000.

Tableau 8 : Résultats obtenus en variant le nombre d'itération

	Nombre d'itération	***Nombre de solutions***	***Coût moyen*** $(\boldsymbol{\alpha = 1})$	***Nombre de TSVs moyen***
PIP	*100*	*2*	*704*	*1.5*
	250	*2*	*704*	*1.5*
	500	*2*	*704*	*1.5*
	1000	*5*	*742.4*	*1.2*
MWD	*100*	*3*	*1493.3*	*2.6*
	250	*2*	*1456*	*1.5*
	500	*1*	*1504*	*1*
	1000	*2*	*1408*	*2*
MPEG	*100*	*2*	*3892*	*2*
	250	*1*	*3861.5*	*1*
	500	*2*	*4002*	*1.5*
	1000	*2*	*3816.75*	*1.5*
MPEG_4	*100*	*2*	*7865.5*	*2.5*
	250	*2*	*7957*	*1.5*
	500	*2*	*7659.5*	*1.5*
	1000	*2*	*7848.5*	*1.5*
VOPD	*100*	*2*	*5252*	*2*
	250	*2*	*5175*	*3*
	500	*2*	*5141.5*	*2*

	1000	*1*	*4903*	*1*
40 IP	*100*	*3*	*8336.6*	*3.6*
	250	*4*	*8017.5*	*5*
	500	*4*	*7859.5*	*4.5*
	1000	*4*	*7788*	*4.5*
80 IP	*100*	*5*	*26222.8*	*6.2*
	250	*4*	*25628.5*	*7.25*
	500	*4*	*25487.5*	*4.5*
	1000	*3*	*25818*	*2.6*

Discussion

On conclut que les nombres d'itération qui fournissent des bons résultats ne sont pas toujours les mêmes (chaque benchmark a son propre nombre d'itération idéal).

5 Récapitulatif de choix des paramètres

Le tableau 9 résume les meilleurs paramètres trouvés pour les benchmarks PIP, MWD, MPEG, MPEG_4, VOPD, 40IP et 80IP.

Tableau 9 : Récapitulatif pour le choix des paramètres

Benchmark	***Taille de Population***	***Nombre d'itération***
PIP	*1000*	*100*
MWD	*1000*	*1000*
MPEG	*1000*	*1000*
MPEG_4	*1000*	*500*
VOPD	*1000*	*1000*
40 IP	*1000*	*1000*
80 IP	*1000*	*500*

6 Etude comparative

Pour tester l'efficacité de notre algorithme ACA multi-objectif, on a comparé nos résultats avec les résultats de la méthode exacte [101].

Tableau 10 : Comparaison entre ACA et la méthode exacte

	ACA			***Méthode exacte [101]***		
	Nombre de solutions optimales	***coût de communication moyen***	***Nombre de TSVs moyen***	***Nombre de solutions optimales***	***coût de communication moyen***	***Nombre de TSVs moyen***
PIP	*2*	*704*	*1.5 (37.5%)*	*4*	*617.6*	*2.5 (62.5%)*
MWD	*2*	*1408*	*2 (33.33%)*	*4*	*1208*	*2.5 (41.66%)*
MPEG	*2*	*3816.75*	*1.5 (25%)*	*3*	*3385.4*	*2 (33.33%)*
VOPD	*1*	*4903*	*1 (12.5%)*	*6*	*3934.2*	*4 (50%)*

(Suite du Tableau 10)

	ACA	***Méthode exacte***
	Temps de CPU	***Temps de CPU***
PIP	*19.968 s*	*13 s*
MWD	*187.354 s (3.122 min)*	*10 jr*
MPEG	*199.971 s (3.33285 min)*	*8 jr*
VOPD	*272.617 s (4.5436 min)*	*14 jr*

D'après le tableau précédent, on remarque que la méthode exacte offre des meilleurs résultats en terme de coût de communication par rapport à la méthode approché ACA, ceci est logique car les pourcentages des TSVs trouver par la méthode exacte sont supérieures aux pourcentages des TSVs trouver par l'ACA. Cependant le temps de CPU de la méthode exacte prend des jours selon [101] contrairement à l'algorithme ACA qui prend quelques minutes (voir Tableau 10) pour trouver les résultats présenter dans le tableau précédent. Pour le nombres de TSVs, on remarque que les deux méthodes permis de trouver des solutions qui utilisent un pourcentage de TSV en dehors des pourcentages de la littérature (25,50 et 100%).

6.1 Comparaison entre ACA mono-objectif et ACA multi-objectif

Pour comparer les deux versions mono-objectif et multi-objectif, on a fixé les pourcentages des TSVs par (37.5%) pour le benchmark PIP, (33.33%) pour le benchmark MWD, (25%) pour le benchmark MPEG, (12.5%) pour le benchmark VOPD, (25%) pour le benchmark MPEG_4, (22.5%) pour le benchmark (40IP) et (11,25%) pour le benchmark IP80. Nous avons fixé ces valeurs selon le nombre des TSVs trouver par l'ACA multi-objectif. La taille de population et le nombre d'itération sont fixés selon Tableau 9.

Tableau 11 : Comparaison entre ACA mono-objectif et ACA multi-objectif

Benchmarks	***Pourcentage des TSVs***	***Coût de communication trouver par ACA mono-objectif***	***Coût de communication trouver par ACA multi-objectif***
PIP	*37,5 %*	*640*	*704*
MWD	*33.33%*	*1376*	*1408*
MPEG	*25%*	*3752*	*3816.75*
MPEG_4	*25%*	*7562*	*7659.5*
VOPD	*12.5%*	*4892*	*4903*
IP40	*22.5%*	*7204*	*7788*
IP80	*11.25%*	*23952*	*25487.5*

Discussion

On remarque que l'ACA mono-objectif offre des meilleurs résultats que l'ACA multi-objectif, Et c'est logique que les résultats obtenus lors ce qu'on a un seul critère à optimiser soit meilleur que les résultats obtenus lors ce que on a plusieurs critères(contradictoires).

7 Conclusion

Dans ce chapitre, nous avons testés l'algorithme de mapping ACA dans deux architectures 2D et 3D. Pour l'architecture 3D, on a testé les deux version, mono-objectif et multi-objectif. Une série de tests de validation et une étude de paramétrage ont été effectués sur les différents benchmarks pour avoir les meilleurs résultats possibles.

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

Conclusion générale

Les concepteurs intègrent de plus en plus d'unité de traitement dans les systèmes sur puce, pour le but de répondre aux besoins de nouvelles applications et résoudre le problème de communication entre les composantes sur la même surface de silicium.

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. Trouver une combinaison optimale de ces paramètres afin de maximiser les performances d'un NoC est un problème d'optimisation combinatoire.

D'après les recherches effectuées sur les différentes solutions proposées dans la littérature, on a proposé de traiter en parallèle la phase de mapping et celle du choix de topologie.

L'objectif principal du travail présenté dans ce mémoire 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). Pour cela nous avons utilisé l'algorithme ACA (algorithme condor des Andes).

L'optimisation par condor des Andes est inspirée principalement par l'intelligence des condors pour la recherche de nourriture dans les différentes périodes de l'année, ce qui permet de réduire le coût de communication ainsi que le nombre TSVs dans un réseau sur puce.

Enfin, plusieurs expériences ont été réalisées sur différentes applications pour démontrer l'efficacité de l'algorithme utilisé par rapport aux travaux de la littérature.

Perspectives

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 la phase de mapping au reste des phases de conception des réseaux sur puce 3D, tel que le routage et l'ordonnancement.
- Utilisant par exemple un heuristique pour la génération de la population initiale.
- Utilisant par exemple un heuristique pour le mouvement d'exploration ou le mouvement d'intensification.
- Hybridant notre algorithme avec un autre, par exemple ACA avec BA (bat algorithme).
- Explorant d'autres méthodes par exemple le deep learning pour résoudre ce problème.

Bibliographie

[1] G. E. Moore, « Cramming more components onto integrated circuits », vol. 38, nº 8, 1965.

[2] G. J. Ritchie, *Transistor circuit techniques: discrete and integrated*, 3rd ed. Boca Raton Chapman & Hall/CRC, 2003.

[3] Z. B. Amor, « Validation de systèmes sur puce complexes du niveau transactionnel au niveau transfert de registres ». thèse de doctorat , Université de Grenoble, 2014.

[4] J. Hu, Y. Deng, et R. Marculescu, « System-Level Point-to-Point Communication Synthesis Using Floorplanning Information », *th International Conference on VLSI Design*, 2002.

[5] D. Julien, « Méthodologie de modélisation et d'exploration d'architecture de réseaux sur puce appliquée aux télécommunications », INSA de Rennes ,2008.

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

[7] X. T. Tran, « Méthode de Test et Conception en Vue du Test pour les Réseaux sur Puce Asynchrones: Application au Réseau ANOC ». Thèse de Doctorat, Institut National Polytechnique de Grenoble, 2008.

[8] W. Stallings, *Computer organization and architecture: designing for performance*, 8th ed. Upper Saddle River, NJ: Prentice Hall, 2010.

[9] M. Mitic et M. Stojcev, « An overview of on-chip buses », *Facta Univ Electron Energ*, vol. 19, nº 3, p. 405-428, 2006, doi: 10.2298/FUEE0603405M.

[10] N. Toubaline, « AIDE A LA CONCEPTION D'UN RESEAU SUR PUCE POUR UN SYSTEME INTEGRE ». Thèse de doctorat , université saad dahlab de blida, 2018.

[11] S.Kundu . et S. u Chattopadhy , *The Next Generation of System-on-Chip Integration,* Taylor & Francis Group, 2015.

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

[13] W.-C. Tsai, Y.-C. Lan, Y.-H. Hu, et S.-J. Chen, « Networks on Chips: Structure and Design Methodologies », *Journal of Electrical and Computer Engineering*, vol. 2012, p. 1-15, 2012, doi: 10.1155/2012/509465.

[14] A. Charite, « Approches d'optimisation et de personnalisation des réseaux sur puce (NoC : Networks on Chip) » thèse de doctorat, université de technologie Belfort-montbéliard, 2014.

[15] T. Bjerregaard et S. Mahadevan, « A survey of research and practices of Network-on-chip », *ACM Comput. Surv.*, vol. 38, nº 1, p. 1, juin 2006, doi: 10.1145/1132952.1132953.

[16] S. Evain, « Environnement de Conception de Réseau sur puce », INSA de Rennes ,2007.

[17] 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.

[18] J. Quartana, « Conception de réseaux de communication sur puce asynchrones: application aux architectures GALS » thèse de doctorat, Institut National Polytechnique de Grenoble, 2004.

[19] J. A. Bernier, « MÉTHODE DE RECONFIGURATION DYNAMIQUE POUR UN RÉSEAU-SUR-PUCE TOLÉRANT AUX FAUTES ». université de Montréal, 2012.

[20] A. Ben Abdallah, *Advanced Multicore Systems-On-Chip*. Singapore: Springer Singapore, 2017. doi: 10.1007/978-981-10-6092-2.

[21] Michael Opoku Agyeman et al. (2018). Energy and Performance-Aware Application Mapping for Inhomogeneous 3D Networks-on-Chip. Journal of Systems Architecture.

[22] 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.

[23] Konstantinos Tatas, Kostas Siozios, Dimitrios Soudris and Axel Jantsch, "Designing 2D and 3D Network-on-Chip Architectures," Springer Science+Business, 2014.

[24] K. Manna et J. Mathew, *Design and Test Strategies for 2D/3D Integration for NoC-based Multicore Architectures*. Cham: Springer International Publishing, 2020. doi: 10.1007/978-3-030-31310-4.

[25] A. Zia, S. Kannan, G. Rose, et H. J. Chao, « Highly-scalable 3D CLOS NOC for many-core CMPs », in *Proceedings of the 8th IEEE International NEWCAS Conference 2010*, Montreal, QC, Canada: IEEE, juin 2010, p. 229-232. doi: 10.1109/NEWCAS.2010.5603776.

[26] J.-Q. Lu, "3-D hyper integration and packaging technologies for micronano systems," Proc. of the IEEE, vol.97, no.1, Jan.2009.

[27] V. F. Pavlidis et E. G. Friedman, « Physical Design Issues in 3-D Integrated Technologies », 2010, p. 1-21. doi: 10.1007/978-3-642-12267-5_1.

[28] Topol, A.W. et al. (2006). Three-Dimensional Integrated Circuits. IBM J. Research and Development, vol. 50.

[29] Al Attar, S. «Conception et mise au point d'un procédé 3D d'assemblage de puces silicium amincies, empilées et inteérconnectées par des via électriques traversant latéralement les résines polyméres d'enrobage ». 2012,Thèse de doctorat.

[30] Xie, Y. et al. (2010). System-level 3d ic cost analysis and design exploration. in Three Dimensional Integrated Circuit Design, pp. 261–280.

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

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

[33] V. F. Pavlidis et E. G. Friedman, « 3-D Topologies for Networks-on-Chip », *IEEE Trans. VLSI Syst.*, vol. 15, nº 10, p. 1081-1090, oct. 2007, doi: 10.1109/TVLSI.2007.893649.

[34] B. Feero et P. P. Pande, « Performance Evaluation for Three-Dimensional Networks-On-Chip », in *IEEE Computer Society Annual Symposium on VLSI (ISVLSI '07)*, Porto Alegre, Brazil: IEEE, 2007, p. 305-310. doi: 10.1109/ISVLSI.2007.79.

[35] T.-I.-L. Avh et Z. Lu, « Using Wormhole Switching for Networks on Chip: Feasibility Analysis and ». Royal Institute of Technology, Stockholm , 2005.

[36] V. Rantala, T. Lehtonen, et J. Plosila, « Network on Chip Routing Algorithms ». University of Turku, 2006.

[37] C. Nicopoulos, V. Narayanan, et C. R. Das, *Network-on-Chip Architectures*, vol. 45. Dordrecht: Springer Netherlands, 2010. doi: 10.1007/978-90-481-3031-3.

[38] X. Li, « Méthodologie de conception automatique pour multiprocesseur sur puce hétérogène ». Thése de doctorat, Université Paris Sud, 2009.

[39] K. Srinivasan et K. S. Chatha, : « A Genetic Algorithm based Technique for Custom On-Chip Interconnection Network Synthesis », 2005.

[40] U. Y. Ogras et R. Marculescu, « Energy- and Performance-Driven NoC Communication Architecture Synthesis Using a Decomposition Approach », in *Design, Automation and Test in Europe*, Munich, Germany: IEEE, 2005, p. 352-357. doi: 10.1109/DATE.2005.137.

[41] K. Srinivasan, K. S. Chatha, et G. Konjevod, « Linear-programming-based techniques for synthesis of network-on-chip architectures », *IEEE Trans. VLSI Syst.*, vol. 14, nº 4, p. 407-420, avr. 2006, doi: 10.1109/TVLSI.2006.871762.

[42] U. Y. Ogras et R. Marculescu, « Application-specific network-on-chip architecture customization via long-range link insertion », in *ICCAD-2005. IEEE/ACM International Conference on Computer-Aided Design, 2005.*, San Jose, CA: IEEE, 2005, p. 246-253. doi: 10.1109/ICCAD.2005.1560072.

[43] K. S. Chatha, K. Srinivasan, et G. Konjevod, « Automated Techniques for Synthesis of Application-Specific Network-on-Chip Architectures », *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 27, nº 8, p. 1425-1438, août 2008, doi: 10.1109/TCAD.2008.925775.

[44] M. Bahmani, A. Sheibanyrad, F. Petrot, F. Dubois, et P. Durante, « A 3D-NoC Router Implementation Exploiting Vertically-Partially-Connected Topologies », in *2012 IEEE Computer Society Annual Symposium on VLSI*, Amherst, MA, USA: IEEE, août 2012, p. 9-14. doi: 10.1109/ISVLSI.2012.19.

[45] Chia-I Chen; Bau-Cheng Lee; Juinn-Dar Huang; , "Architectural exploration of 3D FPGAs towards a better balance between area and delay," Design, Automation & Test in Europe Conference & Exhibition (DATE), 2011 , vol., no., pp.1-4, 14-18 March 2011.

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

[47] G. Palermo, G. Mariani, C. Silvano, R. Locatelli, et M. Coppola, « Mapping and Topology Customization Approaches for Application-Specific STNoC Designs », in *2007 IEEE International Conf. on Application-specific Systems, Architectures and Processors (ASAP)*, Montreal, QC, Canada: IEEE, juill. 2007, p. 61-68. doi: 10.1109/ASAP.2007.4429959.

[48] M. Bo Stuart et J. Sparso, « Custom topology generation for network-on-chip », in *Norchip 2007*, Aalborg, Denmark: IEEE, nov. 2007, p. 1-4. doi: 10.1109/NORCHP.2007.4481044.

[49] Ogras, U. Y., & Marculescu, R. (2006). " It's a small world after all": NoC performance optimization via long-range link insertion. IEEE Transactions on very large scale integration (VLSI) systems, 14(7), 693-706.

[50] Gulzari, U. A., Anjum, S., Aghaa, S., Khan, S., & Sill Torres, F. (2017). Efficient and scalable cross-by-pass-mesh topology for networks-on-chip. IET Computers & Digital Techniques, 11(4), 140-148.

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

[52] P. Iff, M. Besta, M. Cavalcante, T. Fischer, L. Benini, et T. Hoefler, « Sparse Hamming Graph: A Customizable Network-on-Chip Topology ». 11 janvier 2023. Consulté le: 7 avril 2023. Disponible sur: http://arxiv.org/abs/2211.13980.

[53] Srinivasan, Krishnan, and Karam S. Chatha. "ISIS: a genetic algorithm based technique for custom on-chip interconnection network synthesis." 18th International Conference on VLSI Design held jointly with 4th International Conference on Embedded Systems Design. IEEE, 2005.

[54] 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.

[55] M. R. Tavares et al., "Multi-Objective Mapping of NoC-Based MPSoCs Using a Tabu Search Algorithm with Topology Optimization", Journal of Systems Architecture, vol. 72, pp. 33-47, 2016.

[56] F. André et J.-L. Pazat, « Le placement de taches sur des architectures parallèles ».thése de doctorat, 2006.

[57] E. Carvalho, C. Marcon, N. Calazans, et F. Moraes, « Evaluation of Static and Dynamic Task Mapping Algorithms in NoC-Based MPSoCs », 2009.

[58] T. Nesrine, B. Djamel, et M. Ali, « A Classification and Evaluation Framework for NoC Mapping Strategies », *J CIRCUIT SYST COMP*, vol. 26, nº 02, p. 1730001, févr. 2017, doi :10.1142/S021812661730001X.

[59] Delorme Xavier, « Optimisation combinatoire et problème de capacité d’infrastructure ferroviaire », thèse de doctorat, université de valencienne, 2003.

[60] P. K. Sahu et S. Chattopadhyay, « A survey on application mapping strategies for Network-on-Chip design », *Journal of Systems Architecture*, vol. 59, nº 1, p. 60-76, janv. 2013, doi: 10.1016/j.sysarc.2012.10.004.

[61] S. Murali et G. De Micheli, « Bandwidth-constrained mapping of cores onto NoC architectures », in *Proceedings Design, Automation and Test in Europe Conference and Exhibition*, Paris, France: IEEE Comput. Soc, 2004, p. 896-901. doi: 10.1109/DATE.2004.1269002.

[62] N. Koziris, M. Romesis, P. Tsanakas, et G. Papakonstantinou, « An efficient algorithm for the physical mapping of clustered task graphs onto multiprocessor architectures », in *Proceedings 8th Euromicro Workshop on Parallel and Distributed Processing*, Rhodos, Greece: IEEE Comput. Soc, 1999, p. 406-413. doi: 10.1109/EMPDP.2000.823437.

[63] 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.

[64] Z. Lu, L. Xia, et A. Jantsch, « Cluster based Simulated annealing for maping cores onto 2D mesh networks on chip ».2008.

[65] A. Mehran, S. Saeidi, A. Khademzadeh, et A. Afzali-Kusha, « Spiral: A heuristic mapping algorithm for network on chip », *IEICE Electron. Express*, vol. 4, nº 15, p. 478-484, 2007, doi: 10.1587/elex.4.478.

[66] Alagarsamy, A., Gopalakrishnan, L., Mahilmaran, S., & Ko, S. « A Self-Adaptive Mapping Approach for Network on Chip With Low Power Consumption ». IEEE Access, 7, 84066–84081, 2019 https://doi.org/10.1109/ACCESS.2019.2925381.

[67] S. Sikandar, N. K. Baloch, F. Hussain, W. Amin, Y. B. Zikria, et H. Yu, « An Optimized Nature-Inspired Metaheuristic Algorithm for Application Mapping in 2D-NoC », *Sensors*, vol. 21, nº 15, p. 5102, juill. 2021, doi: 10.3390/s21155102.

[68] Tavanpour, M.; Khademzadeh, A.; Janidarmian, M. Chain-Mapping for mesh based Network-on-Chip architecture. IEICE Electron. Express 2009, 6, 1535–1541.

[69] C. E. Rhee, H. Y. Jeong, and S. Ha, ''Many-to-many core-switch mapping in 2-D mesh NoC architectures,'' in Proc. IEEE Int. Conf. Comput. Design, VLSI Comput. Processors (ICCD), Oct. 2004, pp. 438–443.

[70] A. Aravindhan, S. Salini, and G. Lakshminarayanan, ''Cluster based application mapping strategy for 2D NoC,'' Procedia Technol., vol. 25, pp. 505–512, 2016.

[71] F. Moein-darbari, A. Khademzade, and G. Gharooni-fard, ''CGMAP: A new approach to Network-on-Chip mapping problem,'' IEICE Electron. Exp., vol. 6, no. 1, pp. 27–34, 2009.

[72] P. K. Sahu, P. Venkatesh, S. Gollapalli, and S. Chattopadhyay, ''Application mapping onto mesh structured network-on-chip using particle swarm optimization,'' in Proc. IEEE Comput. Soc. Annu. Symp. VLSI, Jul. 2011, pp. 335–336.

[73] J. Wang, Y. Li, S. Chai, and Q. Peng, ''Bandwidth-aware application mapping for NoC-based MPSoCs,'' J. Comput. Inf. Syst., vol. 7, no. 1, pp. 152–159, 2011.

[74] C.-H. Cheng and W.-M. Chen, ''Application mapping onto mesh-based network-on-chip using constructive heuristic algorithms,'' J. Supercomput., vol. 72, no. 11, pp. 4365–4378, Nov. 2016.

[75] S. Tosun, ‘‘New heuristic algorithms for energy aware application mapping and routing on mesh-based NoCs,’’ J. Syst. Archit., vol. 57, no. 1, pp. 69–78, Jan. 2011.

[76] 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.

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

[78] J. Li *et al.*, « Bat Algorithm Based Low Power Mapping Methods for 3D Network-on-Chips », in *Theoretical Computer Science*, D. Du, L. Li, E. Zhu, et K. He, Éd., in Communications in Computer and Information Science, vol. 768. Singapore: Springer Singapore, 2017, p. 277-295. doi: 10.1007/978-981-10-6893-5_21.

[79] N. Marco, A. Désidéri, S.Lanteri « Multi-objective optimisation in CFD by Genetique Algorithme ». Rapport de recherche 3686, Avril 1999.

[80] M SAMIR « Optimisation Multiobjectif Par Un Nouveau Schéma De Coopération Méta/Exacte » thèse de Magister, 2006.

[81] N Piegay « Optimisation multiobjectif et aide à la décision pour la conception robuste :Application à une structure industrielle sur fondations superficielles » thèse de Doctorat, 2015.

[82] M. M. Karima et D. S. Meshoul, « L’optimisation multiobjectif et l’informatique quantique » thèse de doctorat.

[83] A Berro « Optimisation Multiobjectif et Stratégie d’évolution en environnement Dynamique » thèse de Doctorat, l’université des sciences sociales Toulouse 2001.

[84] V. Pareto. Cours d’économie politique. Rouge, Lausanne, 1896.

[85] E. Talbi « Métaheuristiques pour l’optimisation combinatoire multi-objectif : Etat de l’art » Lille : LIFL, 2001.

[86] O. Mouelhi « Contribution À L’optimisation Multiobjectif En conception multidisciplinaire»,

thèse de doctorat, 2013.

[87] I. Bousaid « PERFECTIONNEMENT DE MÉTAHEURISTIQUES POUR L’OPTIMISATION CONTINUE ». thèse de doctorat, 2013.

[88] K. M. Miettinen « Nonlinear multiobjective optimization, editions Kluwer academic publisher » 1999.

[89] R. L. Carraway, T. L. Morin, and H. Moskowitz « Generalized dynamic programming for multicriteria optimization » European Journal of Operational Research.

[90] D. J. White « The set of e cient solutions for multiple-objectives shortest path problems » Computers and Operations Research.

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

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

[93] 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.

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

[95] 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.

[96] Pavez, E.F., 2014. Patrón de movimiento de dos cóndores andinos Vultur gryphus (Aves: Cathartidae) en los andes centrales de Chile y Argentina. Bol. Chil. Ornitol. 20 (1–2), 1–12.

[97] F. Mehmood *et al.*, « An efficient and cost effective application mapping for network-on-chip using Andean condor algorithm », *Journal of Network and Computer Applications*, vol. 200, p. 103319, avr. 2022, doi: 10.1016/j.jnca.2021.103319.

[98] W. Amin *et al.*, « Performance Evaluation of Application Mapping Approaches for Network-on-Chip Designs », *IEEE Access*, vol. 8, p. 63607-63631, 2020, doi: 10.1109/ACCESS.2020.2982675.

[99] K. Manna, V. S. S. Teja, S. Chattopadhyay, et I. Sengupta, « TSV Placement and Core Mapping for 3D Mesh Based Network-on-Chip Design Using Extended Kernighan-Lin Partitioning », in *2015 IEEE Computer Society Annual Symposium on VLSI*, Montpellier, France: IEEE, juill. 2015, p. 392-397. doi: 10.1109/ISVLSI.2015.9.

[100] P. K. Sahu, P. Venkatesh, S. Gollapalli, and S. Chattopadhyay, « Application mapping onto mesh structured network-on-chip using particle swarm optimization » in IEEE ISVLSI, 2011, pp. 335–336.

[101] Abbad. S, Taileb.S « Optimisation multiobjectif des performances des réseaux sur puce », mémoire de Master, Université Saad Dahleb Blida-1, 2020.

Annexe A : Benchmarks aléatoires

L'application 40 IP

Cette application contient 40 IPs et 40 liens, l'image suivante 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

IPx	IPy	Coût
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

L’application 80 IP

Cette application contient 80 IPs et 81 liens l’image suivante 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 obtenus

Résultats d'ACA Multi-objectif

```
Plan1 :6,2,5,1,Plan2 :4,3,7,8,TSV :1,1,1,0,le cout = 640.0
```

6	2
5	1

4	3
7	8

Plan 1 plan 2

1	1
1	0

TSV

PIP

```
Plan1 :3,2,4,1,Plan2 :7,6,8,5,TSV :1,0,0,1,le cout = 640.0
Plan1 :6,7,5,8,Plan2 :1,4,2,3,TSV :0,1,0,0,le cout = 768.0
le cout moyen = 704.0
le nombre de TSVs moyen = 1.5
```

MWD

```
Plan1 :6,2,4,7,1,5,Plan2 :3,11,12,10,9,8,TSV :1,0,0,1,0,1,le cout = 1248.0
Plan1 :12,11,4,10,9,8,Plan2 :2,3,1,6,7,5,TSV :0,0,0,0,0,1,le cout = 1568.0
le cout moyen = 1408.0
le nombre de TSVs moyen = 2.0
```

MPEG

```
Plan1 :6,3,2,4,5,9,Plan2 :12,7,8,11,10,1,TSV :0,0,0,0,1,0,le cout = 3861.5
Plan1 :9,10,2,8,7,12,Plan2 :4,5,6,11,1,3,TSV :0,1,0,0,1,0,le cout = 3772.0
le cout moyen = 3816.75
le nombre de TSVs moyen = 1.5
```

MPEG_4

```
Plan1 :2,9,1,3,6,4,Plan2 :12,7,10,8,11,5,TSV :0,0,0,0,0,1,le cout = 8057.0
Plan1 :4,5,1,6,10,3,Plan2 :9,12,2,11,7,8,TSV :0,1,0,0,1,0,le cout = 7262.0
le cout moyen = 7659.5
le nombre de TSVs moyen = 1.5
```

VOPD

```
Plan1 :10,9,6,5,8,11,7,4,Plan2 :14,13,1,15,16,12,2,3,TSV :0,0,0,0,0,0,0,1,le cout = 4903.0
le cout moyen = 4903.0
le nombre de TSVs moyen = 1.0
```

Acti

40 IP

```
Plan1 :22,35,29,15,40,1,7,20,8,31,2,5,25,3,39,16,19,28,23,33,Plan2 :32,34,14,26,21,11,36,6,17,10,37,38,4,9,18,12,24,30,27,13,
Plan1 :8,15,25,3,26,18,1,21,36,34,37,20,29,38,33,10,7,4,17,22,Plan2 :12,39,40,13,35,14,28,16,23,9,2,5,32,24,27,19,31,6,30,11,
Plan1 :14,34,33,39,7,19,17,35,5,36,32,30,22,2,10,1,29,12,38,20,Plan2 :16,28,37,31,4,23,3,13,27,24,26,11,9,15,21,8,25,6,40,18,
Plan1 :25,18,21,26,33,8,31,1,32,27,20,5,28,2,4,14,38,30,36,22,Plan2 :35,16,3,40,24,13,23,6,29,39,10,12,11,15,34,7,9,17,37,19,
```

```
,TSV :0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,1,0,le cout = 7788.0
,TSV :0,0,0,1,0,0,1,0,0,1,0,1,1,1,1,0,1,0,0,0,le cout = 7214.0
,TSV :0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,le cout = 8508.0
,TSV :0,0,1,1,1,0,0,0,1,1,0,0,1,1,0,0,0,0,0,le cout = 7642.0

le cout moyen = 7788.0
le nombre de TSVs moyen = 4.5
```

80 IP

```
Plan1 :67,16,23,63,9,12,39,50,14,73,36,13,28,7,30,47,17,1,77,41,24,10,33,80,58,62,65,29,38,76,74,52,61,3,25,59,46,69,31,5,
Plan1 :6,67,31,40,41,20,66,45,24,11,9,32,74,51,2,1,18,61,25,33,10,47,54,4,60,29,64,39,80,22,77,44,58,37,69,71,8,23,12,19,
Plan1 :3,13,62,39,41,11,47,70,7,12,64,71,44,40,43,21,51,14,75,1,56,49,17,42,67,66,46,5,65,78,45,76,28,34,72,20,4,18,32,23,
Plan1 :34,32,21,10,72,31,60,7,68,58,5,45,4,15,50,8,73,49,22,35,12,18,27,23,80,25,51,77,69,26,67,3,66,24,2,52,38,30,57,54,
```

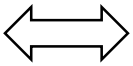
```
,Plan2 :4,53,26,20,37,32,34,68,8,43,72,40,11,22,48,18,35,15,51,27,56,44,21,75,55,42,2,78,6,66,45,79,64,54,49,19,57,70,71,60,
Plan2 :21,34,75,78,36,38,65,72,27,49,28,76,52,62,5,48,3,63,53,43,42,68,7,59,79,57,30,50,15,70,26,55,13,46,16,14,17,56,73,35,
,Plan2 :22,19,55,29,16,38,68,52,61,58,59,36,2,80,53,30,6,9,60,50,31,73,79,25,10,69,48,54,57,15,8,74,27,24,35,77,26,63,33,37,
Plan2 :36,61,20,9,63,41,16,13,28,43,65,40,55,29,19,56,46,48,53,39,76,1,47,78,75,71,79,33,42,70,74,64,44,59,11,6,17,62,14,37,
```

```
TSV :0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,le cout = 28538.0
TSV :1,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,le cout = 23850.0
TSV :0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,le cout = 26284.0
TSV :1,0,0,0,0,0,0,1,0,0,0,0,0,1,0,0,0,0,1,1,0,0,0,0,0,0,1,1,0,1,0,0,0,0,0,0,0,0,0,0,le cout = 23278.0

le cout moyen = 25487.5
le nombre de TSVs moyen = 4.5
```

Résultats d'ACA mono-objectif

Exemple pour le cas 2D Mesh :

```
Solution
8,7,4,3,5,6,1,2,

le cout = 640.0
```



8	7	4	3
5	6	1	2

PIP 2D

```
Solution
8,7,4,3,5,6,1,2,

le cout = 640.0
```

PIP 25% TSV

```
Plan1
2,6,1,5,
Plan2
4,3,7,8,
TSV
1,0,0,0,

le cout = 768.0
```

PIP 50% TSV

```
Plan1
3,2,4,1,
Plan2
6,5,7,8,
TSV
0,1,1,0,

le cout = 640.0
```

PIP 100% TSV

```
Plan1
8,6,1,5,
Plan2
7,4,2,3,
TSV
1,1,1,1,le cout = 640.0
```

MWD 2D

```
Solution
3,2,12,6,4,1,11,7,5,8,9,10,

le cout = 1344.0
```

MWD 25% TSV

```
Plan1
10,8,9,4,5,1,
Plan2
7,6,12,3,2,11,
TSV
1,0,0,0,0,1,

le cout = 1344.0
```

MWD 50% TSV

```
Plan1
3,6,7,2,1,5,
Plan2
11,9,10,12,8,4,
TSV
0,0,1,1,0,1,

le cout = 1312.0
```

MWD 100% TSV

```
Plan1
7,11,12,10,9,8,
Plan2
6,2,3,4,5,1,
TSV
1,1,1,1,1,1,

le cout = 1280.0
```

MPEG 2D

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

le cout = 3962.0
```

MPEG 25% TSV

```
Plan1
12,8,2,6,4,9,
Plan2
7,10,3,11,5,1,
TSV
1,0,0,0,1,0,

le cout = 3897.5
```

MPEG 50% TSV

```
Plan1
2,5,4,3,1,6,
Plan2
9,10,11,8,7,12,
TSV
1,1,1,0,0,0,

le cout = 3771.5
```

MPEG 100% TSV

```
Plan1
6,5,4,2,10,1,
Plan2
3,11,9,12,7,8,
TSV
1,1,1,1,1,1,

le cout de best condor est : 3757.0
itération :15
```

MPEG_4_2D

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

le cout = 7516.0
```

MPEG_4 _25%TSV

```
Plan1
11,7,8,1,10,3,
Plan2
9,12,2,4,5,6,
TSV
0,1,0,0,1,0,

le cout = 7802.0
```

MPEG_4 _50% TSV

```
Plan1
11,8,7,4,5,10,
Plan2
6,3,12,9,1,2,
TSV
0,0,1,1,1,0,

le cout = 7544.0
```

MPEG_4 _100% TSV

```
Plan1
14,3,15,1,5,4,
Plan2
16,13,12,2,9,7,
TSV
1,1,1,1,1,1,

le cout = 7093.0
```

VOPD 2D

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

le cout = 4641.0
```

VOPD 25% TSV

```
Plan1
3,4,1,11,2,5,6,15,
Plan2
8,10,13,12,7,9,16,14,
TSV
0,0,0,1,0,1,0,0,

le cout = 5300.0
```

VOPD 50% TSV

```
Plan1
2,10,8,9,3,1,16,14,
Plan2
5,7,12,15,4,6,11,13,
TSV
0,0,1,1,1,0,0,1,

le cout = 5124.0
```

VOPD 100% TSV

```
Plan1
10,7,6,5,14,13,1,2,
Plan2
9,8,15,4,16,12,11,3,
TSV
1,1,1,1,1,1,1,1,

le cout = 4614.0
```

40IP 2D

```
19,39,8,24,11,9,15,4,28,22,30,25,36,27,37,6,23,33,38,14,21,34,18,12,1,7,5,17,2,31,40,32,16,29,10,35,20,3,13,26,

le cout = 8554.0
```

40IP 25% TSV

```
Plan1
30,29,31,37,26,14,17,10,35,19,2,22,11,33,21,23,3,7,28,12,
Plan2
39,24,1,16,32,36,34,27,18,40,38,4,6,5,8,13,25,20,9,15,
TSV
1,1,0,0,0,0,0,0,0,1,0,0,1,0,0,1,0,0,0,0,

le cout = 7628.0
```

40IP 50% TSV

```
Plan1
40,11,20,7,32,10,31,35,33,12,15,17,4,29,26,8,14,23,25,5,
Plan2
3,37,13,24,18,39,9,6,16,1,30,22,2,21,27,38,36,34,19,28,
TSV
0,0,0,0,1,0,1,0,1,0,1,0,1,0,0,1,1,1,1,1,

le cout = 7276.0
```

40IP 100% TSV

```
Plan1
24,34,9,23,37,39,21,13,3,14,38,36,18,7,1,29,32,27,33,20,
Plan2
35,28,12,15,25,26,11,17,8,5,16,4,31,10,2,30,6,19,40,22,
TSV
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,

le cout = 7076.0
```

80IP 25% TSV

```
Plan1
19,28,59,29,8,40,11,17,12,62,49,41,60,54,57,37,50,77,10,7,35,6,31,65,75,68,23,66,27,32,48,5,26,3,42,39,69,55,4,67,
Plan2
56,34,79,71,70,30,25,24,36,80,9,72,51,20,47,38,1,74,13,18,15,58,22,64,76,78,45,2,61,46,33,21,73,14,53,43,44,63,52,16,
TSV
0,0,0,0,0,0,0,0,1,0,1,0,1,0,0,1,0,0,0,0,1,0,0,1,0,1,1,0,0,0,0,1,0,0,0,0,1,0,0,0,

le cout = 22846.0
```

80IP 50% TSV

```
Plan1
73,8,15,26,2,12,23,31,11,37,68,69,5,62,70,33,53,60,43,61,75,16,45,3,36,52,38,48,9,59,63,79,54,29,6,18,27,71,77,13,
Plan2
51,64,22,35,32,17,25,41,72,34,76,56,19,30,78,40,47,80,4,66,58,44,14,24,67,49,21,74,20,1,7,39,65,46,28,57,50,55,42,10,
TSV
1,1,1,1,0,1,0,0,0,0,0,0,0,0,1,1,0,0,0,1,0,1,0,1,0,1,1,1,0,0,1,0,1,0,1,0,1,1,1,1,

le cout = 22334.0
```

80IP 100% TSV

```
Plan1
8,50,15,25,56,39,12,58,63,3,13,41,30,7,26,38,16,28,6,20,23,74,80,52,9,60,67,10,36,53,76,44,29,55,64,48,59,75,24,47,
Plan2
51,37,77,18,35,17,31,40,19,72,21,46,33,54,5,61,14,2,42,11,4,79,32,66,71,45,62,27,49,34,69,65,70,22,1,73,57,68,43,78,
TSV
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,

le cout = 22256.0
```