République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université SAAD DAHLAB – 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

Génération de topologies personnalisées pour les réseaux sur puce dédiés aux applications spécifiques

Réalisé Par :

BOURENANE Menad

DJOUDI Youcef

Soutenu devant un jury constitué de :

- M. OULD KHAOUA Mouhamed	Président Jury
- M. KAMECHE Abdellah	Examinateur
- Mme. TOUBALINE Nesrine	Promotrice

Année universitaire: 2018/2019

Résumé

Le développement des technologies des circuits intégrés permet l'intégration de plusieurs composants sur une même puce. Ces composants communiquent entre eux à travers des interconnexions de communications (interconnexions point à point ou bus). Cependant, ces derniers trouvent leurs limites en communication. Un nouveau paradigme d'interconnexion est alors apparu (le réseau sur puce - Network on Chip) afin de surmonter les problèmes des méthodes d'interconnexion classiques. Plusieurs problèmes peuvent surgir lors de leur conception du fait du nombre de paramètres qui sont à prendre en compte. Citons le coût en communication, le coût en surface de silicium des composants, le choix de la topologie du réseau, etc.

On s'intéresse dans notre travail d'une part à la phase de mapping. Cette dernière permet de placer les composants qui exécutent une application sur la topologie du réseau sur puce. Le but est de proposer un placement des composants le plus optimale possible, en minimisant le coût de communications. Et d'autre part, à la génération des topologies personnalisées pour le réseau sur puce dans le contexte d'applications spécifiques (une topologie sur mesure permet de réduire les coûts en communication et en surface).

Dans notre travail nous avons étudié l'hybridation du problème de mapping avec le problème de personnalisation de la topologie en considérant la technologie 3D.

Mots clés : système sur puce, réseau sur puce, application spécifique, optimisation, mapping, topologie personnalisée et technologie d'intégration 3D (TSV).

Abstract

The development of the integrated circuit technologies allows the integration of several components on the same chip. These components communicate with each other through interconnections like point-to-point or bus. However, these last interconnections find their limits in term of communication. A new interconnection has emerged called Network on Chip in order to overcome the problems of the traditional interconnections methods. Several problems can arise during the design of the new paradigm due to the number of parameters that must be taken into account (the cost in communication, the silicon surface cost of the components, the choice of the network topology, etc.).

So, we are interested in our work on two part. The first one is making the most optimal component placement possible of an application that execute on a network-on-chip topology, by minimizing the cost of communications in the mapping phase. And secondly, made the generation of custom topologies for the network-on-chip in the context of specific applications (a custom topology can reduce the costs in communication and on the surface).

In our work we studied the hybridization of the mapping problem with the problem of customization of the topology by considering the 3D technology.

Keywords: system-on-chip (SoC), network-on-a-chip (NoC), specific application, optimization, mapping, custom topology and 3D integration technology (TSV).

ملخص

سمح التطورفي تقنيات الدوائر المتكاملة بتجميع العديد من المكونات الإلكترونية على الشريحة نفسها. تتواصل هذه المكونات مع بعضها البعض من خلال اتصالات تقليدية كالتوصيل من نقطة إلى نقطة أو توصيلات الناقل. ومع التطور في الإتصلات و نقل البيانات لم تعد هذه التقنيات صالحة حيث وصلت إلى حدودها في التواصل. فظهر نموذج جديد للتوصيل البيني ألى و هو Network on Chip أو الشبكة على الشريحة، و ذلك للتغلب على مشاكل طرق التوصيل البيني التقليدية. يمكن أن تنشأ العديد من المشكلات أثناء تصميم هذا النموذج الجديد التواصل نظرًا للعدد المعتبر من نقاط التي يجب مراعاتها، نذكر منها تكلفة الاتصال، تكلفة سطح السيليكون للمكونات، اختيار هيكل الشبكة، إلخ.

نحن مهتمون في عملنا هذا من ناحية، في مرحلة رسم الخرائط. هذا الأخير يجعل من الممكن وضع المكونات التي تنفذ أحد التطبيقات على طبولوجيا شبكة على رقاقة، و تقديم أفضل موضع ممكن للمكون، مما يقلل من تكلفة الاتصالات. وثانيا، إنشاء طبولوجيا مخصصة للشبكة على رقاقة في سياق تطبيقات محددة (يمكن لطوبولوجيا مخصصة تقليل تكاليف الاتصالات و مساحة المكونات المستعملة أيضا).

في عملنا هذا أعطين حل ممكن من أجل الوصول إلى أفضل النتائج من خلال درستنا لمشكلة رسم الخرائط مع مشكلة تخصيص الهيكل مستعملين التكنولوجيا ثلاثية الأبعاد في آن واحد.

الكلمات الرئيسية: نظام على رقاقة (SoC)، شبكة على رقاقة (NoC)، تطبيق معين، التحسين، ورسم الخرائط، الطوبولوجيا المخصصة، تكنولوجيا التكامل ثلاثية الأبعاد(TSV).

Remerciements

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

Nous tenons à remercier également ***l'ensemble des professeurs*** *du département d'informatique de l'université Saad Dahleb (USDB), qui ont assurés notre formation tout au long des (05) années d'études, et nous ont transmis leur savoir sans réservé.*

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

Nous tenons à remercier également nos familles, amis, et toutes personnes qui de près ou de loin ont contribué à la réalisation de ce travail.

Dédicaces

DJOUDI Youcef

Je dédie ce travail à Mes parents :
Ma mère, qui a œuvré pour ma réussite, de par son amour, son soutien, tous les sacrifices consentis et ses précieux conseils, pour toute son assistance et sa présence dans ma vie, reçois à travers ce travail aussi modeste soit-il, l'expression de mes sentiments et de mon éternelle gratitude.

Mon père, qui peut être fier et trouver ici le résultat de longues années de sacrifices et de privations pour m'aider à avancer dans la vie. Puisse Dieu faire en sorte que ce travail porte son fruit. Merci pour les valeurs nobles, l'éducation et le soutient permanent venu de toi.

Ma famille qui n'a cessé d'être pour moi des exemples de persévérance, de courage et de générosité.

À mes chers amis de l'université avec qui j'ai passé des moments inoubliables avec.

A Mes professeurs qui vont voir dans ce travail la fierté d'un savoir bien acquis.

MERCI

Dédicaces

Menad BOURENANE

Je commence d'abords à remercier DIEU par le fait qu'on a pu terminer tout un cycle d'étude par un fruit final tel que ce projet.
Je dis « Elhamdoulillah » pour tous ces moments exceptionnels vécus de joie et de difficulté et pour tous les bons accomplissements de savoir-être et de savoir-faire réalisés jusqu'à maintenant.

Je dédie ce modeste travail à mes deux chers parents et à ma sœur, qui ont été toujours un modèle d'honneur et de principes, qui ont été toujours à mes côtés, en sacrifiant leur temps et leurs efforts pour moi et qui ont su être compréhensifs et voulant toujours le bien et la réussite pour moi, à mon chers père et mère et sœur.

A Toute ma famille maternelle et paternelle qui ont eu un impact positif sur ma personne pour atteindre le meilleur de mois même, de près ou de loin, par leurs soutiens et leurs efforts même si avec un souhait ou un sourire.

Je tiens également à remercier mon cher réseau social de la vie réelle ou virtuelle d'avoir été toujours à mon écoute, de me souhaiter toujours le mieux, et de me permettre de faire face à chaque occasion à des challenges de dépassement de soi même en atteignant ma meilleure capacité.

À ma promotrice, mes enseignants, mes camarades et à tous mes amis de l'université SAAD DAHLAB Blida.

MERCI

Table des matières

Résumé....1
Abstract....2
ملخص....3
Remerciements....4
Dédicaces....5
Introduction générale....10
Chapitre 1 : Les réseaux sur puce (NoC)....12
1. Introduction....12
2. Types d'interconnexions....13
3. Les composants du réseau sur puce....17
4. Les caractéristiques du réseau sur puce....19
5. Le mapping et la technologie 3D....23
6. Conclusion....24
Chapitre 2 : Méthodes d'optimisation combinatoire....25
1. Introduction....25
2. L'optimisation combinatoire....25
3. Les heuristiques et les métaheuristiques....26
4. L'optimisation multi objective....26
5. Les métaheuristiques pour l'optimisation multi-objectives....28
6. Exemples de métaheuristiques....33
7. Conclusion....38
Chapitre 3 Mapping et réseau sur puce 3D....39
1. Introduction....39
2. La phase de Mapping....39
3. Le mapping avec d'autres phases....44
4. Technologie 3D....47
5. Topologie 3D (Hybride, Mesh)....47
6. Liens verticaux - TSV....49
7. Synthèse....51
8. Conclusion....52
Chapitre 4 : Implémentation et Tests....53
1. Introduction....53
2. Formulation du problème....53
3. Résolution du problème....56
4. Tests et résultats....65
5. Présentation de l'application....80

6. Conclusion 84
Conclusion générale 85
Références 87

Liste des figures

Figure 1: L'infrastructure de communication point à point [3] 13
Figure 2: L'infrastructure de communication bus partagé [3] 14
Figure 3: L'infrastructure de communication bus hiérarchique [13] 15
Figure 4: Exemple illustratif d'Interconnexion crossbar [15] 15
Figure 5: Paradigme d'interconnexion réseau sur puce [16] 16
Figure 6: Topologie Mesh et ses principaux composants [3] 17
Figure 7: Architecture d'un routeur utilise dans un NoC [17] 17
Figure 8: Interface Ressource-Réseau (IRR) [3] 18
Figure 9: Exemple de topologies réseau sur puce 19
Figure 10: Réseau sur puce 3D [29] 20
Figure 11: Classification partiel des algorithmes de routage [30] 21
Figure 12: Classification partielle des méthodes d'optimisation [46] 28
Figure 13: Principe de base de l'algorithme évolutionnaire (AE) [46] 29
Figure 14: Représentation chromosomique [59] 30
Figure 15: Fonctionnement de l'algorithme génétique [59] 31
Figure 16: Exemple d'opération: (a) Croisement un point, (b) Croisement deux point, (c) Mutation [59] 33
Figure 17: Distance de crowding [59] 34
Figure 18: Fonctionnement du NSGA-II [59] 35
Figure 19: Fonctionnement du SPEA-II [59] 37
Figure 20: Exemple du principe de mapping 39
Figure 21: Les critères de classification des stratégies de mapping [29] 40
Figure 22: Classification selon la méthode choisie - heuristique ou métaheuristique [40] 41
Figure 23: Mapping en utilisant la technique SPIRAL [29] 42
Figure 24: Le mapping des PIs suivant la technique de zigzag [80] 42
Figure 25: (a) Tuiles candidates pour le placement du premier PI, (b) Concept de chemins sous forme losange, (c) Définition de 4 mouvements [82] 43
Figure 26: (a) Graphe de communication VOPD, (b) Chaine initiale, (c) Chaine finale [79] 43
Figure 27: (a) Floorplanning initial, (b) topologies final [93] 45
Figure 28: Topologie 2D-Mesh irrégulière 46
Figure 29: Les architectures Meshs: (a) Mesh 2D, (b) Mesh 3D, (c) 3D Mesh empilé. (d) 3D Mesh cilié [101] 48
Figure 30: Commutateurs 3D Mesh: (a) Standard, (b) Empilé, (c) Cilié [101] 48
Figure 31: Empilement de plusieurs puces par Wire Bonding [104] 49
Figure 32: Illustration d'un NoC 3D avec ses composants [107] 49
Figure 33: De la Mesh 2D vers Mesh 3D 50
Figure 34: Représentation explicatif de notre supposition 54
Figure 35: Exemple de mapping dans 1 seul plan 55
Figure 36: Présentation d'une solution mapping 59
Figure 37: Représentation graphique d'affectation de TSV 59
Figure 38: Croissement à un point 61
Figure 39: Croissement à m points 61
Figure 40: Exemple de mutation 62

Figure 41: Exemple d'enfants non valides générés après un croissement 62
Figure 42: Réparation d'un chromosome non valide 63
Figure 43: Benchmark VOPD [111] 65
Figure 44: Benchmark MPEG [40] 66
Figure 45: Front Pareto du benchmark VOPD avec les 2 algorithmes 76
Figure 46: Illustration des 25 exécutions en utilisant Benchmark MPEG 77
Figure 47: interface accueil pour utilisateur 80
Figure 48: Interface principale de travail 80
Figure 49:solution mono-objectif (optimisation de coût).....82
Figure 50: Affichage après le clique sur summarize.....82
Figure 51 :Affichage après le clique sur bouton Display.....83
Figure 52 : Génération de topologie avec les différents LV affectés.....83
Figure 53: l'utilitaire de comparaison en %TSV des NoC partiellement interconnectés.....84

Liste des tableaux

Tableau 1: l'utilisation de L'architecture bus partage par des Entreprise[4] 14
Tableau 2: Récapitulatif des structures d'interconnexions [16].....16
Tableau 3: Exemple des méthodes heuristiques et métaheuristiques [29] 41
Tableau 4: Comparaison entre topologies régulière et personnalisée [96].....46
Tableau 5: Comparaison entre les 2 algorithmes utilisés 58
Tableau 6: Tests fait pour fixer Pc et Pm pour Benchmark Random en utilisant les 2 techniques.....67
Tableau 7: Tests fait pour fixer Pc & Pm pour VOPD en utilisant les 2 techniques 69
Tableau 8: Tests effectue sur le Benchmark MPEG pour fixer Pc et Pm en utilisant les 2 techniques 71
Tableau 9: Résultats de variation de nombre de population sur les différents benchmarks pour les 2 méthodes utilisées.....73
Tableau 10: Récapitulatif des résultats des paramètres.....75
Tableau 11: Résultats obtenus après 15 exécutions en utilisant benchmark VOPD.....76
Tableau 12: Tests effectués en comparant nos résultats à la littérature 78

Introduction générale

De jour en jour, de nouvelles applications plus complexes et plus personnalisées sont introduites. Les systèmes complets dits systèmes sur puce (SoC) permettent de répondre aux besoins de ces applications.

Avec le développement technologique, il est devenu possible d'intégrer plusieurs composants dans un même système sur puce. La communication entre ces composants est assurée par des paradigmes d'interconnexions tels que le point à point et le bus.

Le bus est le média le plus répondu et utilisé actuellement. Il est facilement extensible mais c'est au détriment de l'énergie qui est proportionnelle à la longueur des fils de communications. De plus, une seule communication est autorisée à la fois vu que le canal de communication est partagé par tous les composants. Du fait que les systèmes actuels se caractérisent par une forte communication, les architectures à base de bus sont devenues inefficaces.

C'est dans ce contexte là que le nouveau paradigme réseau sur puce (NoC) est apparu. Cette nouvelle interconnexion inspirée des réseaux informatiques classiques permet une meilleure communication dans le système, en offrant une bonne flexibilité, moins de consommation d'énergie et de surface.

La problématique de la mise en œuvre des réseaux sur puce est à la largeur de son espace de conception. Du fait, du nombre de paramètres à prendre en compte. Comme par exemple : le choix de la topologie, le positionnement des composants (mapping), ou même l'utilisation des nouvelles technologies (3D).

Il est donc nécessaire de disposer d'un outils d'aide à la conception afin d'assister et de guider le concepteur dans ses choix afin de concevoir un circuit à faible coût tout en optimisant le coût en communication et la surface du circuit.

C'est pour cela que nous nous sommes intéressés dans notre projet à l'étude de l'hybridation de la phase de mapping avec la personnalisation de la topologie en considérant la technologie 3D. Ce problème se présente comme un problème d'optimisation combinatoire multi-objective. De ce fait, nous nous somme orientés vers l'utilisation des métaheuristiques de l'approche Pareto.

Plan du document

Ce mémoire est constitué de quatre parties :

- Dans le premier chapitre, nous allons introduire le principe des réseaux sur puce (Network on Chip – NoC). Nous verrons quelques types d'interconnexion standards. Puis, nous présenterons l'ensemble des composants et caractéristiques du nouveau paradigme NoC.

- Dans le second chapitre, nous présenterons l'aspect d'optimisation combinatoire. Nous verrons quelques méthodes d'optimisations multi objectives (heuristiques et métaheuristiques). Et nous donnerons par la suite quelques exemples de métaheuristiques, dont le NSGA-II et SPEA-II.

- Dans la troisième partie, nous parlons du problème de « mapping » dans les réseaux sur puce. Dans un premier temps, nous allons introduire la phase de mapping. Puis, nous présentons les autres phases de conceptions qui peuvent être combinées avec la phase de mapping. Ensuite, nous allons voir l'impact d'utiliser la technologie 3D sur les performances des systèmes sur puce. En terminant avec une synthèse sur laquelle se basera notre contribution présenter dans le chapitre 4.

- Le chapitre 4 contient la phase de conception de notre solution. Nous avons commencé d'abords par la formulation mathématique. Puis, nous avons présenté les deux algorithmes proposés, notre solution SGANova qui est inspirée des algorithmes génétiques et le SPEA-II adaptée pour pouvoir se comparer à la littérature. Des tests sur plusieurs benchmarks sont effectués afin d'analyser les résultats obtenus et mettre en évidence les avantages de la solution élaborée.

Chapitre 1 : Les réseaux sur puce (NoC)

1. Introduction

Selon la loi de Moore, la capacité d'intégration dans les circuits intégrés (CI) augmente chaque 18 mois (des millions de transistors dans une même puce). Plusieurs éléments de mémoires, d'unités de traitement et d'autres composants sont rassemblés sur une même puce, donnent naissance à des systèmes complets appelés communément systèmes sur puce (System on Chip – SoC).

La façon dont les composants d'un système sur puce communiquent influence considérablement sur les performances du système. L'utilisation des paradigmes traditionnels comme le Point à Point ou Bus Partagé trouvent leurs limites en terme d'extension et de bande passante lorsqu'un fort degré de communication est appliqué.

Afin de surmonter ces défis, un nouveau paradigme dit réseau sur puce (Network on Chip – Noc) est apparu. Inspirer des réseaux informatiques standards, le réseau sur puce à l'avantage d'une bonne flexibilité de communication, d'une consommation d'énergie et de surface réduite.

De nombreux problèmes sont rencontrer lors de la conception d'un réseau sur puce dont : le placement des composants d'une application (IP) sur une architecture, ou le choix d'une topologie tout en considérant certains critères pendant la conception, tels que le coût de communication et la surface occupée.

Dans ce qui suit, nous allons définir le concept du réseau sur puce. Nous allons décrire les différents types d'interconnexions utilisées dans les systèmes sur puce. Ensuite, nous citons l'ensemble des composants qui constitue un réseau sur puce. Et on termine par quelques définitions des concepts liés à ce paradigme.

2. Types d'interconnexions

Un système sur puce peut être composé de CPUs, de mémoires, de liens ou de circuits spécifiques. La communication entre ces composants est assurée par une variétés de systèmes d'interconnexions. Le point à point, le bus partagé et le réseau sur puce sont considérer parmi les fameuses interconnexions citées dans la littérature.

2.1. Le point à point

L'infrastructure de communication point à point (**figure 1**) utilise des interconnexions de communications directe entre chaque deux unités (Propriété Intellectuelle - IP) du système **[1]**. Cette dernière permet aux IPs de prendre en charge plusieurs communications simultanément (débits de transfert de données très élevés).

Cependant, cette structure peut être utilisée dans des systèmes qui comportent peu de composants. Car chaque paire de IPs nécessitent un lien de communication direct, ce qui assure un bon parallélisme mais qui engendre un coût de fabrication très élevé, surtout que le taux d'utilisation de ces liens est très bas (environ 10% selon **[2]**).

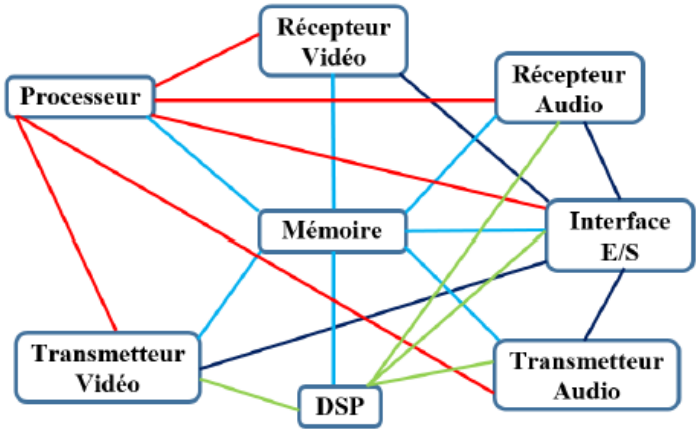

Figure 1: L'infrastructure de communication point à point [3]

2.2 Le bus partagé

De nombreuses fabrications industriels **(tableau 1)** font appels à l'utilisation de la structure d'interconnexion bus partagé **(figure 2)**. Ce type d'interconnexion est composée d'une unité d'arbitrage, de ligne de donnée et une autre ligne de contrôle (ACK), Elle représente l'amélioration de la structure point à point en terme de coût de fabrication. Elle consiste à relier plusieurs composants entre eux par un seul canal de communication (une bande passante partagée).

Tableau 1: l'utilisation de L'architecture bus partage par des Entreprise[4]

Architecture de bus	Entreprise / Propriétaire	Références
Advanced Microcontroller Bus Architecture (AMBA)	ARM	**[5]**
CoreConnect	IBM	**[6]**
CoreFrame	PalmChip	**[7]**
STBus	STMicroelectronics	**[8]**
SiliconBackplane	Sonics	**[9]**
Peripheral Interconnect Bus (IP-Bus)	Open Microprocessor systems Initiative (OMI)	**[10]**
Wishbone Interconnect	Silicore Corporation	**[11]**

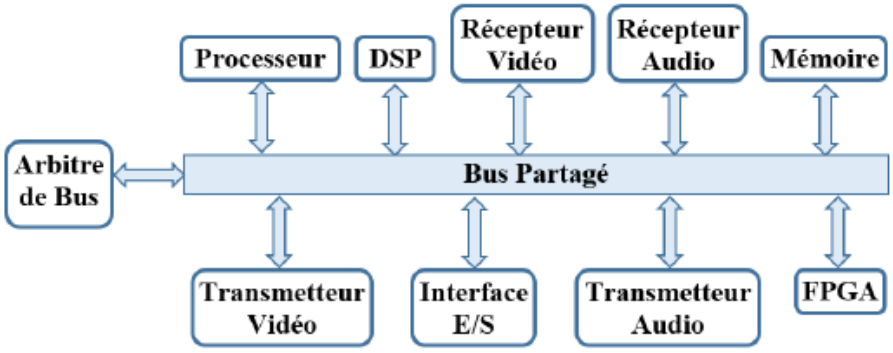

Figure 2: L'infrastructure de communication bus partagé [3]

L'inconvénient majeur de ce type d'interconnexion est qu'elle ne permet qu'une seule communication à la fois. En plus, cette structure de communication présente un goulot d'étranglement des données dès que le nombre de composants interconnectés augmente significativement **[12]**.

2.3 Le bus hiérarchique

L'interconnexion bus hiérarchique est présentée par la connexion des plusieurs bus standards sous format hiérarchique **(figure 3).** L'idée consiste à relier plusieurs bus partagés entre eux à l'aide d'un intermédiaire appelé le pont (Bridge). Chaque bus fonctionne indépendamment de l'autre, sauf dans le cas où l'échange se fait entre des IPs de bus différents. Cependant, la mise en œuvre à faible coût de cette technique nécessite des systèmes à nombre d'unités réduit.

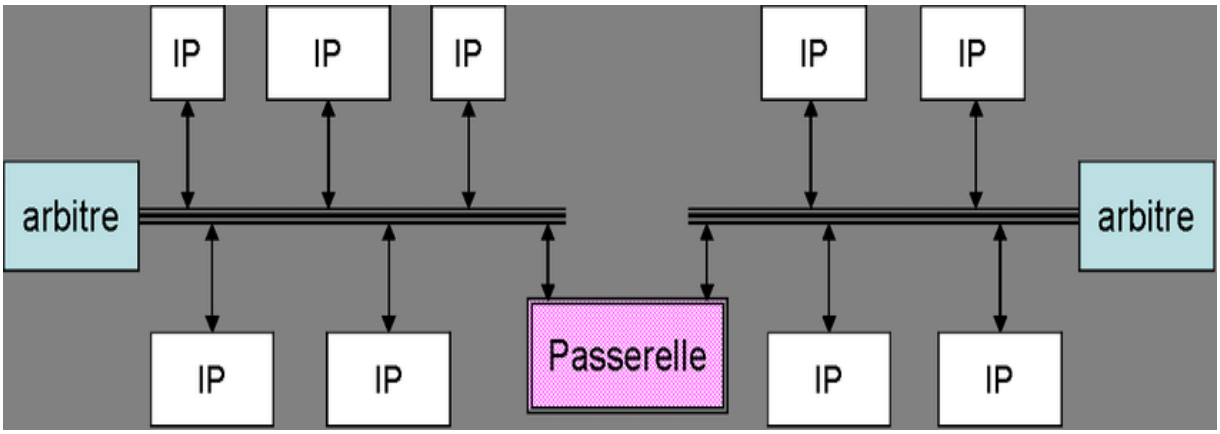

Figure 3: L'infrastructure de communication bus hiérarchique [13]

2.4. Le Crossbar

Le principe consiste à définir une matrice de multiplexeurs permettant à toute unité du système de communiquer avec une autre, en autorisant ainsi des communications en parallèle. Cette approche est très coûteuse en surface mais convient pour des systèmes avec un nombre d'unités de calcul réduit **[14]**, une illustration de ce type d'interconnexion est représentée dans la **figure 4.**

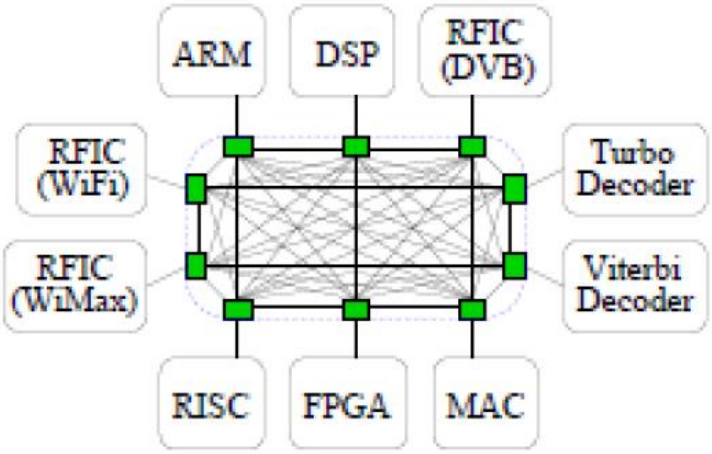

Figure 4: Exemple illustratif d'Interconnexion crossbar [15]

2.5. Le réseau sur puce

L'idée du réseau sur puce est apparue dans les années 90, mais elle n'a commencé à être étudiée qu'à partir de l'an 2000 **[14].** L'évolution actuelle de la technologie d'interconnexion réseau sur puce dans les SoC présente l'atout qui surmonte les problèmes rencontrés dans les anciennes paradigmes d'interconnexions.

La **figure 5** illustre une structure de réseau sur puce (NoC) de type maillé 2D, composée de plusieurs éléments (IPs), connectés les uns aux autres via des routeurs, des fils d'interconnexions et un module d'interface réseau (IRR) dont la fonction principale est la mise en paquets de toutes les données qui circulent sur le réseau (la **section 3** détaille les rôles des composants d'un NoC).

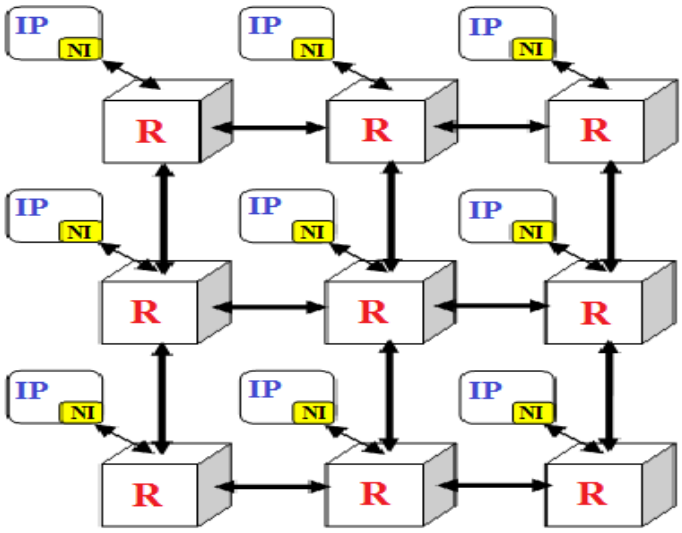

Figure 5: Paradigme d'interconnexion réseau sur puce [16]

Le **tableau 2 [16]** résume les différentes caractéristiques des trois interconnexions utilisées pour la communication dans les systèmes sur puce.

Tableau 2: Récapitulatif des structures d'interconnexions [16]

	Parallélisme	**Consommation**	**Mise en échelle**	**Réutilisation**
Point à Point	Très bon	Bon	Très mauvais	Très mauvais
Bus Partagé	Très Mauvais	Très Mauvais	Très Mauvais	Très bon
Bus Hiérarchique	Bon	Mauvais	Mauvais	Très Bon
Réseau sur Puce	Très bon	Très bon	Très bon	Très bon

La nouvelle structure réseau sur puce offre de nombreux avantages de performances, à savoir : la mise en échelle (ajout de nouveaux nœuds) sans altérer le fonctionnement général du système, moins de latence en communications et en transfert de données à cause des fils d'interconnexions qui sont plus courtes, et un bon parallélisme et réutilisation de ressources pendant les communications établies.

Dans ce qui suit, nous détaillons les concepts (composants et caractéristiques) de base des réseaux sur puce.

3. Les composants du réseau sur puce

Le réseau sur puce est composé de trois éléments de base (**figure 1.5**). Le routeur qui assure la bonne orientation des paquets (données) partagés, depuis la source vers la destination. L'interface ressource-réseau (IRR) qui adapte le traitement des communications au niveau des IPs pour accéder au réseau sur puce. Et les liens de communications point à point qui se chargent du transfert des données entre les routeurs.

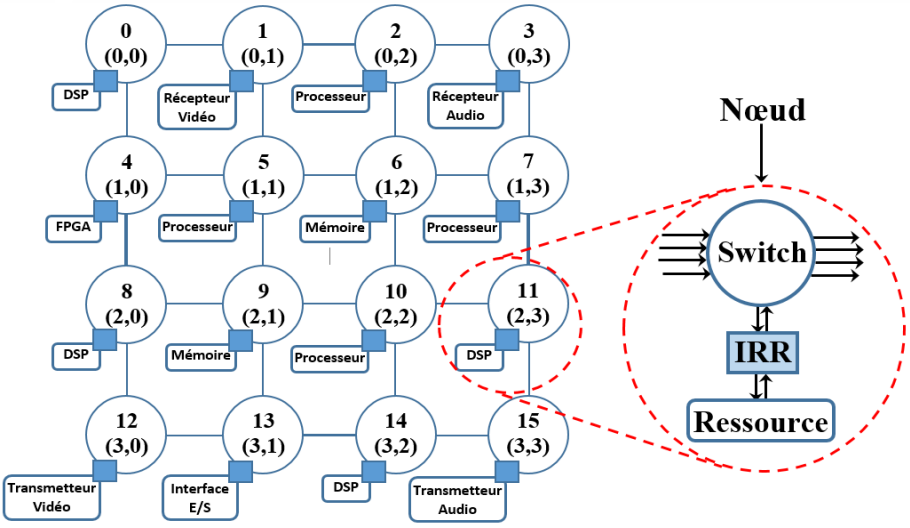

Figure 6: Topologie Mesh et ses principaux composants [3]

3.1. Routeur

Le premier élément de base qui construit un réseau sur puce est le routeur (**figure 6)**. Les routeurs ou les commutateurs sont des composants qui utilisent des algorithmes de routage pour déterminer l'acheminement des données de la source à la destination. Ils sont souvent constitués de :

1- Un commutateur qui connecte les files d'attente au différents ports E/S,
2- Une unité de routage et d'arbitrage afin de gérer les situations de conflits,
3- Des files d'attente pour stocker les paquets qui passent sur le réseau.

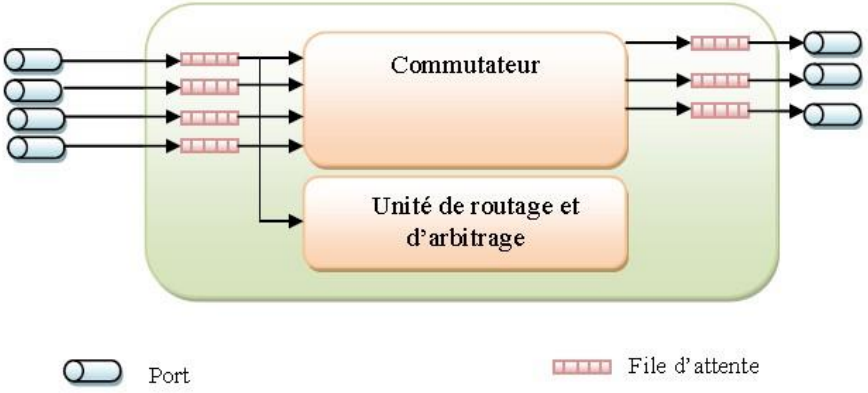

Figure 7: Architecture d'un routeur utilise dans un NoC [17]

3.2 Interface Ressource-Réseau (IRR)

L'IRR est un autre composant essentiel qui relie une ressource à un routeur. Elle permet de traduire les messages envoyés par les IPs en données compréhensible par la structure d'interconnexion réseau sur puce.

L'IRR se compose de deux parties, la partie dépendante des ressources et la partie indépendante des ressources **(figure 8)**. La partie dépendante des ressources est conçue pour adapter les protocoles de communication entre la ressource et le routeur. Par contre, la partie indépendante fait en sorte que les données adapter soient transférer pour être partager dans le réseau.

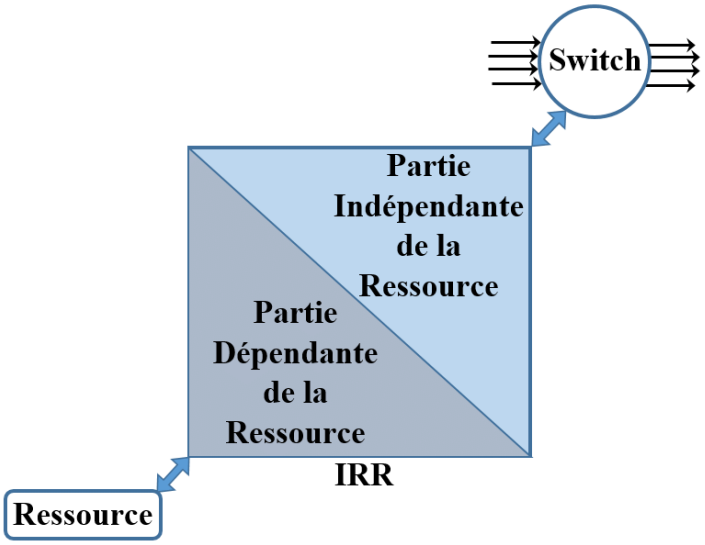

Figure 8: Interface Ressource-Réseau (IRR) [3]

3.3. Les liens

Les liens sont des canaux de transport de données, entre deux composants du réseau sur puce (entre deux routeurs ou bien entre un routeur et une IRR) **[14]**. Ils offrent la bande passante nécessaire pour établir une communication.

4. Les caractéristiques du réseau sur puce

4.1. Topologies

Les composants d'un réseau sur puce sont reliés selon une certaine topologie. De nombreuses topologies **(figure 9)** ont été utilisées pour la construction des interconnexions de communication (principalement le réseau sur puce).

Dans les réseaux sur puce, chaque routeur est connecté à un IP et aux autres routeurs voisins. Ils utilisent souvent les réseaux maillés 2D mesh, torus, folded torus et octagon **[2, 18 - 20]**.

La topologie matricielle mesh semble être la topologie la plus répandue dans le domaine des réseaux sur puce. Ceci est dû à la même longueur de tous les canaux dans le réseau et à l'extensibilité de sa structure qui croît linéairement avec le nombre de noeuds connectés **[12]**.

Une topologie peut être régulière standard comme Mesh et Torus, irrégulière comme Fat -Tree et Butter Fly-Fat-Tree ou personnalisée en mélangeant les deux types précédents **[2]**.

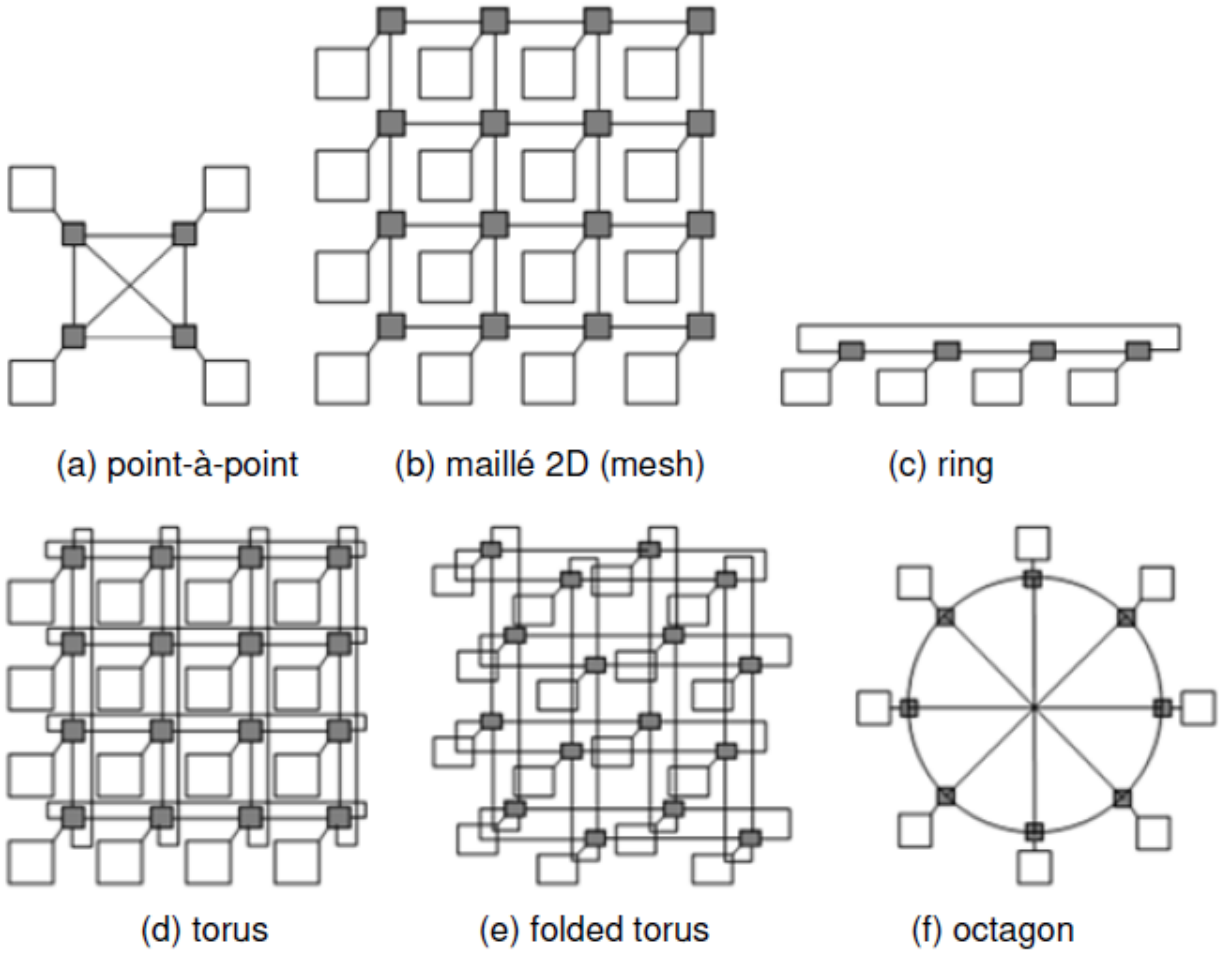

Figure 9: Exemple de topologies réseau sur puce

Plusieurs travaux **[21 - 28]** basés sur des topologies en maille tridimensionnelle **(figure 10)** ont été développées ces dernières années. Des NoCs 3D nécessitent la construction de nouveaux routeurs (routeurs 3D), ainsi que le développement d'algorithmes de routage qui prennent en considération la 3ème dimension.

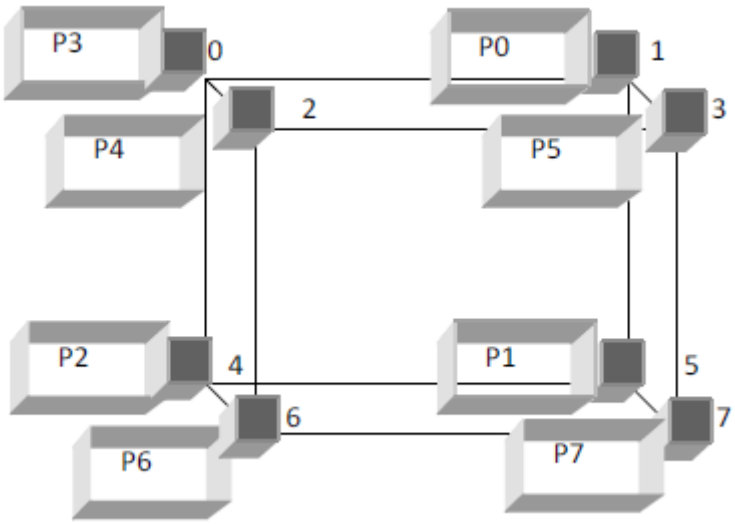

Figure 10: Réseau sur puce 3D [29]

Un mécanisme de routage est invoqué pour éviter toute situation de surcharge ou d'interblocage dans le réseau. Le paragraphe suivant donne des définitions de ce mécanisme, avec d'autres concepts liés à la communication.

4.2. Techniques de routage

Les algorithmes de routage déterminent le chemin dont les données sont acheminées de la source à la destination. Plusieurs techniques de routage (**figure 11** de **[30]**) sont présentées dans la littérature, citons les algorithmes de routage : déterministe, adaptatif **[31]**, tolérant au pannes ou non **[29]**.

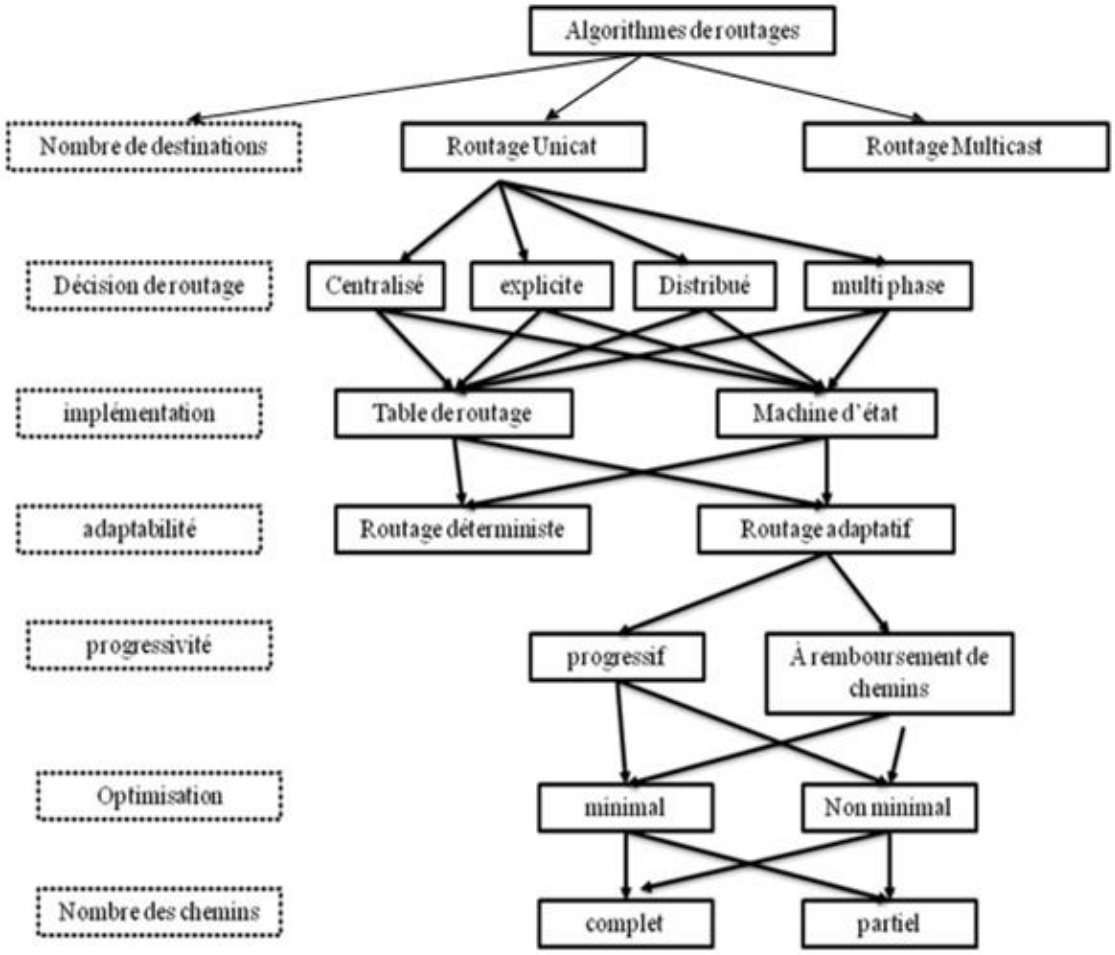

Figure 11: Classification partiel des algorithmes de routage [30]

Dans un algorithme de routage déterministe, les chemins sont définis et utilisés indépendamment de l'état actuel du réseau (Algorithme XY **[31]**). Ce type d'algorithme de routage ne prend pas en compte la charge actuelle des routeurs et des liens du réseau lors des décisions de routage **[14]**.

Au contraire, dans un algorithme de routage adaptatif, les décisions de routage sont prises en fonction de l'état actuel du réseau (la charge du réseau, la disponibilité des liens). Par conséquent, le trafic entre une source et une destination peut changer ses chemins de routage au cours du temps en fonction des conditions du réseau **[14]**.

Comme une dernière classification, les algorithmes de routage peuvent être considérés comme tolérants aux fautes ou non. On dit qu'un algorithme est tolérant aux fautes lorsque malgré la défaillance d'un des routeurs du réseau, l'acheminement des paquets peut être effectué. Plusieurs exemples d'algorithmes de routage tolérants aux fautes peuvent être trouvés dans la littérature **[32 - 34].**

L'algorithme XY est un algorithme de routage déterministe. Les éléments dans le réseau sur puce sont bien identifiés par des coordonnées géométriques (**figure 6 pg. 17**) au lieu d'adresses IP (Internet Protocol). Il consiste à router les paquets, horizontalement (sur l'axe X) puis verticalement (sur l'axe Y), de la source à la destination, tout en évitant les situations d'interblocage. Ces éléments suivent un certain mode de commutation que nous allons présenter dans le paragraphe suivant.

4.3. Mode de commutation

Les modes de commutation définissent la façon dont la communication est faite dans le réseau. La stratégie d'allocation des différentes ressources du réseau sur puce afin d'acheminer les données suit deux modes de commutation **[35, 14]** : la commutation de circuits et la commutation de paquets.

4.3.1. La commutation de circuits

Ce mode de commutation consiste à établir un circuit dédié au sein du réseau pour chaque paire émetteur/récepteur **[35]**. C'est-à-dire que les chemins de communication doivent être connu préalablement avant que les transmissions puissent avoir lieu.

Ceci garanti la connaissance de la bande passante et la latence de transmission quand une connexion est établie. Cependant, un tel mode de commutation réquisition les ressources tout au long du transfert.

4.3.2. La commutation de paquets

Le principe de commutation par paquets consiste à envelopper le message qui sera communiqué entre deux routeurs en paquets. Ce mode de commutation permet un meilleur partage d'éléments du réseau, vu que les voies sont libérées dès qu'une partie du message est envoyé. En revanche, le routage de ces paquets augmente la latence dans le système car l'acheminement dans ce type de commutation est plus complexe.

Ce mécanisme découpe le paquet en petites parties appelées flits. En plus de cela, l'enveloppe doit contenir les informations nécessaires pour son routage avant son envoi. L'ensemble des unités flits sont constituées de trois façons possibles **[36, 16]** : Store and Forward, Virtual Cut-Throught ou le Wormhole.

4.4. Contrôle de flux

Le contrôle de flux est un ensemble de mécanismes qui permet d'éviter la surcharge du réseau et réguler le trafic. L'équilibre de charge (échange de données) dans le réseau est assuré en utilisant deux signaux : signal de requête et signal d'acquittement. Le contrôle de flux est réalisé par deux méthodes : le HansdShake et le Crédit-Base.

4.4.1. HandShake

Cette technique se base sur le principe d'attente d'un acquittement (ACK). L'émetteur n'envoie plus de paquets et reste en attente tant que l'extrémité récepteur ne lui a pas encore envoyé un ACK de réception de l'ancienne donnée envoyée. Par conséquent, la latence du système augmente et les performances dégradent **[14]**.

4.4.2. Crédit-Base

Cette stratégie consiste à utiliser des crédits qui vont indiquer à l'émetteur la quantité de places disponibles dans la mémoire du destinataire. Au départ, l'émetteur dispose de la totalité des crédits, puis, au fur et à mesure qu'il envoie les données, il décrémente son nombre de crédits jusqu'à ce que ce dernier devienne nul. Lorsque le récepteur consomme les données qui lui ont été envoyées, il informe l'émetteur du nombre de places nouvellement libérées **[14]**.

4.5. Paramètres d'un réseau sur puce

De nombreux critères sont à prendre en compte pour évaluer un système réseau sur puce. Parmi les critères les plus discutés dans la littérature, il y a : la surface, l'énergie et la diversité dans les chemins de communication :

- **La surface** qui représente l'espace ou l'aire occupée par l'ensemble des composants qui constituent le réseau sur puce (Routeurs, Liens, …etc).
- **L'énergie** nécessaire consommée par le réseau sur puce pendant son fonctionnement. Elle est représentée par la somme des énergies de chaque composant. Elle est aussi relative au nombre de ressources utilisé **[37]**.
- **La diversité de chemins** qui caractérise le réseau sur puce. Elle donne la possibilité de trouver des chemins optimaux et des chemins de secours en cas de panne lors d'une communication dans le réseau **[38]**.

En plus de ces caractéristiques, il y a la bande passante nécessaire pour l'envoi des données, la latence qui correspond au délai nécessaire pour acheminer un paquet **[37].** L'extensibilité du réseau par le rajout de nouveaux éléments tout en assurant la non provocation d'un dysfonctionnement du système **[38]**. Ainsi que la flexibilité lorsque le système opère dans divers autres systèmes **[39].**

5. Le mapping et la technologie 3D

De nos jours, les concepteurs des circuits électroniques optent à utiliser l'architecture réseau 2D-Mesh vu sa facilité à être implémentée sur la technologie silicium, sa connectivité rapide et son efficacité pour les communications. Parmi les phases de conception pour la construction d'un réseau sur puce, on a la phase de mapping.

5.1. Le Mapping

La phase de mapping est l'une des phases les plus importantes pour construire un réseau sur puce. Elle associe chaque IP d'une application à une ressource du réseau. Théoriquement, s'il y a N IPs à placer sur M nœuds du réseau, alors on peut avoir M!/(M-N)! possibilités de placement **[40]**.

Plusieurs techniques et méthodes ont été proposées pour le placement des IPs dans un réseau sur puce. Selon la référence **[40]**, les IPs qui communiquent souvent doivent être placés le plus proche possible afin de minimiser les coûts de communication. Le problème de mapping est un problème d'optimisation combinatoire NP-difficile. L'espace de recherche croit avec la taille du système. 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 **[40]**.

Le placement des composants peut être décidé pendant la conception du circuit, c'est-à-dire avant l'exécution de l'application, dans ce cas, on parle du mapping statique (possible pour les applications spécifiques : ASIC). Ou bien, au cours d'exécution, où le placement des tâches peut être changé, et ceci est appelé le mapping dynamique (applicable pour les applications MPSoC).

La technologie 3D a permet de construire des topologies 3D. Par conséquence, de nouvelles techniques de placement (mapping) ont été étudiées.

5.2. La technologie 3D

L'évolution de la technologie 3D dans les circuits intégrés donne naissance au réseau sur puce 3D. Selon **[41]**, les réseaux sur puce traditionnels (2D) peuvent être élargir à la troisième dimension en ajoutant des routeurs 3D, en assurant des communications verticales par l'interconnexion Throught Silocon Via (TSV) **[42]**, en offrant une meilleure communication entre les IPs de différents niveaux de l'architecture, et ceci par la diminution des chemins parcourus pour transférer les données.

6. Conclusion

Dans ce chapitre, nous avons introduit le nouveau paradigme réseau sur puce. Nous avons vu l'ensemble de composants qui le constitue, quelques concepts liés aux réseaux sur puce, ainsi que les paramètres à prendre en compte pendant la conception de cette structure d'interconnexion.
Trouver une combinaison optimale pour le choix des paramètres du réseau sur puce (topologie, placement des composants, etc.). Afin de maximiser ses performances est un problème d'optimisation combinatoire multi objectives.
Dans le chapitre suivant, on va expliquer plus en détail le concept d'optimisation, décrire quelques méthodes liées à ce concept, ainsi que des exemples illustratifs.

Chapitre 2 : Méthodes d'optimisation combinatoire

1. Introduction

Le nouveau paradigme réseau sur puce a pu surmonter les problèmes des méthodes d'interconnexion classiques en répondant aux nouveaux besoins des utilisateurs. Néanmoins, des problèmes peuvent surgir lors de la conception de ce type d'architecture. Du fait que, plusieurs paramètres sont à prendre en compte et qui influencent sur les performances globales du système.

Le choix des paramètres d'un réseau sur puce est un problème très complexe. Citons le coût de communication, la surface occupée, le choix de la topologie du réseau, etc. Trouver une combinaison optimale de ces paramètres afin d'améliorer les performances d'un réseau sur puce est classé comme un problème d'optimisation combinatoire NP-difficile.

Ce chapitre sera consacré à l'étudie des méthodes d'optimisations combinatoires afin d'introduire la méthode proposée dans le cadre de notre projet.

2. L'optimisation combinatoire

Un problème d'optimisation combinatoire est défini par un ensemble de solutions candidates et par une fonction objective qu'associe à chaque solution son coût. La résolution d'un problème d'optimisation combinatoire consiste à déterminer une solution qui optimise une fonction objective donné.

En mathématique, l'optimisation recouvre toutes les méthodes qui permettent de déterminer l'optimum d'une fonction. Pour les problèmes d'optimisations combinatoires de taille raisonnable, les méthodes exactes peuvent trouver des solutions optimales. Ces méthodes explorent de façon exhaustive tout l'espace de recherche jusqu'à ce qu'elles trouvent une solution optimale. Généralement, le nombre de solutions est fini, mais peut être extrêmement grand, ce qui exclut l'énumération exhaustive (méthode exacte) comme une méthode de recherche pour les problèmes complexes.

Dans ce contexte, la classe des problèmes NP-difficiles rassemble les problèmes pour lesquels il n'existe pas d'algorithme qui fournit toujours une solution optimale dans un temps polynomial. Face à cette situation et afin de déterminer la meilleure solution dans un espace de recherche vaste et dans un temps raisonnable, les méthodes approchées ont vu le jour. Ces méthodes explorent partiellement l'espace de recherche et ne garantissent pas l'optimalité, mais, qui offrent des solutions presque optimales en un temps raisonnable.

Deux classes de méthodes approchées sont généralement utilisées en pratique pour de nombreux problèmes difficiles de grande taille : les heuristiques et les métaheuristiques **[43]**.

3. Les heuristiques et les métaheuristiques

En général, une méthode approchée (heuristique ou métaheuristique) examine seulement une partie de l'espace de recherche où le choix des solutions est fait par le biais d'une stratégie. Les heuristiques sont des solutions développées pour résoudre des problèmes spécifiques. Ils ont de grande performances et sont efficace pour trouver l'optimum d'une manière exacte dans des espaces de recherche réduits, mais qui demandent beaucoup de savoir-faire et d'expertise dans le domaine **[43].** Quant aux métaheuristiques, elles sont des méthodes génériques peuvent être appliquées sur un ensemble de problèmes. Ces méthodes sont des algorithmes inespérés généralement de la nature (physique, évolution biologique, éthologique…) et qui sont adaptatives à n'importe quel problème d'optimisation **[43].** Des exemples de métaheuristiques sont présentés dans le **tableau 3** du **chapitre 3.**

Le nombre d'objectifs a optimisés permet de distinguer entre un problème mono objectif et multi objectif. La résolution d'un problème d'optimisation mono-objectif consiste à chercher une solution qui optimise la fonction objective. Cependant, les problèmes de type multi-objectives cherchent à optimiser simultanément plusieurs objectives à la fois, et qui peuvent être contradictoires.

Les métaheuristiques à base de population de solutions sont plus adaptées aux problèmes multi objectif car elles proposent plusieurs solutions optimales (qui optimisent tous les objectifs).

4. L'optimisation multi objective

La résolution d'un problème d'optimisation combinatoire multi objectives consiste à trouver un ensemble de solutions qui optimisent simultanément plusieurs objectives. Dans la littérature, deux classifications des méthodes de résolution des problèmes d'optimisations multi objectives sont proposées : une classification de point de vue décideur, et une classification de point de vue concepteur.

4.1. Classification du point de vue décideur

– **Les méthodes à priori** : sont des méthodes où le décideur intervient dès le début du processus de la conception. Il présente l'importance de chaque objectif voulu atteindre afin de pouvoir transformer le problème multi-objectif en mono-objectif.

– **Les méthodes interactives** : sont des méthodes qui nécessitent la présence du décideur avec le concepteur tout au long du processus de conception. Il peut intervenir itérativement en modifiant certaines valeurs ou contraintes pour diriger le processus de conception vers l'optimum.

– **Les méthodes à postériori** : sont des méthodes qui fournissent pas une solution mais plusieurs bonnes solutions bien répartis. À la fin du processus, le décideur peut choisir la solution qui correspond à ses attentes parmi l'ensemble des solutions trouvées.

4.2. Classification du point de vue concepteur

Selon **[44]** les problème d'optimisation peut avoir deux façon de les résoudre. Avec les approche non Pareto ou au sens du Pareto.

- **Les approches non Pareto :** transforment l'aspect multi-objectif du problème, soit en mono objectif, soit en traitent chaque objectif séparément des autres objectives. Citons :

Approche d'agrégation qui consiste à avoir une seule fonction objective qui rassemble l'ensemble des fonctions en multipliant chacune par le coefficient attribuer par le décideur.

But programmé qui consiste à définir les buts de chaque objectif voulu atteindre par le décideur, dans une seule fonction transformative. Le but est de minimiser l'écart entre ces derniers et les résultats qui seront obtenus.

Approches e-contraintes qui consiste à choisir une seule fonction objective à optimiser et faire la transformation du reste des objectifs en contraintes.

Sélection lexicographique proposée par **[45]**, elle consiste à classer les fonctions objectives par ordre de priorité. Pour trouver la solution optimale, chaque solution d'un objectif plus prioritaire est présentée comme une entrée contrainte pour trouver la solution du moins prioritaire.

- **Les approches Pareto :** elles sont de type à postériori.

La résolution d'un problème d'optimisation multi objectives ne donne pas une solution unique mais plusieurs solutions possibles. L'ensemble de ces solutions est souvent non optimal car elles ne minimisent pas toutes les fonctions objectives (ceci est dû au fait que des objectifs peuvent êtres antagonistes), mais qui sont des solutions de compromis.

Pour identifier ces meilleurs compromis, il faut définir une relation d'ordre entre ces éléments.
Cette relation est appelée relation de dominance. La solution présentée pour un problème multi-objectives n'est pas unique, mais qu'est un ensemble de solutions non dominées appelé l'ensemble de Pareto.

Une solution réalisable X dite non dominée si et seulement si toutes les fonctions objectives de X dominent les fonctions objectives de n'importe quelle solution Y de l'ensemble des solutions réalisables.

Toute solution de l'ensemble Pareto peut être considérée comme optimale puisqu'aucune amélioration ne peut être faite sur un objectif sans dégrader la valeur relative à un autre objectif. Ces solutions forment le Front Pareto.

5. Les métaheuristiques pour l'optimisation multi-objectives

Comme nous avons vu précédemment, la résolutions d'un problème d'optimisation multi-objectives nécessite la connaissance à priori des objectives voulus atteindre. Les méthodes métaheuristiques sont souvent les plus vues dans la littérature pour la résolution des problèmes de complexité élevée. Par contre, leur difficulté réside dans l'adaptation de la méthode au problème d'optimisation et leurs paramètres.
Nous citons dans les paragraphes suivants des exemples de métaheuristiques à base de population de solutions qui permettent l'optimisation multi-objectif.

5.1. Les Algorithmes Evolutionnaires AE

L'application des algorithmes évolutionnaires aux problèmes d'optimisations multi objectives ont permets de mettre en avant l'intérêt d'utiliser des méthodes basées sur le concept de population pour trouver un ensemble de solutions. La **figure 12** présente une classification partielle des méthodes d'optimisation rencontrer dans la littérature.

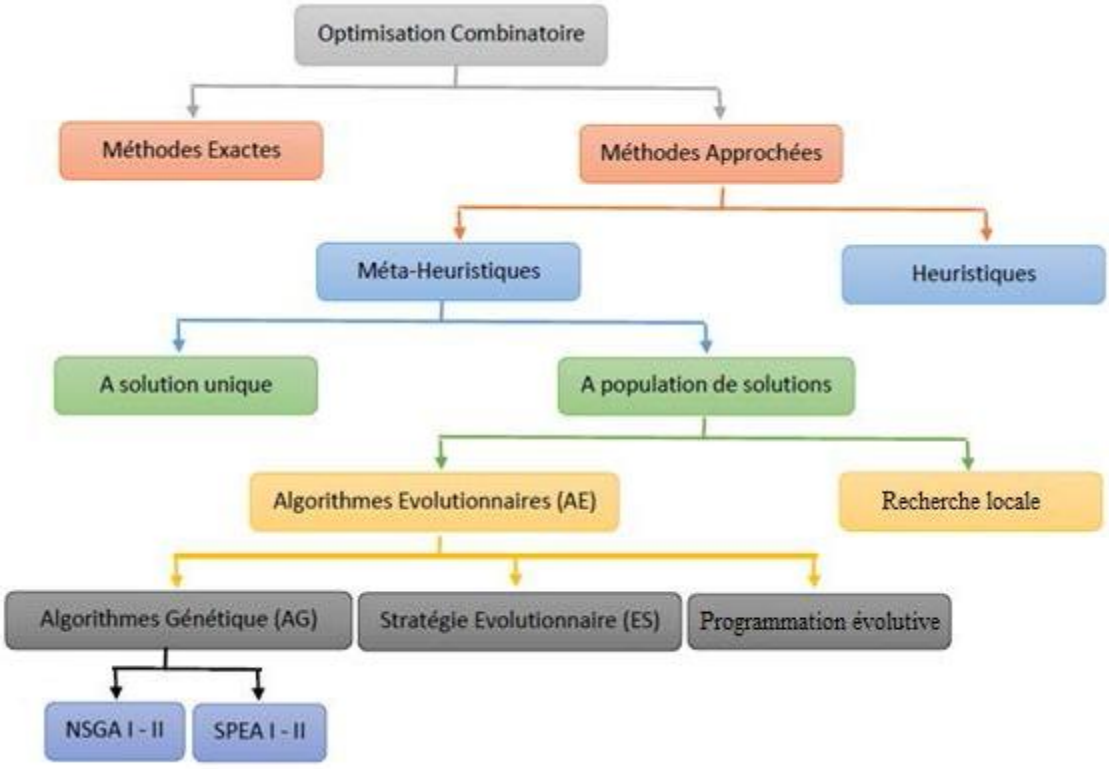

Figure 12: Classification partielle des méthodes d'optimisation [46]

La première classe de métaheuristiques présentées regroupe les méthodes utilisant les principes de la recherche locale. Ces méthodes résolvent le problème d'optimisation de manière itérative. Elles font évoluer la configuration courante (solution) en la remplaçant par une autre issue de son voisinage, ce changement de configuration est couramment appelé un mouvement **[44]**.

Quant à la deuxième classe des métaheuristiques, il y a les algorithmes évolutionnaires (AE) **[47].** Ils sont présentés en 3 grandes catégories : les algorithmes génétiques AG **[48, 49],** les stratégies d'évolution SE **[50]** et la programmation évolutive PE **[51].** Dans notre cas d'étude, on s'intéresse aux AGs.

Une hybridation entre ces méthodes évolutionnaires peut être appliquée pour donner une meilleure performance **[52].** A titre d'exemple, substituer la fonction de mutation de AG avec la recherche locale **[53]**, utilisé la méthodes Pareto Simulated Annealing (PSA) **[54]** qui combine l'algorithme génétique avec le recuit simulé.

Les algorithmes évolutionnaires (dits Evolutionary Computation), sont une famille d'algorithmes s'inspirant de la théorie de l'évolution « darwinienne » pour résoudre des problèmes divers **[55]**. Selon la théorie du naturaliste Charles Darwin, énoncée en 1859 **[56]**, l'évolution des espèces est la conséquence de la conjonction de deux phénomènes : d'une part la sélection naturelle qui favorise les individus les plus adaptés à leur milieu à survivre et à se reproduire, laissant une descendance qui transmettra leurs gènes et d'autre part, la présence de variations non dirigées parmi les traits génétiques des espèces (mutations).

La **figure 13** ci-dessous décrit le principe de fonctionnement d'un algorithme évolutionnaire type. Chaque étape de l'algorithme correspond à une nouvelle instance générer pour constituer une population de solutions. Une valeur initiale est associer à chacune de ces solutions de la population. Une sélection de parents est faite pour produire de nouveaux enfants par le biais des opérateurs d'adaptation de AG. Ce processus est itéré jusqu'à ce que la condition d'arrêt est atteinte **[55].**

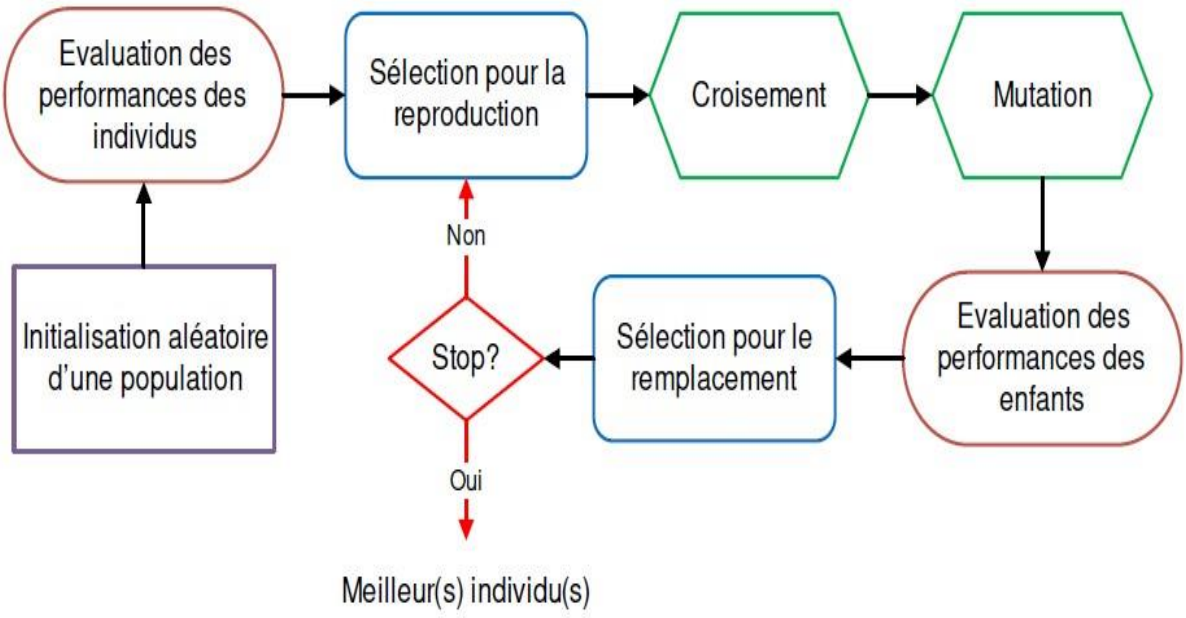

Figure 13: Principe de base de l'algorithme évolutionnaire (AE) [46]

5.2. Les Algorithmes Génétiques AG

Les algorithmes génétiques (AG) sont présentés comme la technique la plus populaire utilisée par les AEs**.** Un individu est présenté par un chromosome, et un groupe de chromosomes forment un ensemble de solutions appelé population **[48].**

5.2.1 Les caractéristiques d'un algorithme génétique

Les AG sont des algorithmes qui opèrent sur une population. Chaque algorithme génétique à les éléments de base qui le définir **(figure 14).** Citons :

Chromosome ou Individu : est une représentation d'une solution parmi plusieurs. Elle peut être codée sous trois forme : binaire (où chaque caractéristique de l'individu est un gène codé en une suite de bits de 0 et 1), gray (où une incrémentation d'un bit dans la valeur d'un chromosome est faite par rapport à son voisin dans la population) **[57],** ou réel (ici les caractéristiques du chromosome sont représentées par un vecteur de valeurs réel) **[58].**

Population : est l'ensemble des solutions trouvé ou généré.

Fonction Fitness : est la fonction d'évaluation d'une solution par rapport à son objectif.

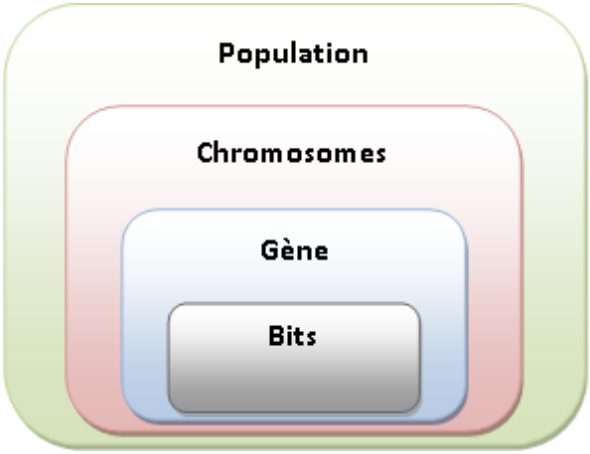

Figure 14: Représentation chromosomique [59]

5.2.2 Les paramètres d'un algorithme génétique

Le processus d'un algorithme évolutionnaire, particulièrement AG suit certaines étapes pour arriver à une solution potentielle. La **figure 15** décrit ces principales étapes tandis que son fonctionnement est présenté dans le pseudo code qui suit l'illustration.

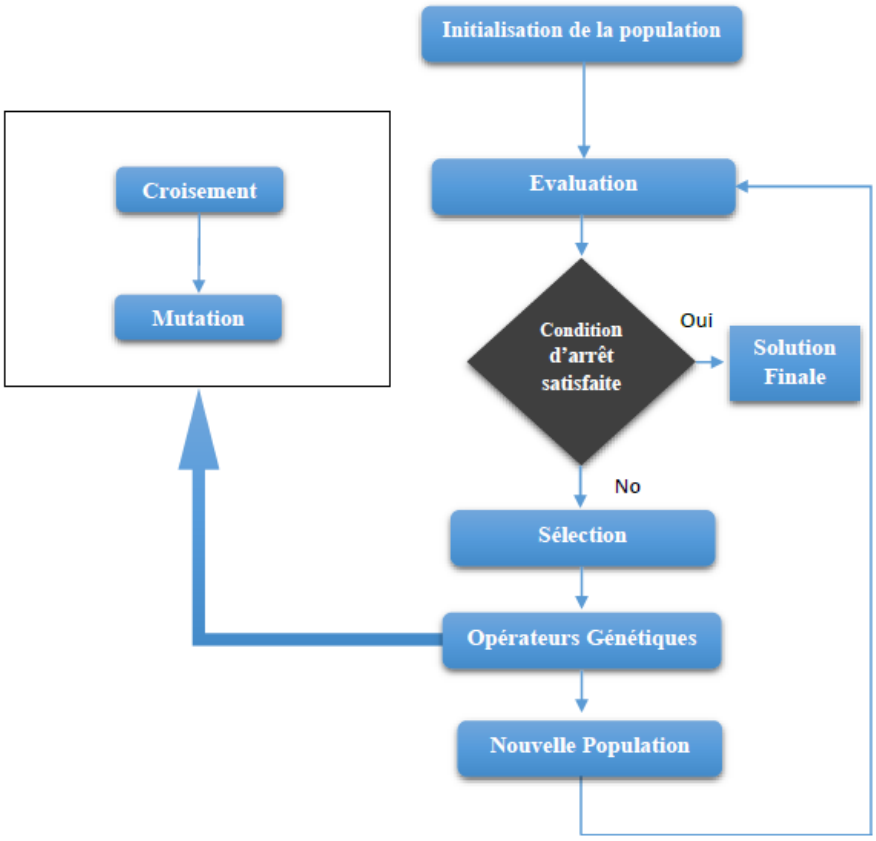

Figure 15: Fonctionnement de l'algorithme génétique [59]

Pseudo code d'un Algorithme Génétique

Initialisation de la population
Évaluation des fonctions objectives - Calcul de l'efficacité
Pour i= 1 to Max Itérations
    %Sélection aléatoire, Sélection proportionnelle à l'efficacité
    Croisement, selon une probabilité
    Mutation, selon une probabilité
    Évaluation des fonctions objectives
    Calcul de l'efficacité
    %Fusion des populations, ou remplacement
Fin Pour

Initialisation : un ensemble de N solutions est généré aléatoirement formant la population initiale des solutions potentielles.

- **Evaluation** : une valeur de fitness est associée à chaque solution de la population afin de mesurer sa pertinence.

Reproduction ou sélection : une des méthodes de sélection est invoquée pour faire la reproduction de N nouveaux individus. Les individus ayants les meilleurs fitness auront plus de chance d'être sélectionnés. La loterie biaisée de **[#48]** qui consiste à dupliquer les individus de la population dont les performances sont relativement bonnes.

La méthode de rang de **[60]** qui consiste à garder les meilleurs individus (selon leurs fitness) d'une population Pt pour ensuite les introduire dans la prochaine population Pt+1. C'est une façon de protéger des solutions potentielles de la disparition lors de la phase de sélection. C'est une méthode performante, mais dans certains cas, le manque de diversité dans les solutions provoque la convergence prématurée ce qui représente l'inconvénient majeur de cette méthode de sélection.

Ou, la sélection par tournoi de **[61]**, cette méthode est une forme de duel entre deux individus, celui qui domine l'autre fonction de sa fitness est sélectionné pour être réintroduit dans la nouvelle population, le perdant est écarté. L'opération est répétée jusqu'à ce que la nouvelle population de n individus soit pleine.

- **Opérateurs génétiques** : qui sont les opérateurs de reproduction dans AG. Le croisement qui divise les vecteurs de valeurs des deux parents choisis (solutions valides) pour la reproduction en un seul point **(figure 16.a) [57]** ou en deux points **(figure 16.b) [62].** Les parties de chacun des parents est changées selon une probabilité de croisement $Pc \in \{0.5, 0.9\}$ afin de former de nouveaux enfants. La mutation qui est un opérateur de diversité évite la convergence prématurée, en couvrant tous les points de l'espace de recherche (**figure 16.c**). Elle choisit aléatoirement un individu et lui modifie un gène selon une probabilité $Pm \in \{0, 1\}$.

- **Remplacement :** les solutions trouvées pendant l'exécution des opérateurs génétiques sont soit :

- Remplacer totalement dans la population initiale,
- Remplacer un pourcentage des solutions qui appartient à la population initiale par les nouvelles solutions.

- **L'archive de Pareto** : l'ensemble des solutions non dominées trouvées pendant la recherche sont préserver et archiver dans une seconde population appelé « Archive Pareto » afin de ne pas les perdre. Une mise à jour est faite à chaque fois une nouvelle solution non dominante est apparu.

- **L'élitisme :** les solutions de l'archive ne sont pas seulement stockées de façon permanente mais qui participent à la phase de sélection pour la reproduction de nouveaux bonnes solutions.

La boucle de l'algorithme génétique tourne jusqu'à ce que la condition d'arrêt soit satisfaite (nombre d'itérations ou nombre de générations maximale, ou bien une convergence entre les solutions est remarquée).

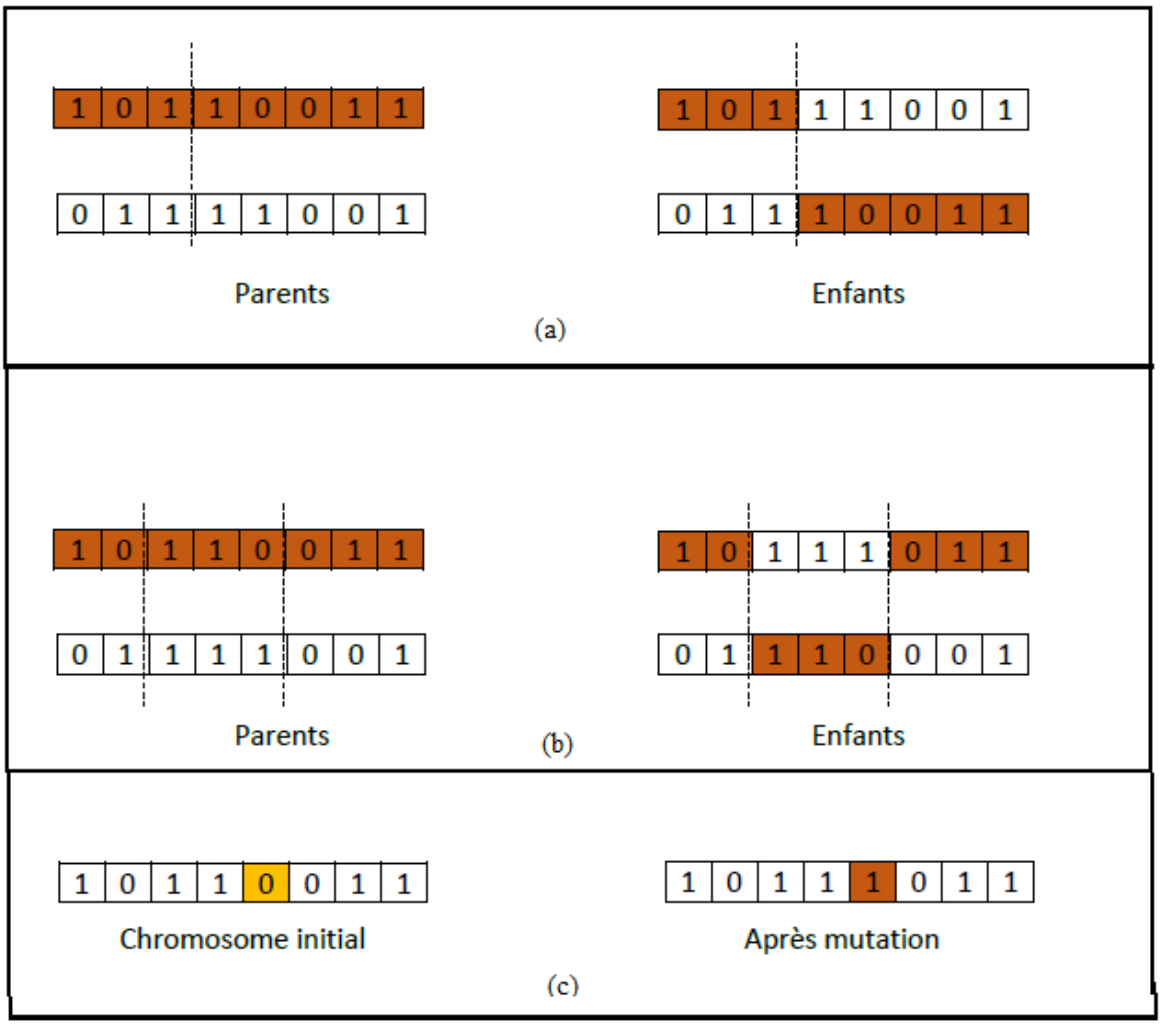

Figure 16: Exemple d'opération: (a) Croisement un point, (b) Croisement deux point, (c) Mutation [59]

6. Exemples de métaheuristiques

Deux métaheuristiques (NSGA-II et SPEA-II) sont illustrées et expliquées dans les paragraphes suivants, afin de comprendre mieux le fonctionnement des algorithmes génétiques (les opérateurs génétiques, le travail sous population et le principe de Pareto).

6.1. Non dominated Sorting Genetic Algorithm II (NSGA II)

L'avancement important du domaine de l'optimisation a permet de cerner les erreurs de la première version du NSGA **[63]** en donnant la version améliorée NSGA-II **[64].** Parmi les principales limites prisent en considération pour améliorer cette algorithme, on a :

- **La complexité de calcul** qui prend beaucoup de temps à trier les solutions de la population. Elle était de $O(mN^3)$ à sa première version mais qui s'est diminuée jusqu'à $O(mN^2)$ avec la nouvelle méthode de tri rapide (où m nombre d'objectifs et N la taille de la population).

• **Le non élitisme** qui provoque la perte des bonnes solutions potentielles donc une convergence prématurée de l'algorithme vers un optimum local. La solution donc est d'intégrer l'élitisme (intégrer les fronts obtenus dans les populations qui suivent) via l'utilisation de la distance de crowding **(figure 17)** qui assure les bonnes solutions du front ainsi que la sélection par tournoi qui garantit une diversification dans les solutions obtenues.

• **Le besoin de l'intervention humaine** vu que l'algorithme utilise une technique qui s'appelle la technique de sharing et qui nécessite de spécifier le paramètre de sharing (**σ**), d'où l'intervention humaine.

Fitness sharing : de **[65]** qui est une technique de partage qui modifie l'espace de recherche en réduisant le rendement dans les régions relativement peuplées. Elle diminue les fitness des éléments semblables de la population afin d'assurer une diversité.
Où : $fi' = fi / mi$,
- fi la fitness de l'élément i,
- mi la niche qui contient un nombre d'individus qui partagent la même fitness que fi.
- $mi = \Sigma sh\ (di, j)$ la fonction de sharing (sh) mesure le degré de similarité entre les individus i fixé et j allant de 1 jusqu'à N

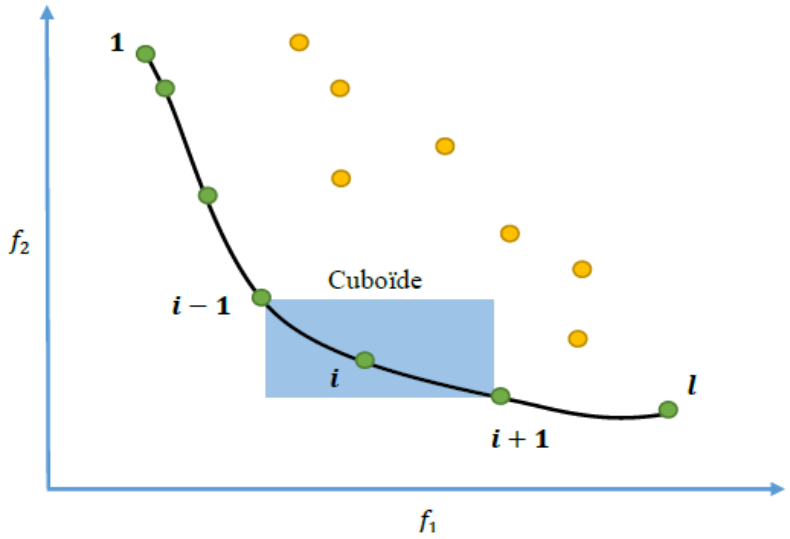

Figure 17: Distance de crowding [59]

Les différents fronts de compromis de la population sont classés selon la dominance de Pareto. Une valeur de ranking de non-dominance est affecter à chaque individu de la population. Cette nouvelle version élitiste de cet algorithme intègre un opérateur de sélection basé sur le calcul de "crowding distance", afin de choisir les bons individus pour la sélection, le croisement et la mutation (la densité de la région autour d'un individu) **[64]**.

Le fonctionnement de l'NSGA-II assure qu'à chaque nouvelle génération, les meilleurs individus rencontrés soient conservés. Pour comprendre le fonctionnement de l'algorithme, plaçons-nous à la génération t, comme le montre la **Figure 18**. Deux populations (Pt et Qt de taille N) coexistent. La population Pt contient les meilleurs individus rencontrés jusqu'à la génération t, et la population Qt est formée du reste d'individus issus des phases précédentes de l'algorithme.

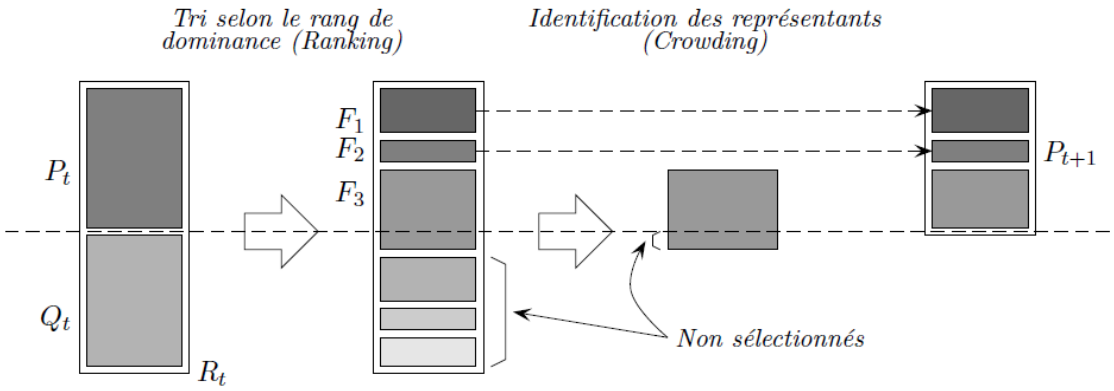

Figure 18: Fonctionnement du NSGA-II [59]

- La première étape consiste à créer la population Rt = Pt U Qt et appliquer une procédure de ranking pour identifier les différents fronts Fi de solutions non dominées.

- La deuxième phase consiste à construire une nouvelle population Pt+1 contenant les N meilleurs individus de Rt. Il faut pour cela inclure intégralement les meilleurs fronts Fi tant que le nombre d'individus présents dans Pt+1 est inférieur à N. Il reste donc à ce stade N - |jPt+1j| individus à inclure dans Pt+1. Pour cela, une procédure de crowding est appliquée sur le premier front Fi non inclus. Les N - |jPt+1j| meilleurs individus au sens de cette procédure de crowding sont insérés dans Pt+1.

- La troisième phase consiste à remplir la population Qt+1. Il suffit alors d'utiliser les opérateurs de sélection, de croisement et de mutation sur les individus de Pt+1, puis insérer les descendants dans Qt+1.

Dans ce qui suit un pseudo algorithme de NSGA-II est présenté pour reprendre toutes les étapes décrites ci-dessus.

Pseudo code de l'algorithme N.S.G.A-II

Initialiser les populations P_0 et Q_0 de taille *N*
Tant que critère d'arrêt non renconté faire
- Création de $R_t = P_t \cup Q_t$
- Calcul des différents fronts F_i de la population R_t par un algorithme de "ranking"
- Mettre $P_{t+1} = \{\}$ et i = 0
- Tant que $|P_{t+1}| + |F_i| < N$ faire
  $P_{t+1} = P_t \cup F_i$
  i = i + 1

FinTantQue
- Inclure dans P_{t+1} les $(N - |P_{t+1}|)$ individus de F_i les mieux répartis au sens de la distance de "crowding"
- Sélection dans P_{t+1} et création de Q_{t+1} par application des opérateurs de croisement et mutation

FinTantQue

6.2 Strength Pareto Evolutionary Algorithm II (SPEA II)

La méthode SPEA-II implémentée par **[66]** est l'amélioration apportée d'un algorithme évolutionnaire élitiste (SPEA de **[67]**). La nouvelle version améliore les performances de l'algorithme considérablement où elle se diffère de son prédécesseur sur les points suivants :

- Le processus d'attribution de fitness est amélioré de telle façon que pour chaque individu, le nombre d'individus qui le dominent et ceux dominés par ce dernier soit comptabilisé.

- La technique d'estimation de densité, où la technique du « k plus proches voisins » est utilisé, afin d'accentuer l'aspect précision durant l'exploration.

- Une nouvelle méthode appelée **« trunction method »** est utilisé dans l'archive pour garantir la préservation des solutions extrêmes (les solutions des frontières).

6.2.1 Le processus de sélection :

Deux type de sélection est utilisé dans cet algorithme, la sélection environnementale et parentale.

6.2.1.1 Sélection environnementale :

Un archive qui contient les solutions non-dominées est actualisé par les meilleurs éléments trouvés lors de la recherche. Un membre d'archive est supprimé en cas de surcharge (taille archive > N) ou bien, parce qu'une solution qui le domine vient d'être repérée dans la population.

L'actualisation de l'archive de SPEA-II diffère de celle de la première version en deux points :
1-la taille de l'archive est toujours la même,
2-la nouvelle méthode de troncation préserve les solutions extrêmes.

La première étape de la sélection environnementale consiste à copier tous les individus non-dominés de l'archive et de la population vers l'archive de la prochaine génération.
Si le nombre des individus non-dominés est identique à la taille de l'archive, alors la première étape de la sélection environnementale est terminée. Sinon, si la taille de l'archive est supérieure à la taille de la population (N) alors une méthode de troncation est appliquée itérativement jusqu'à ce que |archive| = N, sinon l'archive sera remplie avec les meilleures solutions dominées.

6.2.1.2 Sélection parentale :

Elle consiste à créer une nouvelle population de *N* individus (solutions) par l'utilisation d'une méthode de sélection appropriée, par exemple : tournoi, ranking, …etc.

6.2.2 Attribution de fitness :

Deux valeur de fitness sont prise en compte dans cet algorithme. La première valeur fait un ordre partiel de l'ensemble des individus en attribuant un rang de dominance, et une autre valeur de ceux qui dominent cet individu (technique d'estimation de densité de k plus proches voisins inspirer de la méthode "***k-th* nearest neighbor**" de **[68]**) est utilisé pour donner une meilleure précision durant l'exploration des nouvelles solutions.

6.2.3 La technique du clustering :

C'est une méthode qui opère différemment de celle de la première version **[59]**. Où elle permet de résoudre le problème de favorisation de certaines régions dans l'espace de recherche. Un front non-dominé peut être extrêmement large. Par conséquent, la distribution des solutions sur le front est non uniforme. La technique de clustering consiste donc à groupe p éléments en q groupes (clusters) avec $q<p$. Puis, dans chaque groupe, une solution représentative est déterminée, le reste du groupe est supprimé.

Explication : Tout d'abord, une population P_0 est générée aléatoirement et l'archive P_0 est initialisé. La première étape consiste à attribuer des valeurs de fitness à tous les individus de l'archive et de la population ($Pt \cup \overline{Pt}$). Puis, à l'aide de la sélection environnementale l'archive est actualisé afin de contenir les meilleures solutions possibles. Après cela, une sélection avec tournoi est effectuée sur l'ensemble ($Pt \cup \overline{Pt}$). Et enfin, les opérateurs génétiques (croisement, mutation) sont appliqués sur la population initiale (individus sélectionnés), donnant en résultat une nouvelle population P_{t+1}. Ces opérations sont répétées tant que le critère d'arrêt n'est pas rencontré.

La **figure19** ci-dessous illustre le fonctionnement et les étapes de l'algorithme SPEA-II.

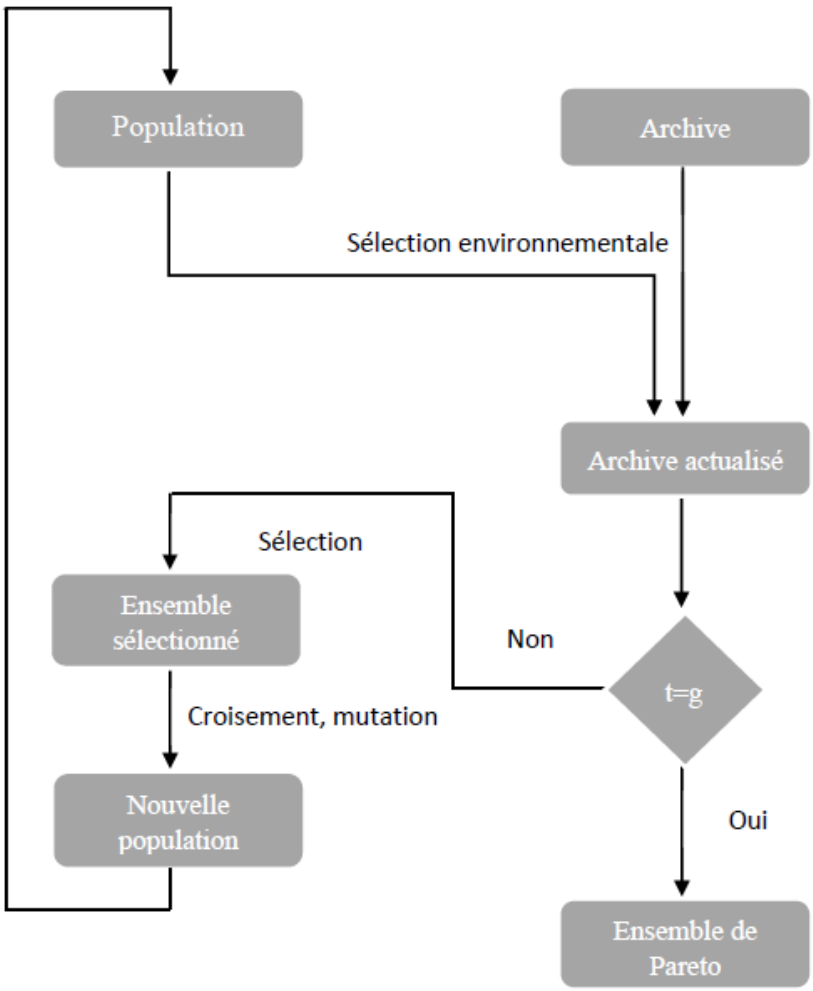

Figure 19: Fonctionnement du SPEA-II [59]

Pseudo code de l'algorithme SPEA-II

Début
SPEA-II (N, g)
P_0 =générer-population-initiale (N)
$P_0 = \emptyset$ initialiser l'archive
t=0 initialiser le compteur de génération
Pour t= 1 jusqu'à g
  Attribuer-fitness (P_tU $\bar{P}_t$) attribuer les fitness archive + population
  Sélection-environnementale ($\bar{P}_t$) actualiser l'archive (selon 3 cas possible **titre 6.2.1.1**)
  S= Sélection-parentale($\overline{P_{t+1}}$) sélectionner les parents (tournoi)
  P_{t+1}=Opérateur-génétique(S) appliquer croissement & mutation pour donner P_{t+1}
t=t+1 ;
Fin

7. Conclusion

Nous avons présenté dans ce chapitre quelques méthodes d'optimisations combinatoires. Puis, nous avons décrit les différentes classifications des méthodes correspondantes. Les méthodes exactes peuvent être appliquées pour des problèmes de petite taille en donnant toujours des solutions optimales.

Pour les problèmes de grande taille, les méthodes approchées sont plutôt utilisées. Deux algorithmes multi-objectives ont été décrits, en expliquant le fonctionnement de base de chacun, ainsi que les notions relatives à AG et aux problèmes multi-objective.

Le NSGA-II avec sa technique de crowding et de tri rapide permet une exécution rapide dans un temps raisonnable, élitiste et efficace. Et le SPEA-II avec la technique de clusetring adaptée donne plus de performances dans l'exploration des nouvelles solutions.

Une étude sur la phase de mapping est présentée dans le chapitre suivant, afin de comprendre mieux les concepts liés aux réseaux sur puce et la résolution des problèmes d'optimisations combinatoire (le placement des composants).

Chapitre 3 Mapping et réseau sur puce 3D

1. Introduction

La conception d'un réseau sur puce efficace qui satisfait les nouvelles exigences des utilisateurs en terme de performances est un processus nécessitant plusieurs phases. Le mapping est une phase qui consiste à l'étude du placement des composants d'une application sur l'architecture du réseau sur puce.

2. La phase de Mapping

La phase de mapping sert à placer les composants IPs d'une application (représenter par un graphe d'application) sur chaque tuile d'une architecture donné (représenter par un graphe d'architecture), dont l'objectif de maximiser (optimiser) les performances du système **(figure 20)**.

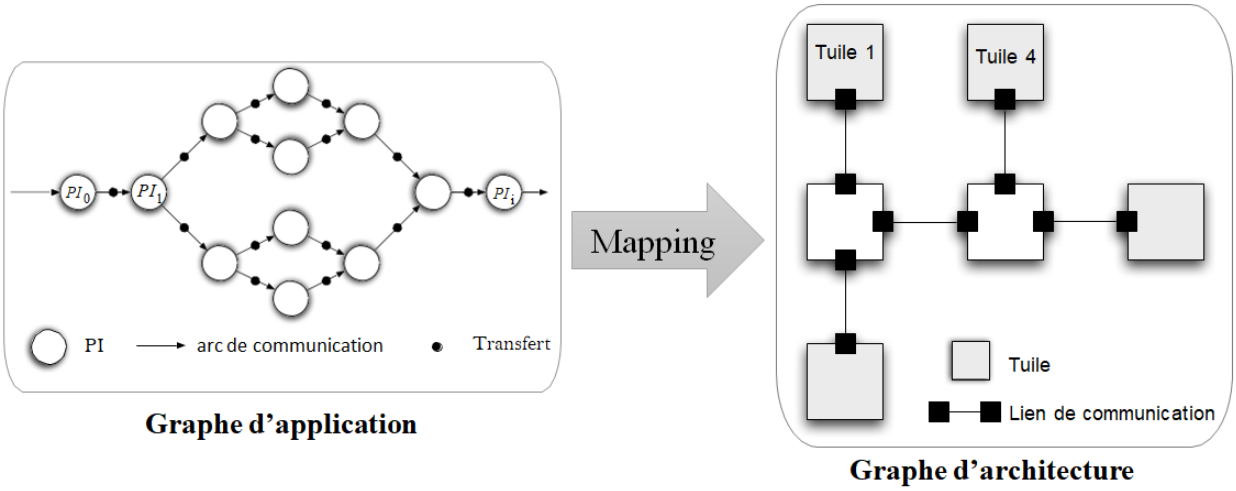

Figure 20: Exemple du principe de mapping

La **figure 21** illustre un Framework de classification des différentes techniques de mapping abordées dans la littérature **[40]** :

- Le choix entre la méthode heuristique ou la méthode métaheuristiques et l'objectif visé (mono ou multi-objectif).
- Le mapping dynamique qui est effectué durant l'exécution de l'application, ou l'approche statique qui est fait pendant la conception.
- Intégrer la phase de mapping avec une autre phase de conception tels que : l'ordonnancement, routage, floorplannig.
- Ou selon la topologie envisagée (régulière ou personnalisée).

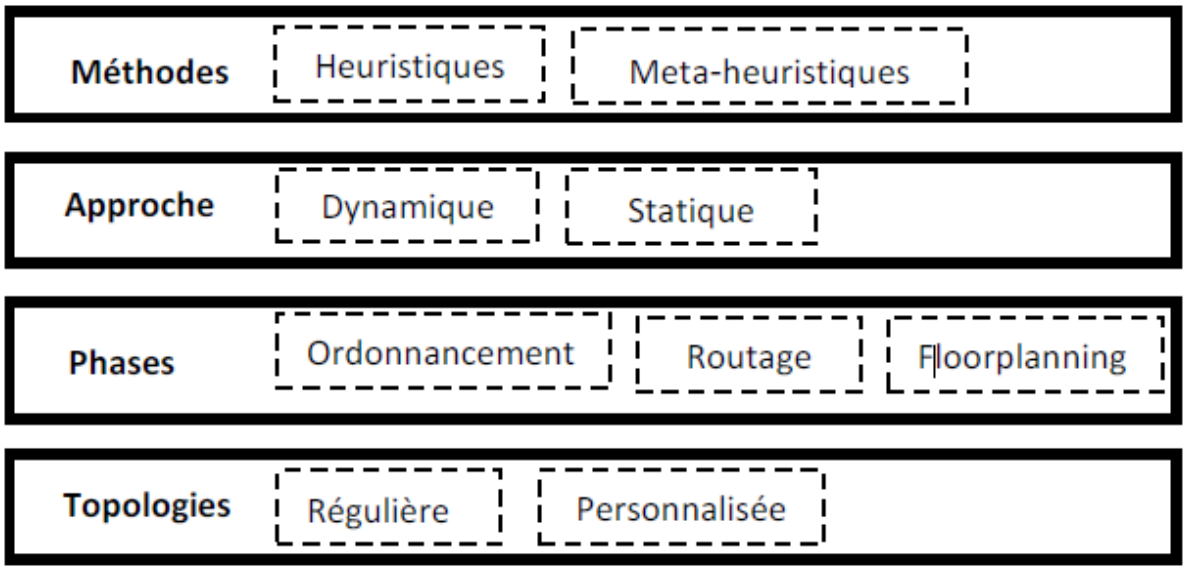

Figure 21: Les critères de classification des stratégies de mapping [29]

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. Les deux solutions métaheuristiques et les heuristiques peuvent être transformatives ou constructives **(figure 22) [69].**

La méthode transformatrice transforme certaines solutions de mapping existantes afin de les améliorer. Les méthodes constructives, des solutions partielles sont générées successivement, et à la fin la solution de mapping complète est obtenue.

Une solution constructive peut être sans amélioration itérative ou avec amélioration itérative. Une solution constructive sans amélioration itérative mappe les IPs sélectionnés (un par un ex. la technique Spiral) d'un graphe d'application sur le graphe de l'architecture du NoC, où il n'y aura pas de changement de position d'un IP une fois le placement effectué.

Par contre, les solutions constructives avec amélioration itérative font la première solution du mapping (considérer comme solution initial), ensuite, le placement des IPs change par rapport à la solution précédente (meilleure position qui donne les bonnes performances au IP, exemple la technique Onyx) jusqu'à l'arrive à la solution finale.

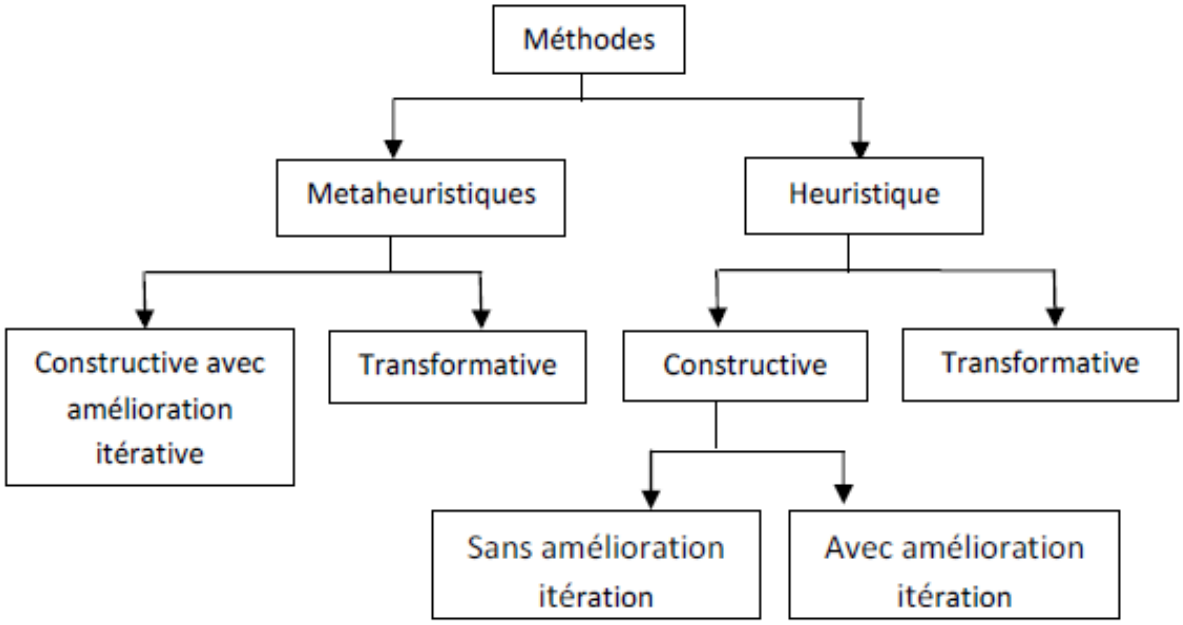

Figure 22: Classification selon la méthode choisie - heuristique ou métaheuristique [40]

Un exemple d'algorithmes métaheuristiques et heuristiques est présenté dans le **tableau 3** selon le classement en méthodes transformative et méthodes constructives avec ou sans amélioration itérative.

Tableau 3: Exemple des méthodes heuristiques et métaheuristiques [29]

Métaheuristiques		**Heuristiques**	
Transformative	Constructive avec amélioration itérative	Constructive sans amélioration itérative	Constructive avec amélioration itérative
SPEA **[70 - 72]** NSGA **[73]** AG **[74 - 76]**	RS **[77, 78]**	CHMAP **[79]** Crinkle **[80]** Spiral **[81]**	Onyx **[82]**

Parmi les métaheuristiques cités dans la littérature, il y a le Recuit Simulé (Simulated Annealing-SA) **[69]**. Il a été comparer avec l'algorithme déterministique Branch-and- Bound de **[83]** pour résoudre le problème de mapping avec la phase de routage. Les deux algorithmes visent à minimiser le coût des communications en utilisant la topologie 2D-Mesh.

Une amélioration du SA est présentée dans **[78]** pour accélérer le mapping des IPs sur les tuiles**.** Le nouvel algorithme à base de cluster appelé Cluster-based Simulated Annealing – CSA regroupe les IPs qui communiquent le plus dans un même cluster. Pour chaque cluster, une région du réseau sur puce est allouée.

Une heuristique appelée la technique SPIRAL est réalisée par **[81].** Elle affecte les IPs d'une application sur les ressources d'une architecture réseau sur puce d'une façon optimale, en classant l'ensemble des éléments (tâches de l'application) selon leurs degrés de communication, en démarrant du centre de l'architecture jusqu'à l'arriver aux tuiles frontières **(figure 23)**.

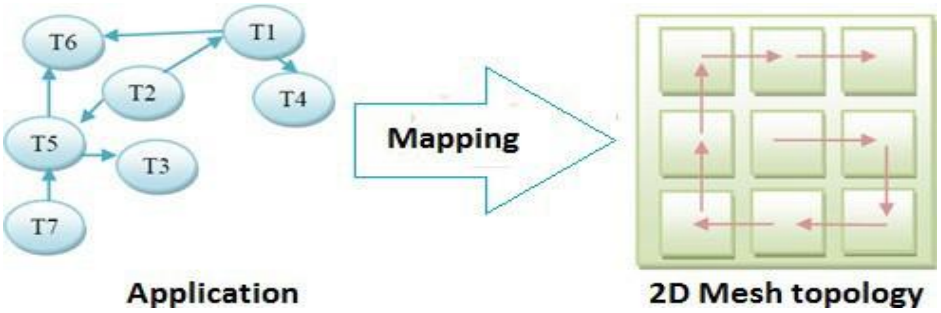

Figure 23: Mapping en utilisant la technique SPIRAL [29]

Une heuristique similaire appelé Crinkle dans **[80]** basés aussi sur une liste de priorités entre les composants, hors que cette technique place les IPs de l'application en forme du zigzag, en commençant par les coins de l'architecture 2D-Mesh (**figure 24**).

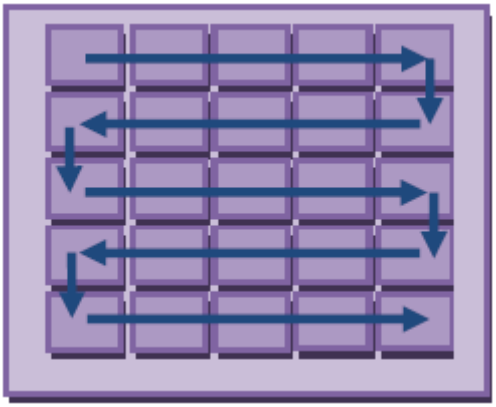

Figure 24: Le mapping des PIs suivant la technique de zigzag [80]

Une autre heuristique appelé Onyx a été proposée par Janidarmian et al. **[82]**. Elle permet de placer les IPs qui communiquent le plus en premier, selon la priorité des IPs ainsi que les tuiles qui est affecter par la fonction highest-priority (**figure 25.a**). Le IP qui communique le plus avec celui déjà placé est positionné dans une tuile libre la plus proche (avec une distance d'un saut, si non deux sauts, etc.) (**figure 25.b**). La recherche d'une tuile libre se fait dans un chemin sous forme losange suivant 4 mouvements (**figure 25.c**). Le type du mouvement donne des priorités différentes aux tuiles.

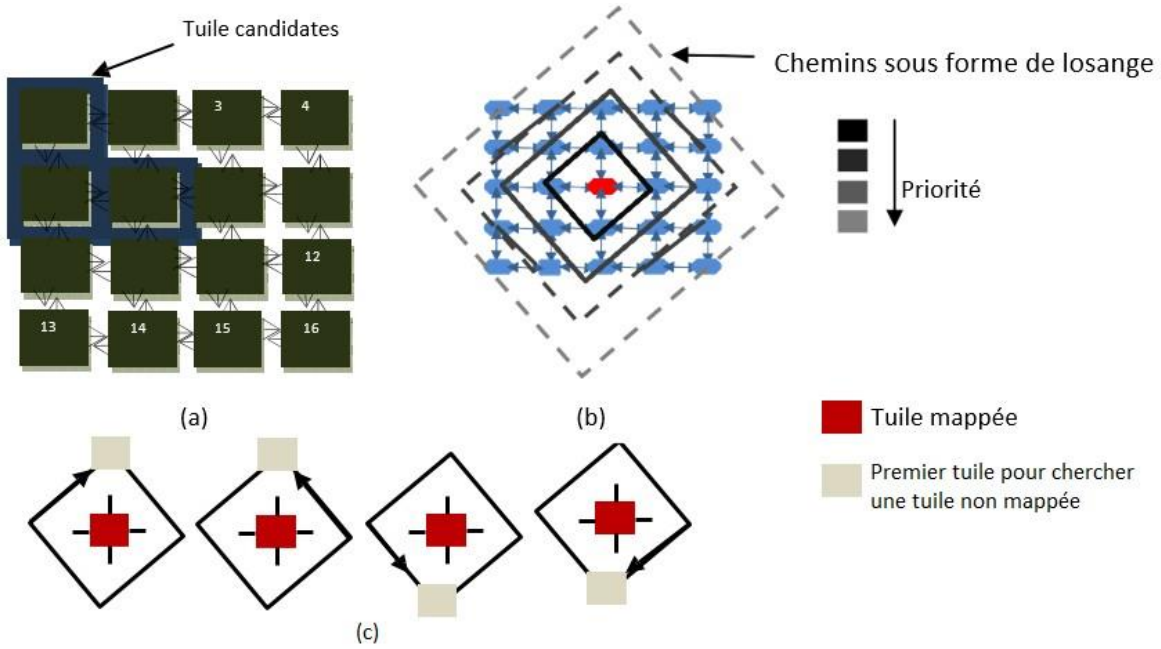

Figure 25: (a) Tuiles candidates pour le placement du premier PI, (b) Concept de chemins sous forme losange, (c) Définition de 4 mouvements [82]

Une seconde technique base sur la notion de priorité est développé dans **[79]** dit CHMAP (CHain-MAPping). Elle fait en sorte à utiliser le graphe de communication (**figure 26.a**) pour créer des chaines trier à base des valeurs de bande passante des PIs (**figure 26.b**). Les chaînes finales optimisés (**figure 26.c**) sont mappées sur une architecture réseau sur puce 2D maillée, en commençant par la chaîne dont la bande passante est plus grande afin d'optimiser le mapping.

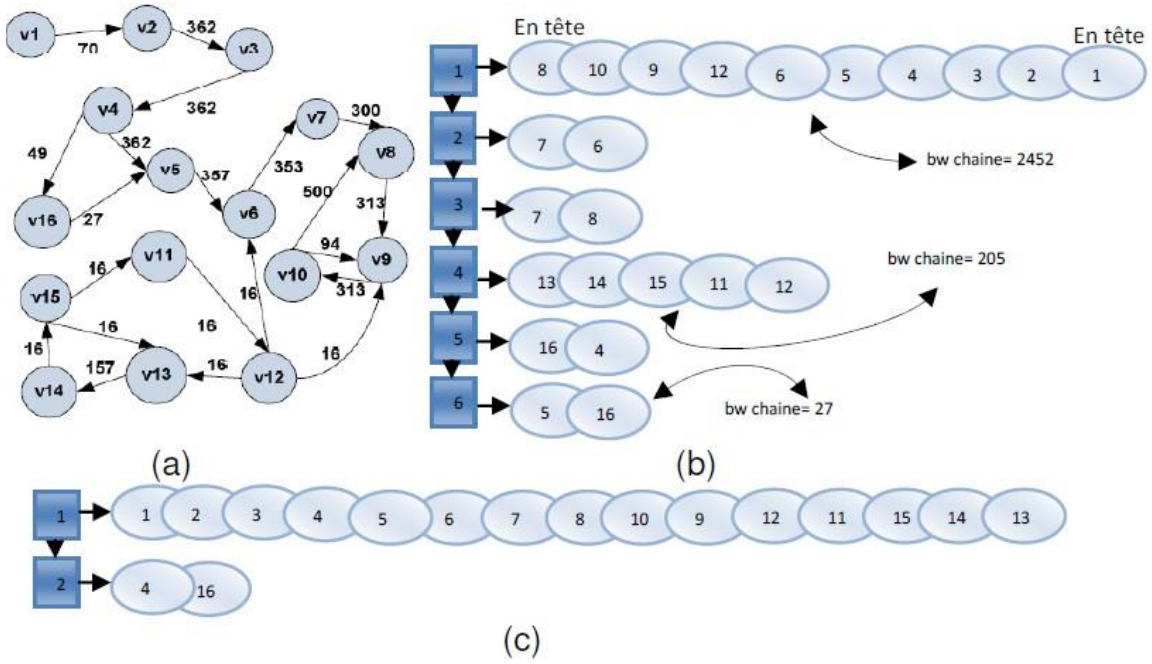

Figure 26: (a) Graphe de communication VOPD, (b) Chaine initiale, (c) Chaine finale [79]

3. Le mapping avec d'autres phases

La combinaison des phases pendant la conception des réseaux sur puce peut donner de grande avantages en performances et en temps de conception. D'autres phases peuvent être combinées avec la phase de mapping : l'ordonnancement, le routage, le floorplanning et même le choix de la topologie.

3.1. Ordonnancement

Les applications exécutées dans le contexte d'un réseau sur puce sont divisées en tâches. La représentation d'une telle application est modélisée sous forme de graphe de tâches : les nœuds désignent les tâches et les arcs désignent les relations de précédences entre ces tâches.

L'ordonnancement consiste à choisir pour chaque tâche le IP qui l'exécute et à quel moment. Plusieurs critères peuvent être considérés dans cette phase tels que : le temps total d'exécution, le temps moyen que passe une tâche dans le système, l'énergie consommée par la tâche dans un IP, etc.

La première étape consiste à sélectionner les tâches qui vont être mapper sur les blocks IPs afin d'avoir une optimisation en temps d'exécution de l'application. Quant à la deuxième partie, elle sert à trouver le meilleur mapping de ces IPs sur l'architecture réseau sur puce. Un exemple de combinaison du problème d'ordonnancement et de mapping est étudié dans **[84, 85]**.

3.2. Routage

Un algorithme de routage définit le chemin que doit emprunter un paquet pour atteindre sa destination. Il doit éviter les situations d'interblocage tout en optimisant l'utilisation des liens de communications (optimisation de la consommation de l'énergie) **[86].**

Le but du routage dans **[87, 88]** est de déterminer un acheminement sans interblocage et avec un chemin minimal pour chaque communication établie. L'interblocage se produit lorsque plusieurs paquets se bloquent mutuellement car ils détiennent des ressources demandées et demandent d'autres déjà détenues.

Hu et Marculescu dans **[83]** ont développés l'algorithme Branch-and-Bound. Leur solution cherche à comment mapper les IPs d'une application aux tuiles, d'abords. Après, trouver comment router les paquets de cette application dans le réseau. Les auteurs de la même référence traitent le problème de routage afin de construire des chemins qui garantissent l'absence d'un interblocage (deadlock-free), et qui minimisent au même temps l'énergie totale de communication.

3.3. Floorplanning

L'étape du floorplannig consiste à placer physiquement l'ensemble des composants IPs du circuit sur la puce. Le problème de placement et d'interconnexion revient à savoir les informations nécessaires pour trouver la bonne affectation physique avec un minimum de surface occuper et de nombre de liens utiliser **[1]**.

Le floorplannig peut être divisé en deux parties **[89]** : la première consiste à trouver un placement relatif des différents composants sur une la topologie, et la deuxième étape consiste à trouver les positions exactes de ces IPs afin de minimiser la surface.

Le résultat de la phase de mapping dans **[90]** est utilisé comme une donnée d'entrée pour la phase de floorplanning. Les mêmes auteurs proposent d'abord l'étape de Floorplanning afin de générer une topologie personnalisée **[91]**. D'autres travaux dans ce genre (topologie personnalisée) sont cités dans la section suivant.

3.4. Topologie régulière et personnalisée

La majorité des travaux déjà cités utilisent la topologie régulière 2D-Mesh. Ceci est dû à la facilité de son implémentation et sa capacité de s'adapter à une taille de réseau variable. Cependant, les liens qui existent entre certains IPs qui ne se communiquent pas restent un inconvénient qui influence sur la performance du système en énergie, en surface **[92]**.

L'approche proposée par Chatha et al. **[93]** consiste à partir du placement des IPs **(figure 27.a)**, allouer à chaque IP un routeur parmi les routeurs de ses coins. Ensuite, minimiser le nombre de routeurs (et des ressources non utilisées) pour générer une topologie finale avec un placement final des composants sur le plan physique **(figure 27.b)**.

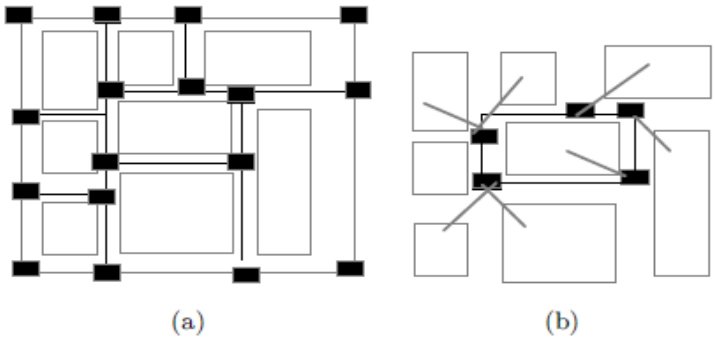

Figure 27: (a) Floorplanning initial, (b) topologies final [93]

La **figure 28** illustre une solution qui démarre avec une topologie régulière 2D-Mesh pour arriver à une topologie irrégulière. Après l'application un certain trafic de donnée sur la topologie initiale **(figure 28 à gauche),** les chercheurs ont remarqué la non utilisation de certaines ressources. D'où, cette dernière a subi un changement en nombre d'éléments et de liens qui composent l'architecture, en donnant au finale une topologie personnalisée **(figure 28 à droite),** avec des performances considérablement amélioré **[12]**.

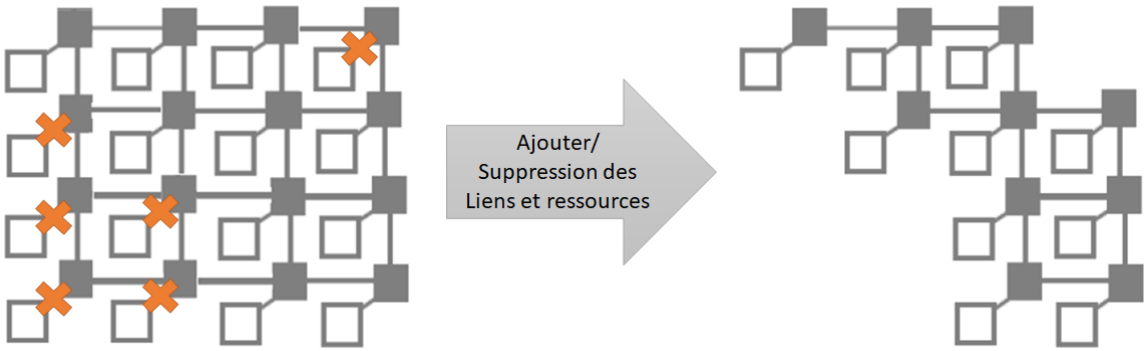

Figure 28: Topologie 2D-Mesh irrégulière

Une autre méthode présentée dans **[94]** détermine les routeurs qui présentent un goulot d'étranglement dans le système, en remédiant à ce problème par l'ajout des liens supplémentaires afin d'alléger la charge et améliorer les performances du système.

D'autres approches personnalisent entièrement l'architecture pour la rendre plus appropriée à une application donnée. Par conséquent, l'architecture sera dépendue de l'application, et n'est pas nécessairement conforme aux architectures régulières (par exemple, 2D maillée, ...) **[29]**.

Un algorithme génétique a été proposé dans **[95]**, dont l'objectif de minimiser la consommation d'énergie et la surface de silicium, en générant plusieurs topologies personnalisées. Il commence à former des configurations (architectures) aléatoirement, 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.

Un récapitulative de comparaison des différents points entre une topologie régulière et autre personnalisée est présenté dans le **tableau 4** de Xinyu Li **[96]**.

Tableau 4: Comparaison entre topologies régulière et personnalisée [96]

Topologie régulière	**Topologie personnalisée**
Maillée, Torus, …	Topologie irrégulière
SoC utilité générale	SoC application spécifique
Routeurs homogènes	Routeurs hétérogènes
Topologies réutilisées	Conception de routeurs réutilisée
Peu de temps de conception	Long temps de conception
Performances basses	Performances élevées
Énergie élevée	Énergie basse

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.

4. Technologie 3D

Noté à présent que toutes les méthodes déjà cité utilisent la topologie en maillage 2D. Le nouveau paradigme NoC introduit, fait appel à l'utilisation de la technologie 3D via les liens de communications verticales nommés (TSV) qui ont une longueur de fils plus cours que les fils horizontaux.

Grâce à ces connexions verticales, les IPs des différents plans peuvent maintenant communiquer plus rapidement, tout en consommant moins d'énergie **[97, 98]**. Il est montré qu'une architecture 3D peut réduire la longueur du fil autant que la racine carrée du nombre de couches empilées **[99]**. Toutefois, le placement de ces liens ainsi que leur nombre qui est coûteux en surface influence considérablement les performances du système.

L'idée d'intégrer les 3D-ICs au sein des réseaux sur puce présente un changement fondamental. Topol et al dans **[100]** ont montrés l'intérêt de cette technologie en analysant les différents défis de fabrication des 3D-IC, où une amélioration dans la consommation d'énergie, un bruit minimal, plus d'emballement de PIs (intégration des IPs dans la même puce) ainsi qu'une réduction des liens d'interconnexions longue a été remarqué. De plus, l'utilisation de cette technologie a permet d'intégrer des composants dissimilaire (hétérogène) sur les systèmes sur puce **[101]**.

Une autre proposition d'utiliser les 3D-IC a été faite par Jacob & al dans **[102]** afin d'améliorer les performances des microprocesseurs en utilisant le concept de pile processeur-mémoire. Leur étude a permis de visualiser que cette intégration à améliorer considérablement les performances du système, par la possibilité d'utiliser des bus plus large (>1024 bits) en transferts de données sur les liens verticaux. Ainsi que, la distance courte entre le processeur et la pile mémoire qui diminue le temps d'accès et le traitement par la suite.

5. Topologie 3D (Hybride, Mesh)

Les chercheurs de **[101]** ont dérivés une variété de topologies 3D Mesh qui est constituer de (m*n) placement de composants (liens plus courts entre les routeur/commutateur). La topologie la plus direct été la 3D Mesh base de NoC (**figure 29.b**)**.** Cette topologie utilise un commutateur à 7 ports : 1 pour l'IP**,** un pour chaque commutateur de haut et en bas et un pour chaque direction (Nord, Est, Ouest et Sud) **(figure 30.a)**.

La deuxième dérivation de la topologie 3D Mesh est appelé maillage empilé. Cette topologie prend l'avantage des distances courtes inter-couches (**figure 29.c et figure 30.b**) grâce aux 3D-IC.
La topologie maillage empilé est un hybride entre un réseau de commutation de paquets et un bus. Elle intègre plusieurs couches du maillage 2D, en les connectant à un bus couvrant toute la distance verticale de la puce.

Comme la distance entre les différentes couches 2D du circuit intégré 3D est extrêmement petite, la longueur totale du bus est également petite, ce qui la met comme un choix approprié pour la communication dans la dimension Z (3D) **[103]**. De plus, chaque bus ne comporte qu'un petit nombre de nœuds (égal au nombre de couches de silicium), ce qui permet de garder une capacité globale sur le bus réduit et de simplifier considérablement l'arbitrage de bus.

Une troisième dérivation du maillage 3D est 3D Mesh cilié. Elle est définie par l'ajout des couches de IPs fonctionnel (**figure 29.d et figure 30.c**) avec une couche (peut être plus) qui contient les commutateurs de la topologie **[101]**. Cette structure considère plusieurs IPs par commutateur (dans leur cas d'étude ils ont considéré deux IPs par commutateur), sauf que les IPs choisis pour le commutateur doivent occuper plus ou moins la même surface de silicium **[101]**.

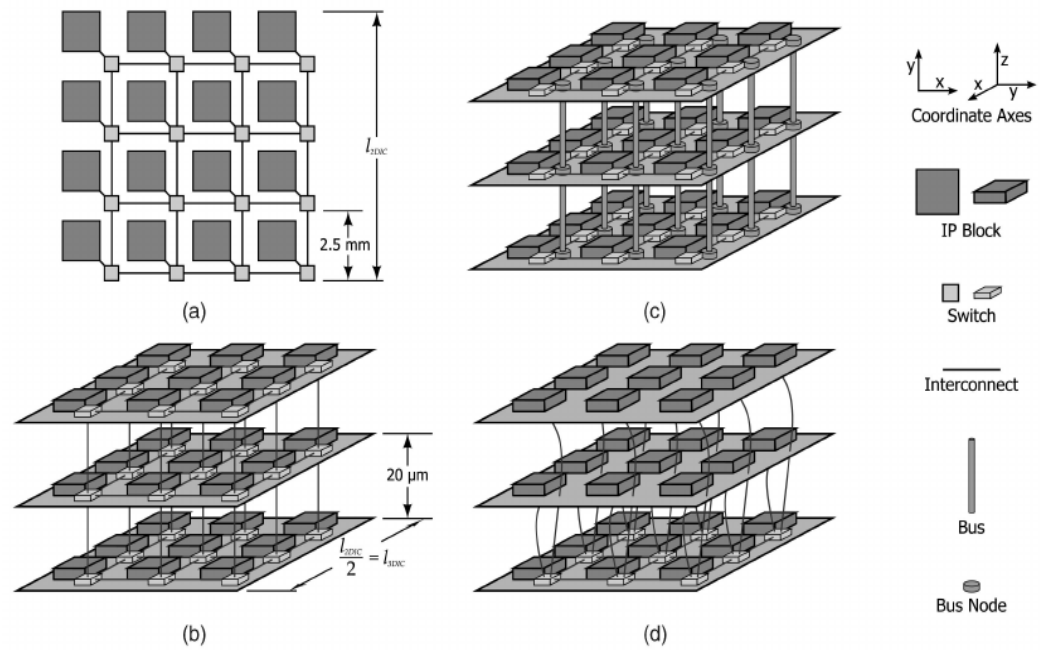

Figure 29: Les architectures Meshs: (a) Mesh 2D, (b) Mesh 3D, (c) 3D Mesh empilé. (d) 3D Mesh cilié [101]

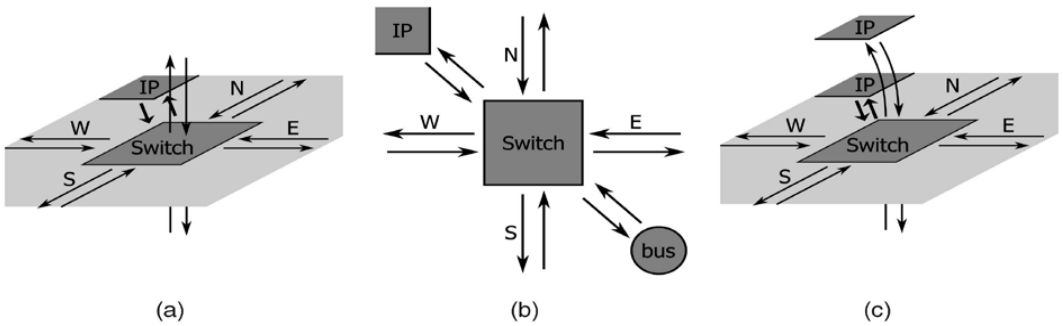

Figure 30: Commutateurs 3D Mesh: (a) Standard, (b) Empilé, (c) Cilié [101]

La même référence **[101]** montre que ce type d'architecture offres explicitement un avantage en terme de dissipation d'énergie, notamment en présence un modèle de trafic spécifique. Cependant, l'inconvénient remarqué dans cette architecture est la bande passante global qui est inférieure à celle d'un réseau sur puce en maillage 3D complet, en raison de la multiplicité des IPs par les commutateurs et de la connectivité réduite.

6. Liens verticaux - TSV

Une proposition a été appliquée sur les diapositives électroniques au début des années 50 pour y arriver à la communication verticale. Le premier essaye été l'empilement d'une puce sur une autre avec des câbles extérieurs à la puce **(figure 31) [104]**. Cependant, la grande occupation de surface par ces liens a obligé les chercheurs d'envisager une alternative d'interconnexion, appelé le Through Silicon Via abrévié en TSV (40% d'énergie économisé après l'application de cette technologie sur des mémoires puce interconnecter par Samsung **[105]**).

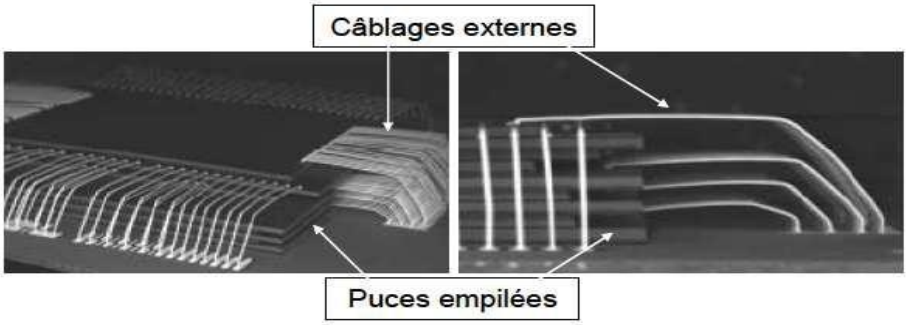

Figure 31: Empilement de plusieurs puces par Wire Bonding [104]

Une exemple illustratif d'une architecture réseau sur puce 3D en utilisant les TSVs est présenté dessous dans la **figure 32.**

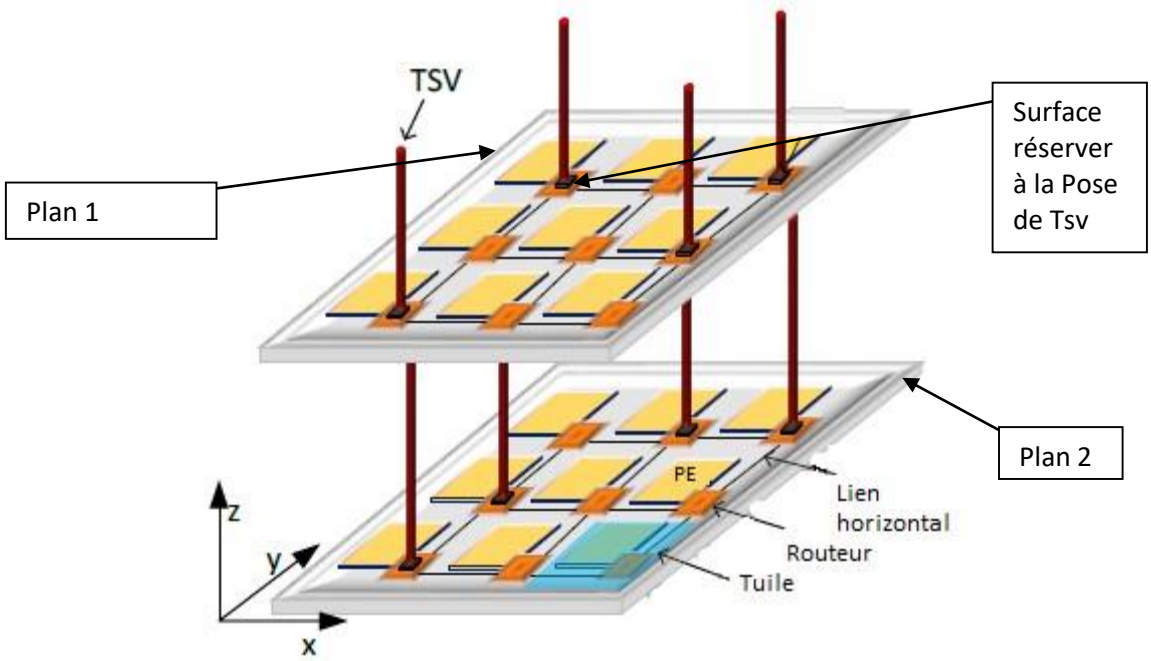

Figure 32: Illustration d'un NoC 3D avec ses composants [107]

Il existe également une autre technologie d'interconnexions, nommée MIVs (Monolithic Inter-tier Via) de taille réduite par rapport aux TSVs (réduction de 8% de surface **[#106]**), mais qui est toujours au stade de recherche.

Une interconnexion à base de TSVs permet de réduire considérablement la latence d'interconnexion par rapport aux interconnexions 2D **(figure 33)**. Néanmoins, cette technologie présente un problème vis à vis la surface occupée par ces derniers. C'est dans ce contexte là que l'idée de réseaux sur puce 3D partiellement connectés verticalement est parvenue, afin d'exploiter l'intégration de la technologie 3D tout en conservant un haut rendement **[29]**.

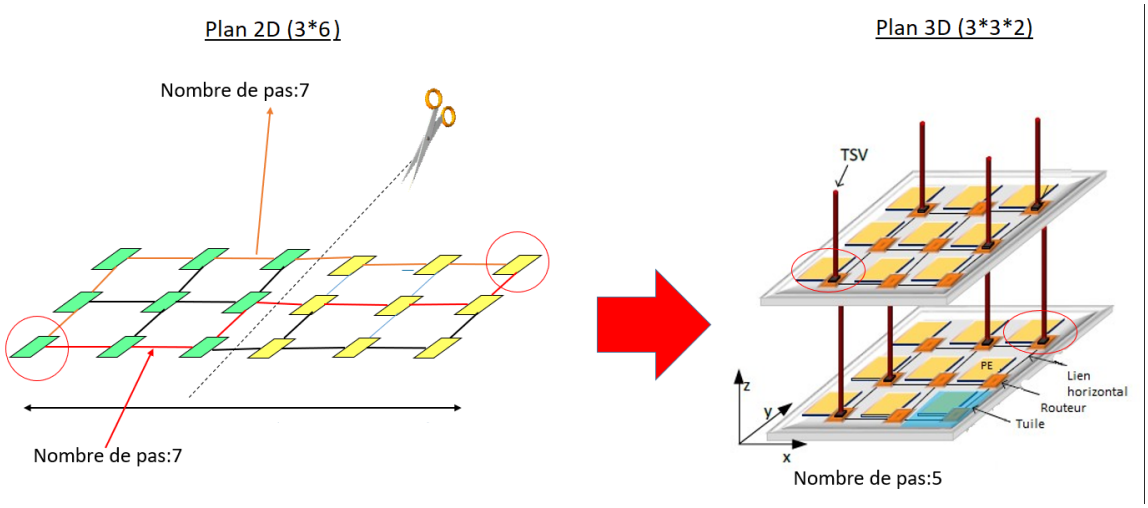

Figure 33: De la Mesh 2D vers Mesh 3D

Xu et al dans **[26]** ont fait des expérimentations sur le placement des liens verticaux avec un nombre de TSVs à 100%, à 50% et à 25%, dans les architectures réseaux sur puce 3D (4*4). Les résultats ont montré une réduction de 5.24% et 2.18% de latence (pour 100% et 50% respectivement) par rapport au 25% de liens. Ce résultat est argumenté par le fait que la topologie à TSV complet le nombre de sauts est moins par rapport à la moitié et à un quart de TSVs.

Pasricha a proposé dans **[108]** une technique de sérialisation pour réduire le nombre de TSVs. L'idée consiste à mettre un seuil de nombre de TSV, s'il dépasse ce seuil, une sérialisation est adoptée afin de réduire la bande passante de certains liens au lieu de les supprimer totalement. Cependant, cette technique entraine une grande latence de transfert de paquets ainsi qu'une grande consommation en énergie.

Une autre étude a permis de présenter l'impact des TSV sur les NoC dans Xiangyu et al **[109].** Ils ont montré que la surface totale consommé par les TSVs augmente chaque fois que la taille du réseau augmente. Dans le cas général, la taille du réseau désigne le nombre de couches verticales (plus de 3 couches peut causer une hausse de température **[110]**).

Des heuristiques ont été proposées par **[29]** afin d'optimiser le nombre et le placement des liens verticaux en minimisant la surface et le coût de communication des IPs d'une application mappée sur une architecture 3D. En revanche, cette étude s'est focalisée sur optimisation des TSVs uniquement (la solution mapping est une donnée entrée), en étudiant ce problème d'un point mono-objectif où leur fitness c'était de calculer le coût de communication en nombre de sauts.

7. Synthèse

Plusieurs travaux de recherche que nous avons cités dans ce chapitre ont indiqués clairement l'avantage de l'utilisation les architecture 3D par rapport au 2D en matière de coût de communication et de surface.

Ces recherches ont souligné aussi l'importance de maintenir un équilibre entre les performances et l'utilisation de la technologie 3D (TSV) vu le coût de fabrication de ces liens pour concevoir un réseau sur puce 3D.

D'après les travaux de littérature déjà cités, les topologies personnalisées peuvent être générées avant ou après la phase de mapping :

Des travaux qui ont générés des topologies irrégulières dès le début du processus de conception, d'une façon aléatoire, ensuite ils ont simulé les performances de chaque topologie trouvée, où ils ont pris la topologie qui maximise le plus les performances d'une application donnée sur le système. Dans l'exemple **[95]**.

D'autres travaux tels que **[22]**, ont divisés l'application en clusters, en affectant à chaque cluster une topologie standard qui maximise ses performances. Puis, l'ensemble de ces topologies est fusionné afin de former une topologie personnalisée finale.

Autres chercheurs ont démarré d'une topologie standard 2D Mesh. Ils ont appliqué le mapping et des simulations du trafic du réseau, et ils ont remarqué des fois, qu'il existe des ressources non utilisées qui occupent la surface et consomment énergie, et d'autres fois une surcharge dans le réseau. D'où ils ont opté soit à la suppression ou à l'ajout des liens afin d'améliorer encore les performances globales du système.

D'autres travaux ont étudié l'impact des architectures partiellement connectées : certains ont fixés des architectures partiellement connectées (jusqu'à 50% et 25% du nombre de TSV). D'autre ont étudié l'optimisation du nombre et placement des TSV afin de minimiser la surface et le coût de communication.

Nous proposons dans notre carde de projet de fin d'étude, l'étude de l'hybridation (proposer une solution algorithmique) de la phase de mapping et la personnalisation de la topologie en considérant la technologie 3D.

L'idée de l'hybridation vise considère plusieurs critères à d'optimiser pour plusieurs objectifs en même temps : Le coût de communication (délai/énergie) pendant le placement des composants (mapping). Ainsi que, la surface du réseau en optimisant le nombre de liens verticaux. Et ceci permettra au final de spécifier la topologie 3D personnalisée partiellement connectée.

8. Conclusion

Ce chapitre donne essentiellement les grandes lignes concernant la phase de mapping dans les réseaux sur puce. Le processus du mapping change selon le type de l'application et sa complexité. Il peut être statique ou dynamique, comme il peut être fait en parallèle avec une autres phases (ordonnancement, routage, floorplanning).

L'avènement de la technologie 3D (TSV) et l'utilisation du nouveau paradigme réseau sur puce a pu donner une architecture 3D plus performant en terme de communication. Cependant, l'utilisation de ces liens verticaux doit être contrôler afin maintenir l'équilibre entre les performances et le coût en surface dans le système.

Des travaux de littérature en permet de voir l'efficacité des topologie 3D (principalement réseau sur puce) partiellement connectée, que ce soit en terme de communication ou en terme de surface.

La synthèse présenté dans ce chapitre nous a permet de concrétiser notre idée de base et de proposer une nouvelle solution (algorithmique, application) qui sera détaillée dans le chapitre suivant.

Chapitre 4 : Implémentation et Tests

1. Introduction

Le développement des technologies a permis l'intégration de plusieurs composants sur une même puce, ces composants communiquent entre eux à travers des interconnexions. Le nouveau paradigme de communication (réseau sur puce) permet un gain considérable en performance. Cependant, plusieurs critères et paramètres influences sur ces performances, nous citons le coût de communication, le coût en surface de silicium et la topologie réseau.

Notre idée consiste à hybrider la phase de mapping avec la personnalisation de la topologie en utilisant la technologie 3D. Cette problématique est considérée comme un problème d'optimisation multi objectifs, de ce fait, nous allons utiliser des métaheuristiques basés sur l'algorithme génétique avec l'approche Pareto afin de trouver des solutions de compromis (non dominées) qui optimisent toutes nos fonctions objectives.

La formulation mathématique de notre problématique ainsi que le détail de notre solution sont expliqués dans les paragraphes suivants.

2. Formulation du problème

Hybrider la phase de mapping avec la personnalisation de la topologie en considérant la technologie 3D est classé comme un problème d'optimisation combinatoire multi objectifs vu le nombre des critères qu'on cherche à optimiser.

2.1. Formule mathématique

$$\begin{cases} \mathbf{MpSoli} \in \mathbf{SMP} = \{\mathbf{MpSol1}, \mathbf{MpSol2}, \dots, \mathbf{MpSoln}\} \textbf{ tels que } \mathbf{F(x)} = \textbf{ minCost est optimisé} \\ \mathbf{NbrTSV}_{\boldsymbol{min}} \in [\mathbf{1}, \mathbf{MaxTSV}] \textbf{ tels que } \mathbf{G(x)} = \textbf{minSurface est optimisé} \end{cases}$$

SMP représente l'ensemble de solutions de mapping généré par l'application « NovaCronoss » (initialiser d'une manière aléatoire).

NbrTsv représente le nombre des liens verticaux dans la topologie du réseau (générer pour chaque solution d'une façon aléatoire entre **[1, MaxTSV]**).

MaxTSV est calculé selon le nombre de IPs total de l'application où :

$$maxTSV = \frac{TotalPIs}{2}$$

2.2. Supposition

Notant que les critères physiques de positionnement de tuiles, liens, l'espace à respecter entre les couches empilés, taille des routeurs, commutateurs, ne sont pas considérés.

Nous avons considéré un réseau 3D constitué de 2 plans seulement (niveaux) reliés par des liens verticaux. Le nombre de plans est limités en raison des problèmes thermiques engendrés par l'empilement de ces plans **[110]**.

Chaque plan contient un nombre de tuiles (tel qu'une tuile est un placement candidat d'un lien vertical) qui est égal à la moitié de nombre de IP totale de l'application. Ces tuiles sont identifiées par des cordonnées géométriques (x, y), ainsi que des TAG = {#1, #2, …, #n} pour pouvoir faire la différence entre les tuiles des différents plans. La **figure 34** représente un exemple illustratif. Exemple d'une application à 12 IPs est utilisé avec mapping et affectation aléatoire de liens (NbrTSV =3 compris entre [1-6]).

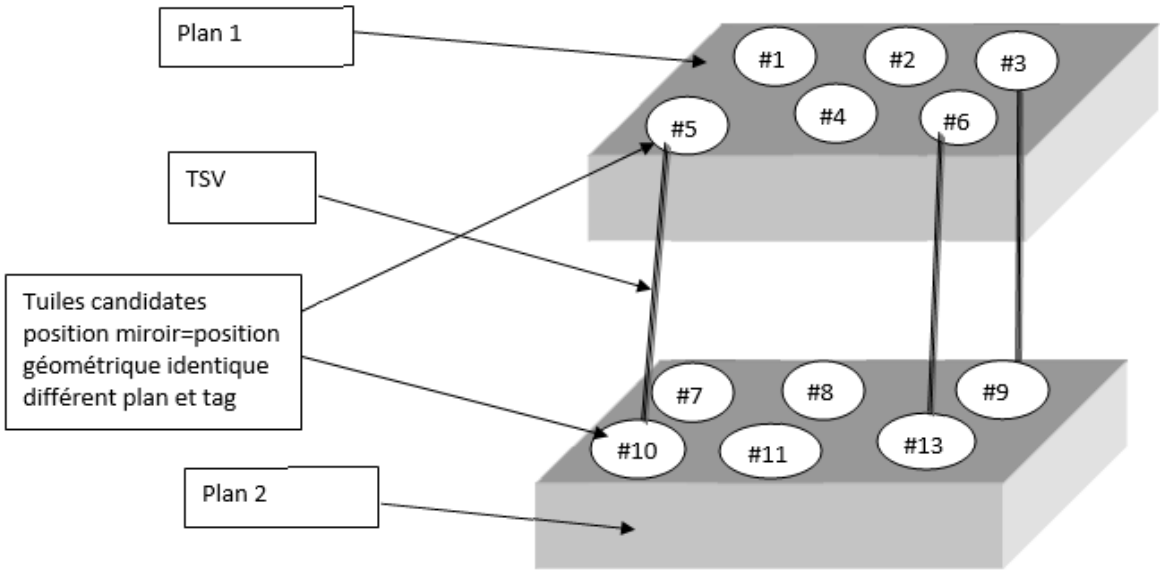

Figure 34: Représentation explicatif de notre supposition

2.3. Graphes d'application et d'architecture

Les deux graphes sont décrits comme suit (**figure 35**) :

- **Un graphe d'application (GAP)** qui est un graphe dirigé G (C, A). Chaque sommet $c_i \in C$ désigne un IP. Le lien dirigé $a_{ij} \in A$ désigne la trace de communication du IP c_i vers le IP c_j. Le poids de chaque transfert w_{ij} indique le volume de communication a_{ij}.

- **Un graphe d'architecture (GAR)** qui est un graphe orienté G (T, L). Chaque sommet $t_i \in T$ représente une tuile de l'architecture. L'arc dirigé $l_{ij} \in L$ représente le lien physique entre la tuile t_i et la tuile t_j.

Le problème de mapping (placements des composants) est modélisé par 2 graphes, la formulation de ce problème étant une fonction **Map: GAP GAR** qui minimise la fonction de coût de communication.

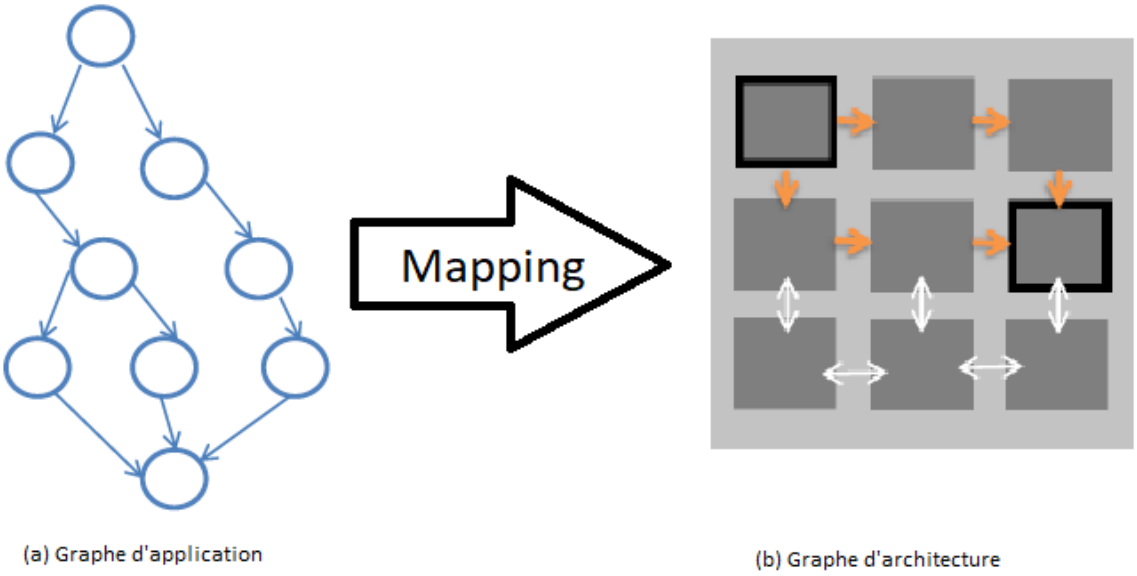

Figure 35: Exemple de mapping dans 1 seul plan

Nous avons considéré une topologie 3D mesh (partiellement connecté verticalement), où les tuiles sont connectées par des liens horizontaux (sur le même plans) et par des TSV (entre différents plans), le nombre de TSV est généré aléatoirement (pour chaque solution proposée par notre application).

Le réseau sur puce peut être décrit comme un réseau de longueur N et de largeur M, où la distance entre tuile t1 (x1, y1) et une autre tuile t2 (x2, y2) dans le même plan est calculé en utilisant la distance Manhattan, vu que l'acheminement des paquets entre une source et une destination peut prendre plusieurs chemins possibles :

$$\mathbf{dManhattan}\ (t_i, t_j) = \mathbf{d(t1, t2) = |x2 - x1| + |y2 - y1|}$$

Afin de résoudre cette problématique nous avons proposé deux métaheuristiques basées sur l'algorithme génétique multi objectif (l'approche Pareto) afin de trouver les solutions de compromis qui optimisent toutes nos objectifs.

2.4. Les fonctions objectives

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

Coût global = min [Coût communication, Coût surface]

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

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

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

$$Cost_{comm} = \sum^{T} W_{ij} * NbrSaut_{ti,tj}$$

Avec $\mathbf{W_{ij}}$ égale le volume de transfert entre IP_i et IP_j qui sont mappés aux tuiles respectivement t_i (x1, y1) et t_j (x2, y2).

Le $\mathbf{NbrSaut_{ti,tj}}$ qui peut être calculé sur deux façon :

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

$$\mathbf{NbrSaut_{ti,tj} = dManhattan\ (t_i, t_j) = |x2 - x1| + |y2 - y1|}$$

- Quand la communication se fait entre deux tuiles qui se trouvent dans différents plans (**voir figure 31 p52**) la formule mathématique est :

$$\mathbf{NbrSaut_{ti,tj} = dManhattan\ (t_i, t_j) + Coef}$$

Avec : **Coef** est le coefficient rajouté (**Coef = 0.8**) pour indiquer l'utilisation d'un lien de communication vertical (TSV).

Minimiser la surface revient dans notre objectif à minimiser le nombre de liens verticaux (à cause de la surface occupée par les TSVs).

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

G(x) = min {Surf : surface de silicium utilisé =optimisé nombre des liens vertical}

3. Résolution du problème

3.1. Présentation des algorithmes adapté (SGANova, SPEA-II Adapté)

Nous avons proposé deux algorithmes nommés respectivement : SGANova et SPEA-II Adapté.

#Algorithme SGANova

Variable :

```
//Taille_archive sera déterminé automatiquement
Taille_pop, Taille_Archive, MaxItération: entier;
Archive_Nova: Archive;
popI, popE, Front_p: list<population> ;
Convergence : booléen ;
Debut
Taille_pop      saisie_utilisateur();
//générer une population initial {mapping + affectation TSV}
popI     generate_population_initial() ;
i    0 ;
//calcul de fitness
Evaluate_pop_initial();
//initialiser l'archive qui va contenir que les meilleures solutions rencontrées lors de la recherche
```

Archive_Nova Front_pareto (popI);

Tant que (!convergence ou i < MaxItération)
//générer population enfant en sélectionnant de la population précédente et en appliquant les opérateurs de croissement & mutation
popE generate_population_Enf(popE) ;
//calcul de fitness
Evaluate_pop_Enf() ;
//mise à jour l'archive avec meilleurs solution de la population enfant front multi-objectif
updateArchive (Front_pareto (popE));
//remplacer population initial par population Enfant générée
Replacement (popI, popE) ;
//si la différence en terme de coût communication moyenne et surface moyenne est petit, alors (converge = vrai)
Convergence convergence(popI, popE) ;
//si sa converge, pas la peine de continuer à itérer
si (convergence = true) alors exit() ;
//si sa converge pas, continuer à itérer
sinon Continue (); i++;
Fin Tant que
//donner l'archive qui contient les solutions de compromis a l'utilisateur
Return Archive_Nova ;
Fin

#Algorithme SPEA-II Adapté

Variable :
//Taille_archive sera déterminé automatiquement
Taille_pop, Taille_Archive, MaxItération: entier ;
Archive_ SPEA-II Adapté: Archive ;
popI, popE, Front_p: list<population> ;
Debut
Taille_pop saisie_utilisateur();
popI generate_population_initial() ;
i 0 ;
//générer une population initial {mapping + affectation TSV}
Evaluate_pop_initial();
//calcul le fitness
Archive_ SPEA-II Adapté Front_pareto (popI);
//initialiser l'archive qui va contenir que les meilleures solutions rencontré lors de la recherche
Tant que (i < MaxItération)
PopE generate_population_Enf (popE, Archive_spea) ;
//générer population enfant en sélectionnant de la popI et l'archive et en appliquant les opérateurs de croissement & mutation sur popI
Evaluate_pop_Enf() ;
//calculer fitness
updateArchive(Front_pareto(popE));
//mise à jour l'archive avec meilleurs solution de la population Enfant
//remplacer population initial par population Enfant générée
Replacement (popI,popE) ;

//si la différence en terme de coût communication moyenne et surface moyenne est petit, alors (converge = vrai)
i++ ; **Fin Tant que**
//tester si la taille de l'archive dépasse la taille de la population ou non
Si (Taille(Archive_ SPEA-II Adapté) > Taille pop)
//méthode tronction permet de supprimer les solutions moins bonnes
Tronction(Archive_ SPEA-II Adapté) ;
Sinon
//remplir l'archive avec les solutions dominées de la popE (dernière itération) jusqu'à que la taille de l'archive est égal à la taille de population
remplir(Archive_ SPEA-II Adapté) ;
//donner les solutions de compromis inclues dans l'archive a l'utilisateur
Return Archive_ SPEA-II Adapté
Fin

Le tableau qui suit **(tableau 5)** présente les différents points entre les deux algorithmes SGNova et SPEA-II Adapté.

Tableau 5: Comparaison entre les 2 algorithmes utilisés

SGANova	SPEA-II Adapté
Taille archive dynamique (taille Front Pareto trouvé)	Taille archive statique (taille population initiale)
Sélection parent depuis population	Sélection parent depuis archive + population
Arrêt sur nombre itération max ou convergence (Temps d'exécution réduit)	Arrêt sur nombre itération max uniquement (Temps d'exécution plus)
Archive contient les meilleurs solution rencontrés (solution de compromis)	Archive contient les meilleurs solution rencontrés ainsi des solution dominés

Cette comparaison a pour but de montrer les avantages et les inconvénients de chaque solution proposée. Dans le cas de SPEA-II Adapté, la reproduction des individus se fait de l'archive et de la population, la taille de l'archive telle qu'elle est (statique), alors que la taille de l'archive de SGANova est dynamique ou à chaque itération l'archive est mise à jour.

Pour la condition d'arrêt dans le cas de SGANova, elle base sur soit le nombre d'itérations ou la convergence qui permet de sortir de la boucle si nous somme entrain de générer la même population à chaque fois (Aspect d'intelligence).

Pour le cas de SPEA-II Adapté, il a comme critère d'arrêt que le nombre d'itération uniquement, un temps de recherche supplémentaire a été constaté pendant les tests, vu que SPEA-II Adapté continue à itérer même s'il est entrain de générer la même population.

Des tests et des résultats vont être présenter et discuter dans ce chapitre par la suite.

3.2. Représentation des paramètres de la solution

3.2.1. Génération de la population initiale

Elle consiste à créer un nombre de solutions de départ qui soient réalisables. Ce nombre est un paramètre choisi par l'utilisateur. L'ensemble des solutions de départ est appelé population initiale.

Une solution est dite réalisable si et seulement si pour chaque tuile on lui affecte un seul IP. Notant que la population initiale représente un ensemble de mapping solution généré d'une façon pseudo-aléatoire.

Les algorithmes génétiques appliquent des opérateurs de croissement et de mutation afin de reproduire des meilleures solutions. Ces opérateurs permettent de créer une nouvelle population appeler population enfant.

3.2.2. Représentation d'une solution de Mapping

Une solution dans le problème de mapping peut être formulée comme un vecteur d'entier où chaque case représente une IP et son contenu représente le numéro de la tuile qui va lui être assignée comme le démontre la **figure 36.**

IPs	1	2	3	4	5	6
Tuiles	**#1**	**#2**	**#5**	**#1**	**#4**	**#6**

Figure 36: Présentation d'une solution mapping

3.2.3. Affectation de TSV {nombre, placement} :

Etant donné un mapping, supposant que l'application est constituée de 6 IPs, donc notre NoC sera constitué de 3 tuiles de chaque plan, l'affectation de TSV est faite comme suit :

- Etant donné qu'on dispose de la matrice des Transfer entre les IPs de l'application.
- Etant donné que le $Mapping_1$ appartient à l'ensemble de mapping SMP = {$mpsol_1$, …, $mpsol_n$}
- NbrTSV = compris [1, 3] ou 3 représente le MaxTSV possible dans cet exemple.
- **Figure 37** représente un exemple d'affectation de TSV pour une solution de $Mapping_1$

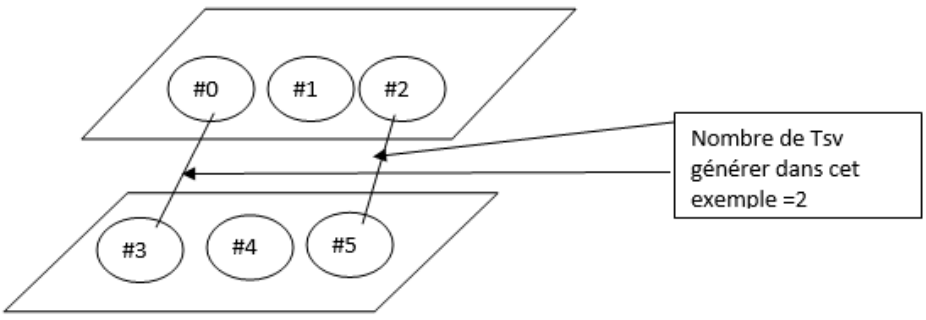

Figure 37: Représentation graphique d'affectation de TSV

Mapping $_1$:

IP	**0**	**1**	**2**	**3**	**4**	**5**
Tuiles	#0	#5	#1	#2	#3	#4

Matrice de Transfer de l'application :

Transfer	**(0,1)**	**(1,2)**	**(3,5)**	**(4,6)**	**(3,1)**
Volume	100	200	30	14	50

Détermination des Tuiles candidat pour placement de TSV :

Transfer	**(0,1)**	**(1,2)**	**(3,5)**	**(4,0)**	**(3,1)**
Mapping	(#0,#5)	(#5,#3)	(#2,#4)	(#3,#0)	(#2,#5)
Tuiles cand	0	0	0	1	1

Placement d'un NbrTSV [1, MaxTsv] d'une façon aléatoire sur les tuiles candidates :

IP	**0**	**1**	**2**	**3**	**4**	**5**
Tuiles	#0	#5	#1	#2	#3	#4
Tsv	1	1	0	0	0	0

L'affectation de TSV se fait d'une manière aléatoire et uniquement pour les tuiles (candidates) qui sont dans des positions identique et qui se trouvent dans différent plans, et un TAG différent (l'exemple de (#0, #3) sur la **figure 37**).

3.2.4. Opérateur de croissement

Nous avons proposé plusieurs types de croisement. L'utilisateur de l'application peut choisir pour une exécution parmi les types suivants :

3.2.4.1. Croissement à un point

Deux parents (chromosomes) sont choisis d'une façon aléatoire. Le principe consiste à réaliser un croisement entre deux parent en 1 seul point fixé à la moitié de la longueur de mapping et affectation TSV de chacun des deux parents. La **figure 38** illustre ce type de croissement.

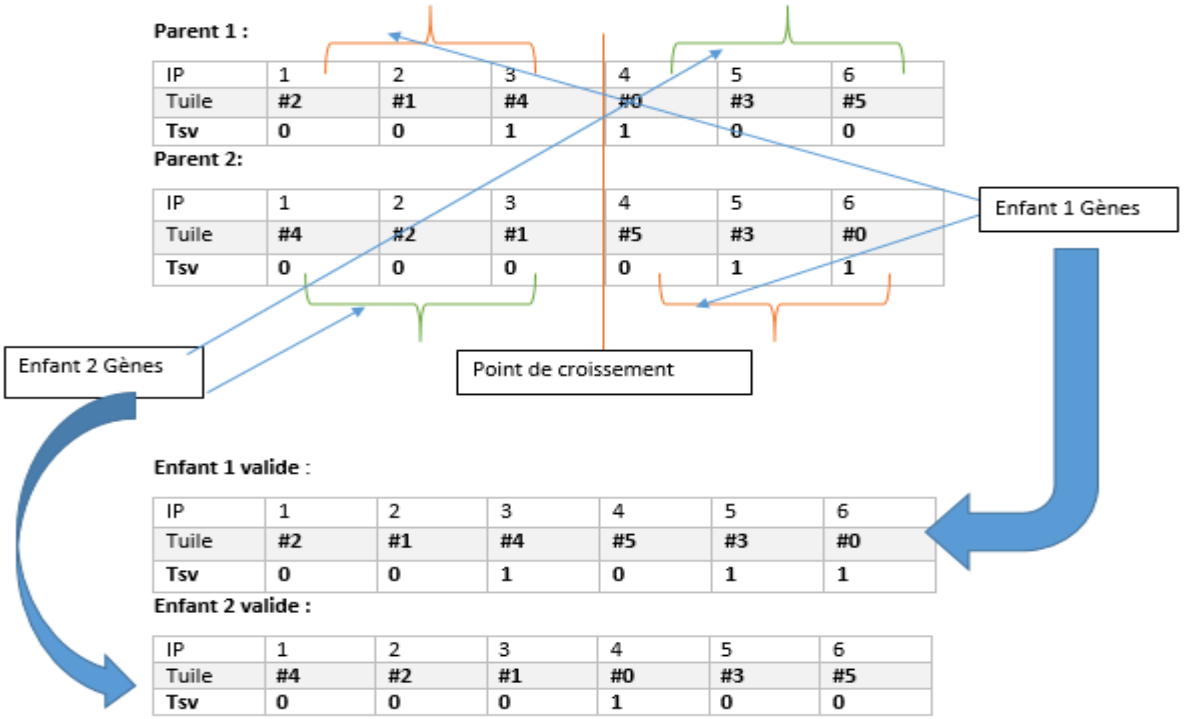

Figure 38: Croissement à un point

3.2.4.2. Croissement à M points (notre cas 2 points)

Le croissement à m points (**figure 39**) consiste de générer deux points aléatoires sur la structure gène des parents. L'échange des parties gènes (aléatoire) fait en sorte de généré dans notre cas d'étude deux nouveaux enfants.

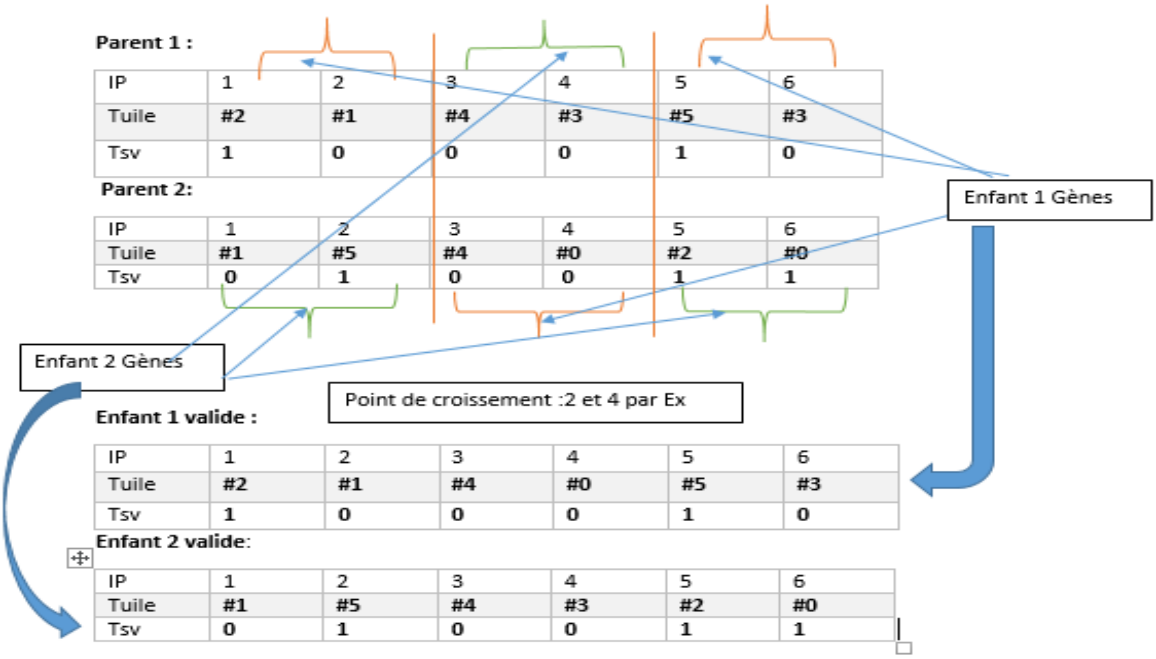

Figure 39: Croissement à m points

3.2.5. Opérateur de mutation

Cet opérateur est utilisé afin d'assurer la diversification des solutions et afin qu'on ne reste pas bloquer dans le local optimum (une même solution qui se répète). Il est réalisé par le fait de substituer un gène parmi les gènes d'une solution choisie, aléatoirement **(figure 40)**.

Chromosome :

IP	1	2	3	4	5	6
Tuile	#2	#0	#1	#5	#3	#4
Tsv	0	0	0	1	0	0

Chromosome Muté non valide:

IP	1	2	3	4	5	6
Tuile	#2	#5	#1	#5	#3	#4
Tsv	0	1	0	1	0	0

Chromosome valide après réparation:

IP	1	2	3	4	5	6
Tuile	#3	#5	#2	#0	#4	#5
Tsv	0	1	0	0	0	0

Figure 40: Exemple de mutation

3.2.6. Chromosome non valide

Des enfants non valides (**figure 41**) peuvent être générés (affectation de plusieurs IPs à une seule tuile) dans les étapes de croisement ou de mutation.

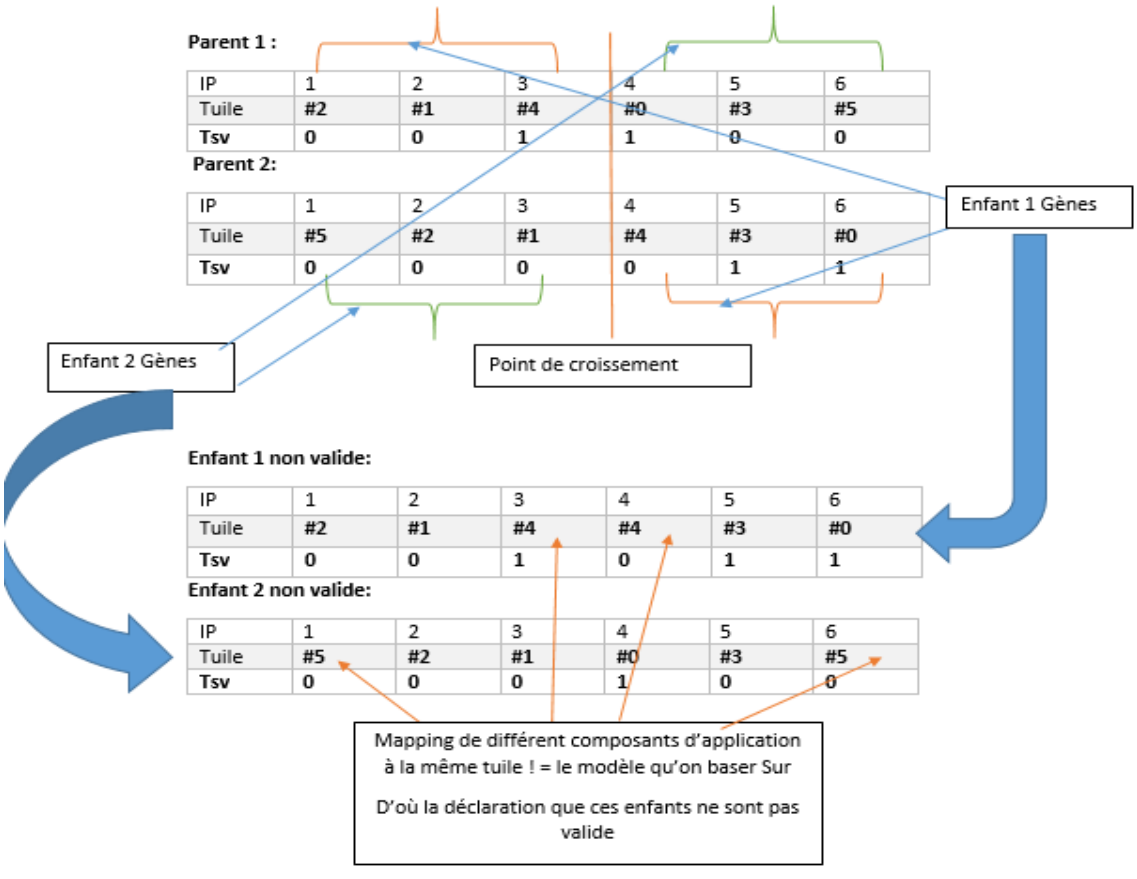

Figure 41: s générés après un croisement

3.2.7. Réparation d'un chromosome :

Afin de régler le problème illustré sur (**figure 41**), on a opté à l'implémentation d'une méthode qui permet de réparer l'enfant non valide et une autre qui permet de le refaire un teste si l'enfant réparé est toujours réalisable ou non, si oui on le garde sinon on le supprime. Le processus est expliqué par la suite **(figure42).**

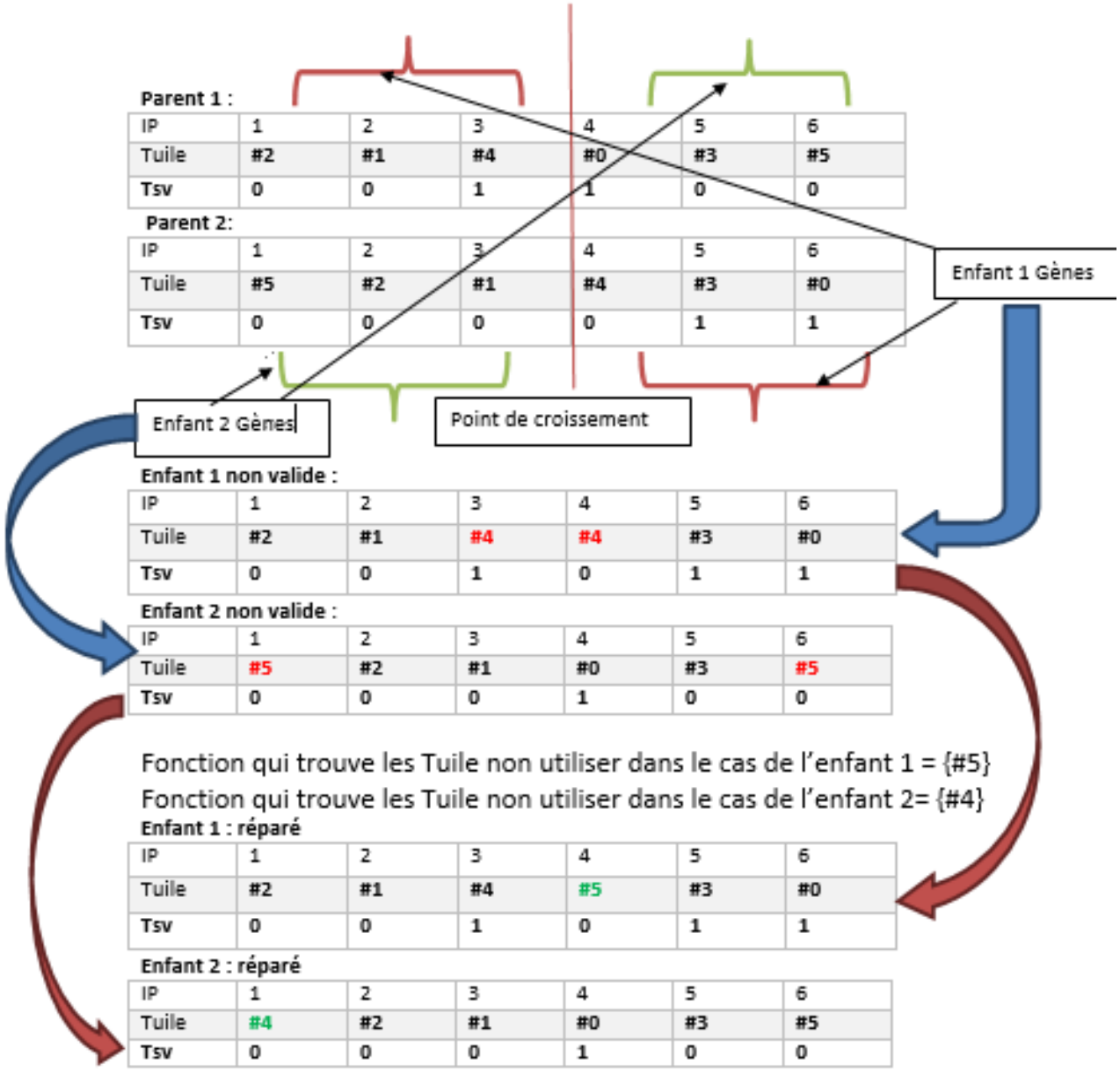

Figure 42: Réparation d'un chromosome non valide

3.2.7. Fonction fitness

A chaque génération de nouvelle population (solutions), les fonctions objectives de chaque solution sont calculées afin de mesurer leurs pertinences (fitness).

3.2.8. Sélection :

SGANova :

Nous avons choisi une sélectionne aléatoire afin de donner à toute les solutions (bonnes ou mauvaises) les même chance (la sélection est faite à partir de la population précédente).

SPEA-II Adapté :

Nous avons utilisé le même type de sélection aléatoire. La seule différence entre cette technique et la précédente est que l'archive participe dans la génération de la nouvelle population. La sélection est faite à partir la population précédente et l'archive.

3.2.9. Remplacement :

La population précédente est remplacée par la nouvelle population (enfants) pour les deux techniques utilisés.

3.3.1. Mise à jour de l'archive :

SGANova :

L'archive est mise à jour itérativement, dans le cas de SGANova la taille de l'archive est dynamique

SPEA-II Adapté :

L'archive est mise à jour itérativement par les solutions non dominées appelées (Front Pareto) et qui représente des solutions de compromis qui optimisent nos objectifs. La taille de l'archive de SPEA-II Adapté est fixée à la taille de la population. En cas de dépassement de la taille, une méthode de tronction est invoquée pour recadrer la taille.

3.3.2. Convergence :

La convergence représente l'écart entre le coût moyenne de communication et le nombre TSV moyenne de la population parent et la population enfant. Si cet écart est égal à une valeur proche d'epsilon, on dit que la recherche converge, sinon on continue à itérer.

Après avoir présenté les différentes phases des 2 méta-heuristiques, nous allons passer aux tests. La partie test est décomposée en deux parties. Dans la première partie, une étude paramétrique est effectuée afin de choisir les paramètres d'entrées de l'application (les probabilités de croisement et mutation, la taille de la population, etc). Dans la deuxième partie, une comparaison entre les deux techniques est effectuée par le biais des paramètres trouvé dans l'étape précédentes.

4. Tests et résultats

Après avoir implémenter les différents algorithmes détaillés dans ce chapitre, nous présentons les différents résultats obtenus lors de l'expérimentation.

Nous commençons par une présentation des différents benchmarks sur lesquelles nous avons effectué nos tests, puis nous allons étudier les résultats obtenus après avoir fait une série de tests sur ces benchmarks.

Notre application intitulé « NovaCronoss » permet de générer un ensemble de solutions de mapping, ainsi que l'optimisation de nombre des liens verticaux. Du fait que ces deux objectives sont antagonistes nous avons opter pour l'optimisation multi objective afin d'obtenir des solutions de compromis.

Les tests sont réalisés sur une machine équipé d'un processeur Intel Core i3 ivy bridge muni de deux cœurs à 1.8 GHz et d'une RAM de 6 GO sous le système d'exploitation Windows 8.1 de 64 bits.

4.1 Présentation des Benchmarks

Dans notre expérimentation on a utilisé 3 benchmarks qui sont : VOPD, MPEG et le 3ème l'aléatoire (Random).

4.1.1 Benchmark VOPD (Video Object Plane Decoder)

L'application VOPD est composée de 16 composants (IPs) qui communiquent entre eux à travers 21 liens. (**Figure 43)**

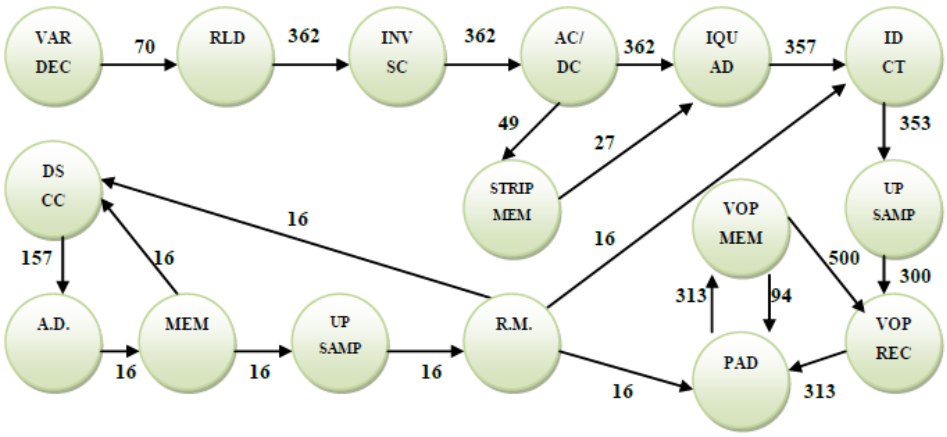

Figure 43: Benchmark VOPD [111]

4.1.2. Benchmark MPEG

Le benchmark MPEG possède 12 nœuds et 13 liens, il représente un bon compromise entre le coût total de communication, la figure suivant montre le graphe d'application de ce benchmark **(figure 44)**

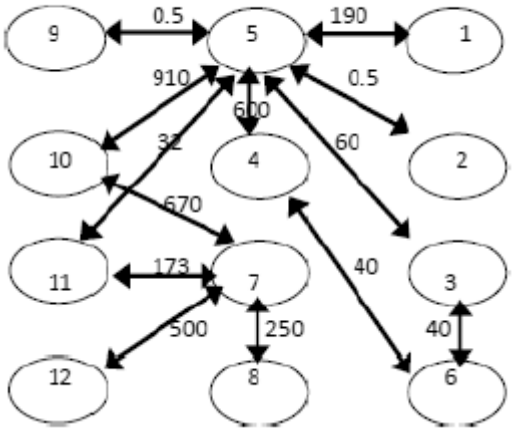

Figure 44: Benchmark MPEG [40]

4.1.3. Random:

L'utilisateur dans ce cas doit préciser le nombre de IPs (entre 20-80 IPs) en remplissant le champ "nombre de IPs" dans l'application. Ensuite, selon le nombre d'IPs entrés, un placement aléatoire est réalisé sur une architecture 3D (02 plans).

La bande passante de communication entre les différents nœuds est affecter d'une façon aléatoire entre 1-600 bits, ainsi que les différents transferts entre les IPs de l'application.

4.2. Etude de paramétrage et résultats des tests :

Pour tester les méthodes implémentées nous considérons le coût de communication et le coût en surface comme fonctions objectives, les valeurs liées aux unités à savoir le coût de communication et le surface sont ignorées, ainsi les contraintes physiques de fabrications.

4.2.1 Effet de variation sur la probabilité de croissement et mutation :

Etant donné que la probabilité de croissement appartient à un intervalle [0.1-0.9] et la probabilité de mutation entre [0-1]. Le but de cette étude est de tester l'effet de variation sur le taux de mutation et de croissement ainsi que le nombre d'itération et la taille de la population sur les différentes techniques qu'on a proposées {SGANova et SPEA-II Adapté}.

Comme les probabilités sont dans un intervalle, il n'est pas possible de tester tous les combinaisons, de ce fait, nous avons choisi 25 combinaisons parmi les combinaisons possibles. Pour effectuer nos tests une série de 5 exécutions est faites (noté que chaque exécution représente une nouvelle population générée), la taille de population est égale à 100 ainsi le nombre d'itération.

-Nombre de TSV moyenne :

$$\sum \frac{\text{TSV des solutions de l'archive}}{\text{taille archive}}$$

-Coût Moyenne :

$$\sum \frac{\text{coût des solutions de l'archive}}{\text{taille archive}}$$

-Nombre de solution optimal moyen trouvé :

$$\frac{\text{nombre des solutions de l'archive}}{\text{nombre d'executions}}$$

Etant donné que le Benchmark Random n'est pas stable (à chaque exécution, le volume de communication est affecté aléatoirement à l'ensemble des IPs). Une seule exécution est présentée dans le tableau suivant.

Random(20IP) :

Tableau 6: Tests fait pour fixer Pc et Pm pour Benchmark Random en utilisant les 2 techniques

Technique utilisé	Probabilité de croissement	Probabilité de mutation	Nombre de solution optimal trouvé 1 Seul Exécution	Nombre de Tsv Moyen	Coût Moyen
SGANOVA	0.9	0.35	3	1	8341,33
	0.8	0.1	4	1	7923,5
	0.7	0.2	1	1	11743
	0.6	0.3	3	1	8183,67
	0.5	0.4	1	1	7800
	0.4	0.5	5	1	8321,5
	0.3	0.6	1	1	10974
	0.2	0.7	21	1	8479
	0.1	0.8	5	1	8754,33
	0.5	0.26	4	1	7948,5
	0.15	0.35	20	1	7977,5
	0.35	0.1	17	1	8538
	0.51	0.2	1	1	14229
	0.71	0.3	9	1	9111,78
	0.91	0.4	3	1	7959,33
	0.43	0.55	50	1	10789,8
	0.32	0.63	4	1	10425
	0.21	0.7	5	1	5516
	0.18	0.87	1	1	10382
	0.5	0.1	12	1	13473,5
	0.32	0.1	3	1	10500,7

	0.28	0.7	1	1	11111
	0.1	0.8	17	1	9683,6
	0.74	0.26	2	1	10610
	0.55	0.98	8	1	8006
SPEA-II Adapté	0.1	0.8	10	1	8492,78
	0.9	0.35	1	1	8356,64
	0.8	0.1	3	1	12751,2
	0.7	0.2	5	1	8141,56
	0.6	0.3	4	1	8411,6
	0.5	0.4	3	2	8382,14
	0.4	0.5	7	1	11057,3
	0.3	0.6	6	1	8841
	0.2	0.7	3	1	8889,25
	0.5	0.26	15	1	8211,65
	0.15	0.35	1	1	8821,5
	0.35	0.1	2	1	12159,6
	0.51	0.2	8	1	9280,29
	0.71	0.3	9	1	9236,55
	0.91	0.4	4	1	11299,6
	0.43	0.55	3	1	11767,8
	0.32	0.63	5	1	7335,5
	0.21	0.7	7	1	10249,9
	0.18	0.87	2	1	13531,6
	0.5	0.1	17	1	10682,2
	0.32	0.1	10	1	11391,5
	0.28	0.7	8	1	10206,1
	0.1	0.8	9	1	11231
	0.74	0.26	1	1	8042,89
	0.55	0.98	2	1	8923,12

Discussion résultat Random

Pour chaque benchmark généré par l'option Random il existe une probabilité de croissement et de mutation qui permet de trouver une ou plusieurs solutions qui ont un coût de communication et de surface moyennement réduite, dans notre cas un exemple d'un benchmark Random à 20 IPs a été utilisé.

VOPD :

Le **tableau 3** représente les résultats obtenues, moyenne des solutions trouvées, de 05 exécutions avec une population de 100 solutions et un nombre d'itération égal à 100, en changeant le taux de croissement et de mutation.

Tableau 7: Tests fait pour fixer Pc & Pm pour VOPD en utilisant les 2 techniques

Technique utilisé	probabilité de croissement	Probabilité de mutation	Nombre de solution optimal Moyenne trouvé après 5 Exécutions	Nombre de Tsv Moyen	Coût Moyen
SGANOVA	0.9	0.35	3	2	4666
	0.8	0.1	3	2	3897
	0.7	0.2	3	2	4182.06
	0.6	0.3	3	2	4532
	0.5	0.4	3	2	4251.5
	0.4	0.5	2	2	4707
	0.3	0.6	2	1	4556.06
	0.2	0.7	2	2	4571.4
	0.1	0.8	2	1	5171
	0.5	0.26	3	2	4517.6
	0.15	0.35	3	2	4770.3
	0.35	0.1	1	2	4670
	0.51	0.2	2	2	4545
	0.71	0.3	4	2	3619
	0.91	0.4	3	2	3891
	0.43	0.55	3	2	5136
	0.32	0.63	1	2	4340
	0.21	0.7	2	2	4748.4
	0.18	0.87	1	3	4244
	0.5	0.1	2	2	4167
	0.32	0.1	2	2	4882.4
	0.28	0.7	3	2	5143
	0.1	0.8	1	2	4768.3
	0.74	0.26	3	2	4098.2
	0.55	0.98	1	2	4410
SPEA-II Adapté	0.9	0.35	3	2	3699.8
	0.8	0.1	4	2	3910.1
	0.7	0.2	4	2	3740
	0.6	0.3	4	2	3929
	0.5	0.4	3	2	3579
	0.4	0.5	2	1	4758
	0.3	0.6	1	1	3877.4
	0.2	0.7	1	2	4324.9
	0.1	0.8	1	2	4091.5
	0.5	0.26	4	2	3978

	0.15	0.35	1	2	4188.06
	0.35	0.1	2	2	4805.6
	0.51	0.2	3	2	3732.2
	0.71	0.3	4	2	3492
	0.91	0.4	2	2	4782.3
	0.43	0.55	2	1	4412
	0.32	0.63	1	1	4360.9
	0.21	0.7	1	2	4250.21
	0.18	0.87	1	2	4338.6
	0.5	0.1	4	2	3893.37
	0.32	0.1	3	2	4982.3
	0.28	0.7	1	1	4511
	0.1	0.8	1	1	4634
	0.74	0.26	1	1	4534.9
	0.55	0.98	4	2	4160

Discussions des résultats VOPD :

Technique SGANova :
On remarque que les meilleurs résultats, en terme coût moyenne réduit, ainsi le nombre des interconnexions verticaux utilisées sont obtenu lorsque la probabilité de croisement est comprise entre [0.9-0.5] et la probabilité de mutation est comprise entre [0.1-0.4], des résultats moins bons sont constatés lorsque la probabilité de croissement est comprise entre [0.4-0.1] et pareil quand la probabilité de mutation est comprise entre [0.4-0.9].

Technique SPEA-II Adapté :
On remarque que les meilleurs résultats, en terme coût moyenne réduit, ainsi le nombre des interconnexions verticaux utilisé, sont obtenu lorsque la probabilité de croisement est comprise entre [0.9-0.5] probabilité de mutation est comprise entre [0.1-0.4], des résultats moins bons sont constatés lorsque la probabilité de croissement est comprise entre [0.4-0.1] et que la probabilité de mutation est comprise entre [0.4-0.9].

Le SPEA-II Adapté permet de trouver des résultats moyennement plus intéressant que le SGANova (des résultats avec un coût moyenne plus réduit), ceci est peut-être due au fait que dans cette algorithme l'archive participe à la génération de la nouvelle population nommé population enfant, ce qui peut augmenter les chances de croiser des solutions intéressant en terme coût et de TSV.

MPEG : total paquets échanger pour ce benchmark égal à 7397.

Tableau 8: Tests effectue sur le Benchmark MPEG pour fixer Pc et Pm en utilisant les 2 techniques

Technique utilisé	probabilité de croissement	Probabilité de mutation	Nombre de solution optimal moyenne trouvé	Nombre de Tsv Moyen	Coût Moyen
SGANova	0.9	0.35	3	1	7796
	0.8	0.1	3	2	6776
	0.7	0.2	3	2	5812
	0.6	0.3	2	3	5802
	0.5	0.4	2	3	6124
	0.4	0.5	4	2	6100
	0.3	0.6	3	3	5862
	0.2	0.7	2	1	6802,5
	0.1	0.8	2	2	6265
	0.5	0.26	3	2	6708
	0.15	0.35	3	2	8016
	0.35	0.1	3	3	5636
	0.51	0.2	2	2	6942,67
	0.71	0.3	3	4	5750
	0.91	0.4	3	2	8539,6
	0.43	0.55	3	3	6700
	0.32	0.63	2	1	7059
	0.21	0.7	4	2	7022
	0.18	0.87	2	3	6398,5
	0.5	0.1	2	1	5749
	0.32	0.1	2	4	7518
	0.28	0.7	3	4	6296
	0.1	0.8	2	2	6250
	0.74	0.26	4	2	7232
	0.55	0.98	3	2	7796
SPEA-II Adapté	0.9	0.35	3	2	5368
	0.8	0.1	2	2	5616
	0.7	0.2	3	2	5217,33
	0.6	0.3	6	3	6078,67
	0.5	0.4	1	3	5750
	0.4	0.5	1	2	6100
	0.3	0.6	1	3	5862
	0.2	0.7	1	2	5724
	0.1	0.8	4	2	5512
	0.5	0.26	1	2	6708
	0.15	0.35	3	2	6458,67
	0.35	0.1	6	3	6752
	0.51	0.2	3	2	5750,67
	0.71	0.3	3	3	5599,2
	0.91	0.4	5	2	4756
	0.43	0.55	1	3	6700

	0.32	0.63	1	2	5692
	0.21	0.7	2	2	7295
	0.18	0.87	6	3	6917,33
	0.5	0.1	1	1	6448
	0.32	0.1	1	4	6698
	0.28	0.7	1	4	6296
	0.1	0.8	3	2	5068
	0.74	0.26	3	2	5902,67
	0.55	0.98	3	2	5368

Discussion des résultats MPEG :

Technique SGANova :

On remarque que les meilleurs résultats, en terme coût moyenne réduit, ainsi le nombre des interconnexions verticaux utilisé, sont obtenu lorsque la probabilité de croisement, mutation sont compris entre [0.5-0.9] et [0.1-0.4] respectivement, des résultats moins bons sont constaté lorsque la probabilité de croissement est comprise entre [0.4-0.1] et que la probabilité de mutation est comprise entre [0.5-0.9]

Technique SPEA-II Adapté :

On remarque que les meilleurs résultats, en terme coût moyenne réduit, ainsi le nombre des interconnexions verticaux utilisé, sont obtenu lorsque la probabilité de croisement, mutation sont compris entre [0.5-0.9] et [0.1-0.4] respectivement, des résultats moins bons sont constaté lorsque la probabilité de croissement est comprise entre [0.4-0.1] et que la probabilité de mutation est comprise entre [0.5-0.9]

Les résultats obtenus par SPEA-II Adapté semble plus intéressant vu le coût de communication réduit par rapport SGANova. Si on compare les résultats obtenus sur les mêmes intervalles de probabilité, le SPEA-II Adapté semble prometteur, ceci peut être expliqué par le fait que sur cette algorithme fait participe l'archive à la génération de la nouvelle population, ce qui peut engendrer la découverte des solutions plus intéressants.

4.2.2 Effet de variation de la taille de population :

Tableau 9: Résultats de variation de nombre de population sur les différents benchmarks pour les 2 méthodes utilisées

Benchmark	Technique	Taille de population	Nombre de solution optimal moyenne	Nombre de TSV Moyenne	Coût de communication Moyenne	Nombre d'itération moyenne
VOPD	SGANova 0.7/0.3	10	2	1	5218.8	27
		20	2	2	4723.9	4
		50	3	2	4653.4	10
		70	2	1	4680	11
		100	4	2	4600.95	4
	SPEA-II Adapté 0.7/0.3	10	3	2	5422.9	100
		20	4	2	4629	
		50	3	2	400.65	
		70	5	2	4106.5	
		100	5	2	3876.45	
MPEG	SGANova 0.5/0.1	10	2	2	7390	20
		20	3	2	6659	9
		50	3	2	6034.26	25
		70	3	2	6539.96	6
		100	3	2	6265	4
	SPEA-II Adapté 0.9/0.4	10	4	2	7285.86	100
		20	3	2	6627.57	
		50	5	3	6484.5	
		70	5	3	5887.79	
		100	5	3	5549.92	
Benchmark	Technique	Taille de population	Nombre de solution optimal 1 seul Exécution	Nombre de Tsv Moyenne	Coût de communication Moyenne	Nombre d'itération moyenne
Random	SGANova 0.5/0.1	10	1	1	9229.0	1
		20	2	1	8837	3
		50	2	1	8754.5	1
		70	1	2	8405	10
		100	3	2	8889.66	1
	SPEA-II Adapté 0.5/0.1	10	2	1	8597.0	100
		20	2	1	8169.0	
		50	2	1	8122.0	
		70	4	2	8555.0	
		100	4	2	6724.0	

Discussion :

Pour la technique SGANova dans le benchmark VOPD on remarque que lorsque la taille de la population est égale à 50 le coût de communication moyenne est équivalent à 4653 alors une fois que la taille est augmenté à 100 le coût de communication moyenne est réduit à 4600.95. Cependant, si la taille est égale à 10 le coût de communication est égal à 5218.8, notant aussi que le nombre de TSV moyenne utilisé est le même dans les 2 cas ou la taille de population est égal à 50 ou 100.

Pour la technique de SPEA-II Adapté on remarque que le coût de communication moyen est grand lorsque la taille de population égal à 10. Cependant, une baisse sur le coût de communication a été constaté lorsque la taille de la population est équivalente à 50, et ceci indique que l'application a trouvé des solutions plus optimaux que l'ancien test. On a continué d'augmenter la taille de la population jusqu'à 100 échantillons, et nous avons remarqué qu'une baisse remarquable a été constaté en terme de coût de communication.

Pour le benchmark MPEG et que ce soit en utilisant la technique SGANova ou SPEA-II Adapté on a constaté que le coût de communication optimal est obtenu lorsque la taille de la population est égale 100.

La technique SGANova converge plus rapidement que la technique SPEA-II adapté vu que cette technique réalise à chaque itération un test afin de déterminer si les populations (précédente et nouvelle ou parent et enfant) ont un écart proche d'epsilon ou non. La convergence est illustrée par le nombre d'itération moyen (tableau précédent), où on constate que à chaque fois le SGANova à un nombre d'itération inferieur à celle de SPEA-II adapté

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

Dans notre série test la taille de population idéal est bien égal à 100, du fait que sa donne plus de solutions optimales trouvées et que le coût de communication / nombre TSV dans les deux techniques est le minimum avec cette valeur, nous n'avons pas exploiter une taille au-delà de 100 échantillons due au limite du temps lors de la réalisation de cette rapport.

4.2.1.4 Etude comparative :

Les benchmarks utilisés sont MPEG et VOPD et Random, sous une architecture Réseau sur Puce 3D. Les valeurs : coûts de communications et nombre TSV moyen sont pris en considération.

VOPD :

La combinaison adéquate est celle qui permet d'avoir des solutions qui optimisent les deux critères en même temps {Communication, Surface}, pour la technique SGANova la combinaison de Pc=0.7, Pm=0.3 permet de donner des solutions qui ont un coût de communication et TSV moyen réduit et qui est égal à : 3619/2 (coût/NbrTSV).

Alors qu'avec l'utilisation de la technique SPEA-II Adapté une solution meilleure en terme {coût de communication, surface} a été trouvée et qui est égale à : 3492/1

MPEG :
Lors de l'utilisation de la technique SGANova avec Pc=0.5, Pm=0.1 une solution avec un coût et Tsv min a été trouvée et qui est égal à : [5749 /1]. De l'autres côté, et en utilisant la technique SPEA-Adapté avec la combinaison de Pc=0.9, Pm=0.4 sa nous a permis de trouver une solution égale à [4756/2] avec un coût de communication plus réduit que celle de SGANova mais elle utilise plus de Tsv les deux solutions min trouvés par les deux techniques sont classé comme des solutions non dominé.

Tableau 10: Récapitulatif des résultats des paramètres

Technique utilisé	Benchmark	probabilité de croissement adéquat	Probabilité de mutation adéquat	Taille population adéquat	Nombre d'itération
SPEA-II Adapté	VOPD	0.7	0.3	100	100
	MPEG	0.9	0.4	100	100
SGANova	VOPD	0.7	0.3	100	100
	MPEG	0.5	0.1	100	100

Discussion :
Le **tableau 11** illustre les meilleurs paramètres trouvés afin d'obtenir des résultats intéressantes en terme de coût de communication et nombre de TSV. Sur les différents tests qu'on a effectués et qui se compose de 5 exécutions pour chaque combinaison. Chaque exécution propose un ensemble de solution. On a calculé le coût et le nombre de liens verticaux moyenne utilisés sur chaque exécution puis on a calculé le coût moyenne et le nombre liens verticaux moyenne par rapport les 5 exécutions. En appuyant sur les différents résultats trouver, on a vu que les deux algorithmes arrivent à trouver des résultats intéressant lors ce que ces paramètres sont choisis.

4.3 Résultats Multi-objective

Le résultat d'une optimisation multi-objectifs est représenté sous forme de graphe ayant comme axes les deux objectifs à optimiser à savoir les coûts communication et en nombre de TSV (surface). Ce graphe est constitué d'un ensemble de solutions non dominées uniquement. Cet ensemble forme le front Pareto.

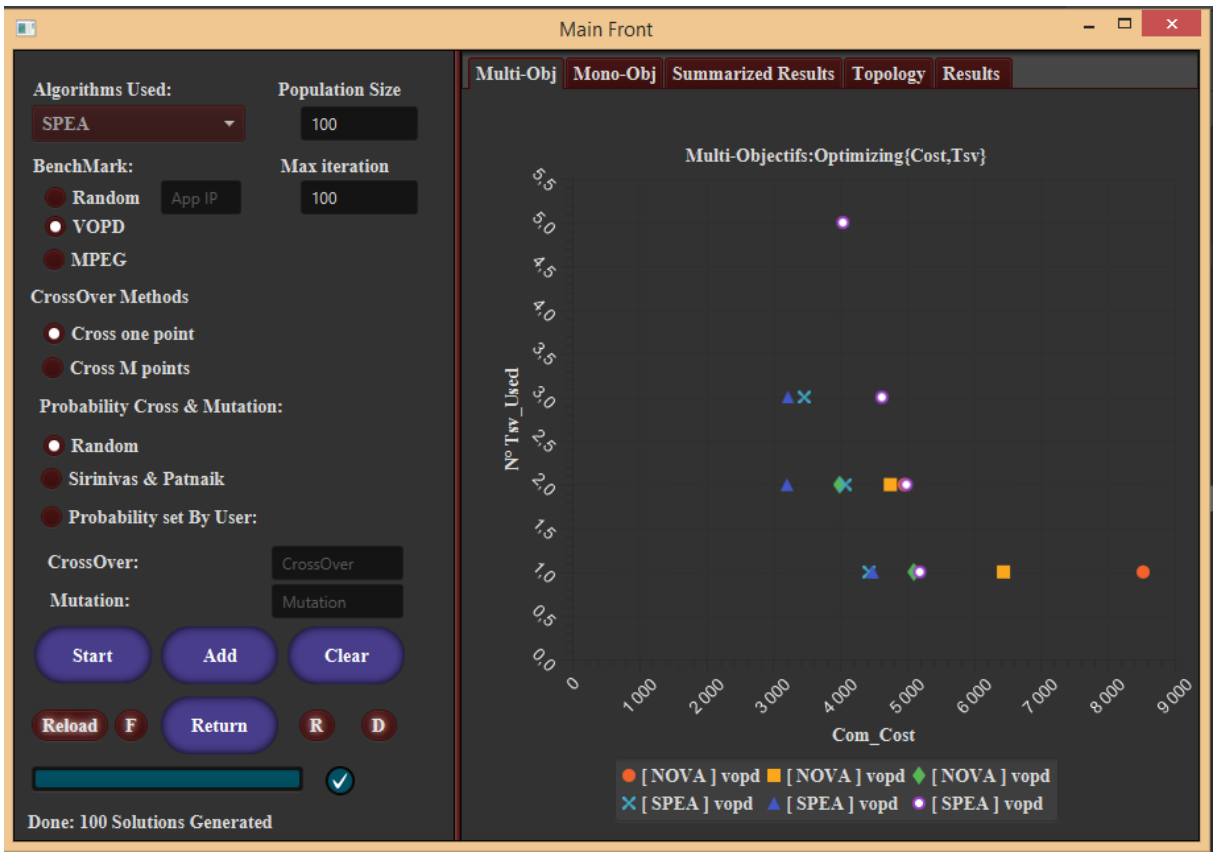

Figure 45: Front Pareto du benchmark VOPD avec les 2 algorithmes

Le **tableau 12** ci-dessous résume les moyennes des résultats obtenus après 15 exécutions. A chaque exécution, on calcule la moyenne des valeurs non dominées en termes de coût et de surface (solutions présentent dans l'archive) **(figure 46).**

Tableau 11: Résultats obtenus après 15 exécutions en utilisant benchmark VOPD

	SPEA-II Adapté	SGANova
Coût de communication Moyenne	5767.91	5874.86
Nombre d'itération moyenne	100	15
Nombre de liens verticale moyen	3	2

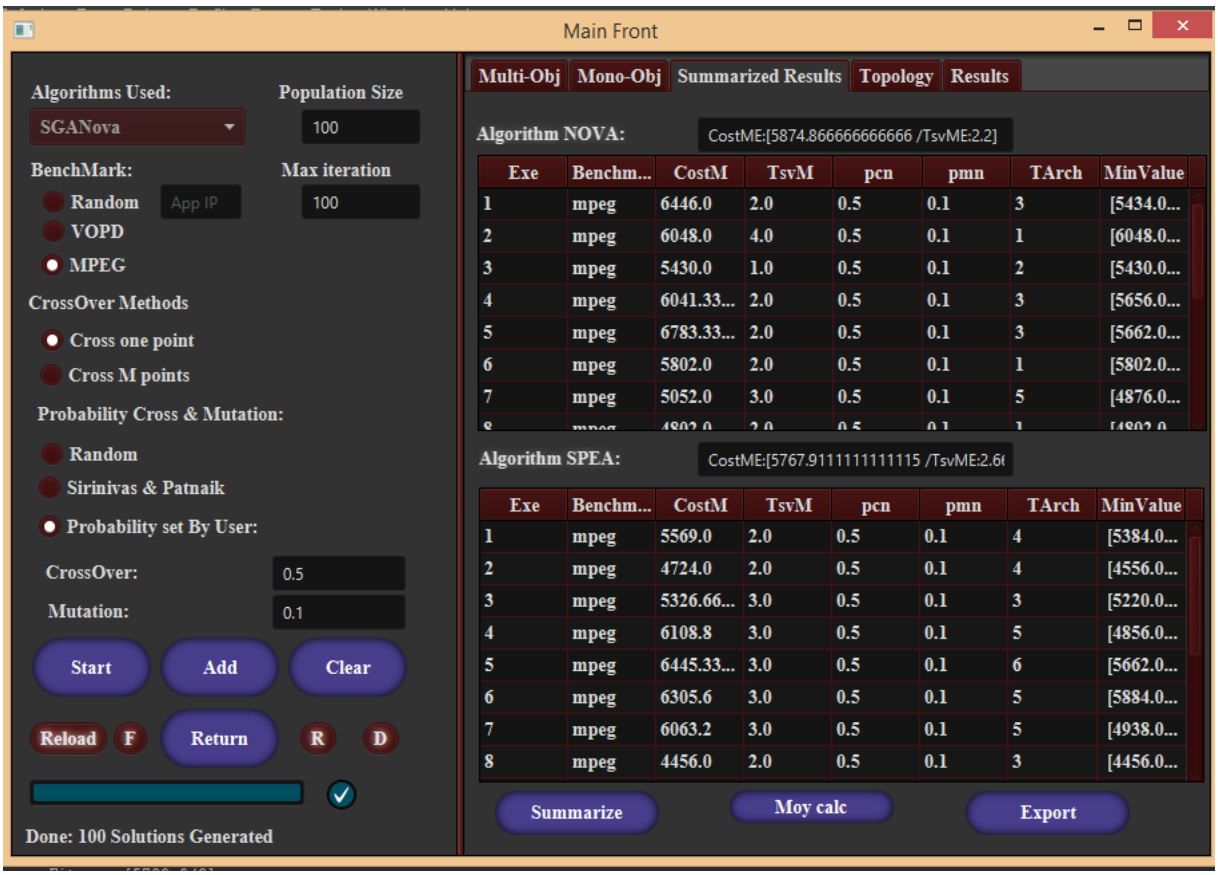

Figure 46: Illustration des 25 exécutions en utilisant Benchmark MPEG

Discussion :

On constate que les résultats moyens obtenus par la SGANova sont en moyenne pas loin des résultats obtenus avec la SPEA-II Adapté en terme de coût de communication. Cependant, le SPEA-II Adapté utilise plus de TSV que le SGANova.

SGANova est intelligent et arrive à trouver des solutions optimales dans un temps rapide (nombre d'itération moyenne), de l'autre côté le SPEA-II Adapté doit itérer à chaque fois, pour juger mieux faut augmenter le nombre de teste de 10 à 100, et constater la différence, le fait d'augmenter le nombre des exécutions va permettre de rencontrer plusieurs populations initiales différentes et du coup d'exploiter plus de solutions.

4.4 Test des résultats de mapping obtenu en fixant un % de TSV

Dans notre application on a proposé un utilitaire qui permet de recalculer pour un mapping sélectionner de l'ensemble de solutions, le coût de consommation tout en fixant le pourcentage des liens vertical utilisé en préalable. Cette utilitaire a pour but de positionner les résultats que l'application génèrent par rapport l'idée de variation sur pourcentage de TSV proposer dans la littérature. Les résultats sont illustrés dans le **tableau 13.**

Paramètres de Test :
1 seule exécution pour chaque benchmark avec 2 technique utilisées.
VOPD: SGANova Pc: 0.7 Pm: 0.3 SPEA: Pc= 0.7, Pm= 0.3;
MPEG: SGANova Pc: 0.5 Pm: 0.1 SPEA: Pc=0.9, Pm=0.4;
Taille population : 100
Méthode de croisement : croisement à un seul point
Nombre itération : 100

Tableau 12: Tests effectués en comparant nos résultats à la littérature

Benchmark	Technique utilisé	Coût de communication min	% TSV trouver générer par notre application 1 Seul Exécution	Coût de com trouver par notre application en appliquant % Tsv fixer	% TSV Fixer Mentionner dans la littérature
VOPD	SGANova	4363	25%	4363	25
	SPEA-Adapté	3749	37.5%	3728	
	SGANova	3660	25%	3586.4	50
	SPEA-Adapté	3330	37.5%	3185.0	
	SGANova	4165	25%	3909	100
	SPEA-Adapté	3042	25%	2946	
MPEG	SGANova	5642	33%	5603	25
	SPEA-Adapté	5200.5	66%	5278.4	
	SGANova	6618.0	16%	6541.2	50
	SPEA-Adapté	5002	50%	5000	
	SGANova	4986	33%	4981	100
	SPEA-Adapté	4986	50%	4957.2	

Discussion :
Afin de pouvoir tester l'impact des réseaux sur puce 3D partiellement interconnecté en considérant plusieurs critères en même temps, on a proposé un outil de calcul qui permet de comparer les solutions multi-objectif obtenu trouvé par l'application avec des solutions mono-objectif ou l'utilisateur fixe le % TSV (une entré pour l'application), les critères sont {coût de communication, surface}

Les deux colonnes à gauche (en couleur verte) du tableau représentent les solutions multi-objectif trouvé par l'application en optimisant {coût de com, surface} en même temps. Les deux colonnes à droite (en couleur jaune) représentent les solutions mono-objectif ou le coût seulement est optimisé et le % TSV est fixé par l'utilisateur.

Les résultats obtenus ont pu montrer que les solutions multi-objectif trouvés ne sont pas loin des solutions mono-objectif en terme de coût de communication et % TSV. 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.

Le but de ce test est de déterminer est ce qu'il existe d'autres pourcentage (configuration TSV) hors ceux déjà mentionner dans la littérature (25, 50 et 100%), qui permettent de donner des performances meilleures ou proche d'optimal pour le réseau sur puce(NoC).

Selon Le **tableau 12** qui illustre les résultats obtenus, notre application a permis de trouver des solutions qui utilisent un pourcentage de TSV en dehors des pourcentages de la littérature (25,50 et 100%). Ces résultats semblent intéressants que ce soit en terme de coût de communication ou de nombre de TSV, comme par exemple lors de l'utilisation du benchmark MPEG et en optimisant les 2 critères en même temps avec la technique SGANova notre application a trouvé une solution qui a un coût de communication égal à 6618.0 et une utilisation de TSV réduite (16%).
De l'autres coté, lorsqu'on a fixé le pourcentage de TSV à 50% le coût de communication trouvé par l'application été égal à 6541.2. Si on compare les 2 résultats trouvés, on remarque que la solution obtenue par l'application (en optimisant les 2 critères) consomme moins de surface que la solution obtenue lors d'optimisation de coût uniquement. En revanche, le coût de consommation des deux solutions trouvées par l'application avec 2 principes n'est pas loin entre eux.

Nous présentons dans ce qui suit notre application et les différents options et fonctionnalités développées.

5. Présentation de l'application

5.1. Interfaces de l'application et fonctionnement :

Interface Accueil :

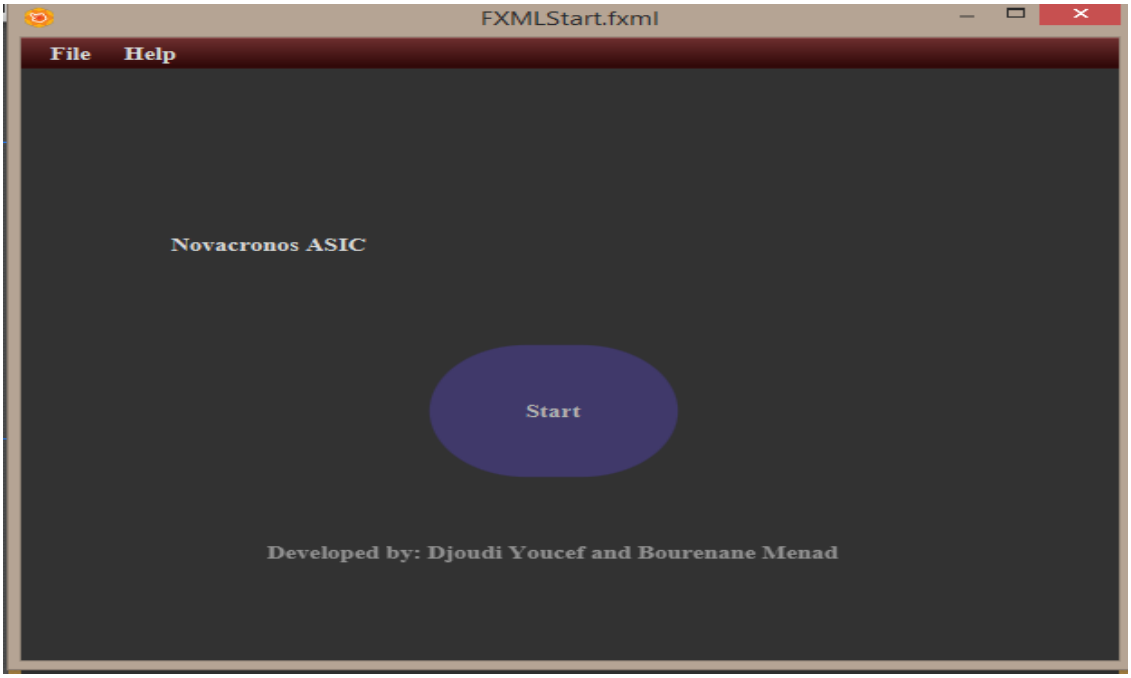

Figure 47: interface accueil pour utilisateur

Interface de travail :

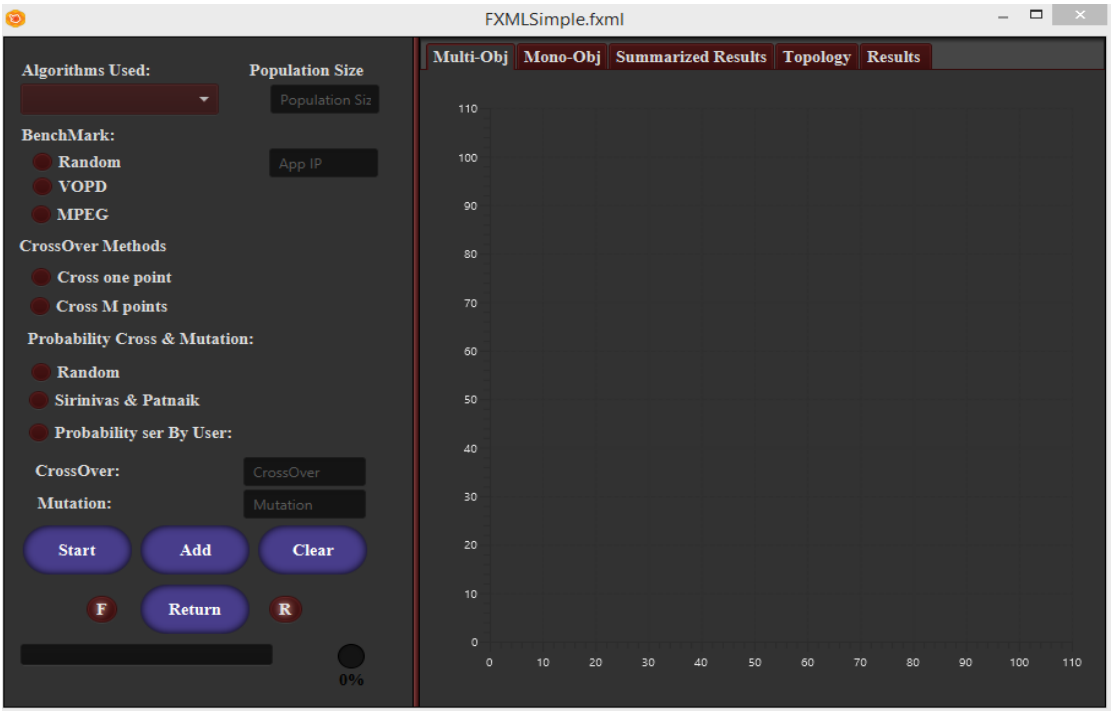

Figure 48: Interface principale de travail

Interface accueil représente la première interface que l'utilisateur va voir après l'exécution de l'application, il a pour but de mentionner les membres du groupe qui ont développer l'application. L'utilisateur clique sur le bouton « START » afin de basculer vers interface main
Le type d'utilisateur que nous allons prés comme un exemple est « concepteur », vu que la grande différence entre ces deux utilisateurs réside dans le savoir-faire dans ce domaine (les probabilités de croisement et de mutation, la méthode sélection et autres sont fixer et simplifier pour l'utilisateur décideur).

Dans l'exemple, le concepteur fait entrée l'ensemble des paramètres présentent dans l'interface qui suit la sélection de type d'utilisateur.

- L'algorithme : SGANova / SPEA
- La taille de la population : donnée utilisateur
- Le benchmark d'une application :
    - VOPD avec 16 composants IPs et 21 liens de communication
    - MPEG avec 12 composants IPs et 13 liens de communication
    - Random avec Nombre IP = 20 composants IPs liés aléatoirement
- Type de croisement : un point / deux points
- Type de probabilités (Pc/Pm) :
    - Probabilités de Sirinivas et Patnaik avec Pc = xx et Pm = xx
    - Probabilités entre par l'utilisateur
    - Probabilités Random :

        Probabilité de croisement : entre 0.1 et 0.9
        Probabilité de mutation : entre 0 et 1

Après avoir choisir les paramètres de la première exécution, l'utilisateur clique sur le bouton « start » de l'interface main. Par le biais du bouton « Add » l'utilisateur peut voir une autre exécution en changeant certains paramètres, aussi faire une comparaison entre les deux algorithmes en gardant les mêmes paramètres.

L'utilisateur peut vider l'espace de travail en cliquant sur le bouton « clear », Le bouton « F » affiche les solutions qui ont moins d'énergie sur le même graphe multi-objective, le bouton « R » sauvegarde les résultats obtenus dans un fichier externe au même temps elle calcule nombre des solutions optimal trouvé pour chaque exécution.
La première fenêtre de l'application affiche l'ensemble des solutions de l'archive en multi-objectives (coût de communication et nombre de TSV) trouvées lors de l'exécution sous un graphe de points. L'axe X représente le coût de communication et l'axe Y représente le nombre de TSV utilisé).

La deuxième fenêtre affiche le même ensemble de solutions mais avec un aspect mono-objectif de l'archive (dans notre cas le coût de communication). **(Figure 49)**

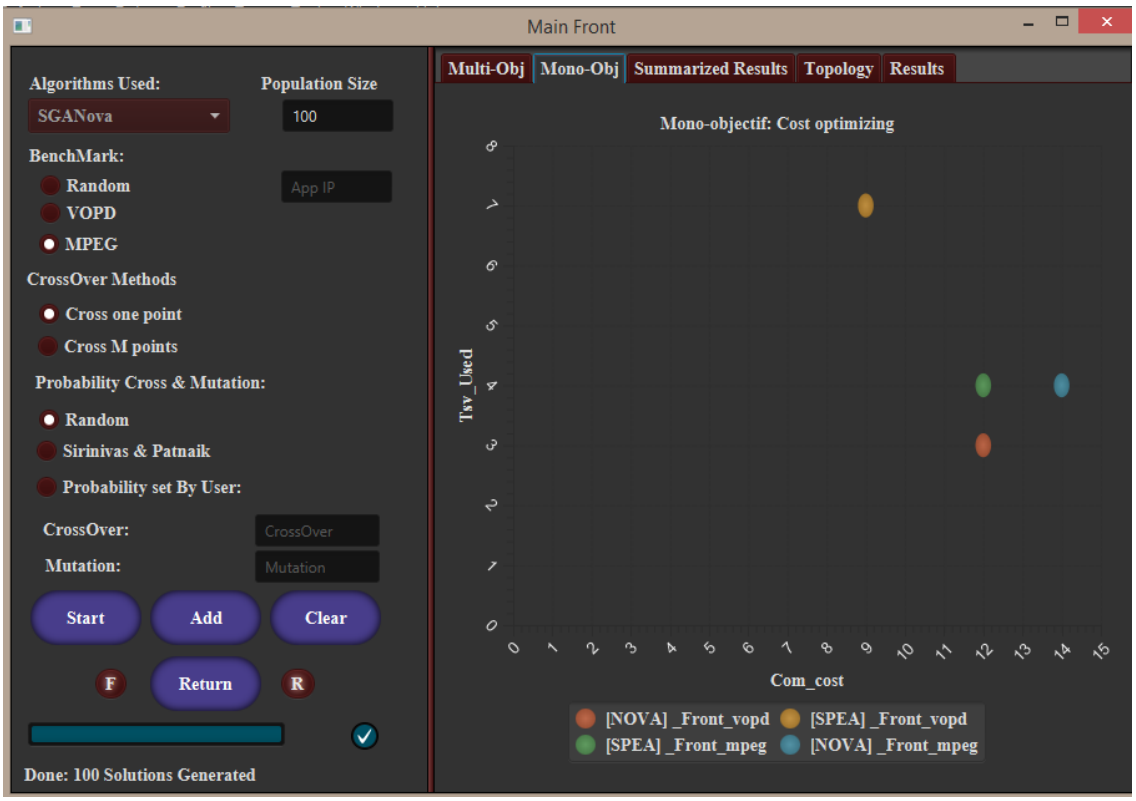

Figure 49:solution mono-objectif (optimisation de coût)

La troisième fenêtre résume les résultats obtenus jusqu'ici dans deux tableaux (un pour chaque algorithme). En cliquant sur le bouton « summarize », les tableaux se remplissent avec les informations suivantes : le numéro de l'exécution, le benchmark, les probabilités Pc/Pm ainsi le coût de communication et le nombre des TSV Moyenne. « Moy Calc » permet d'affiche la valeur moyenne des exécutions pour chaque algorithme dans le champ «Execution Moy de chaque algo». Une exportation des résultats trouvés peut se faire en format «fichier.xis ou fichier.csv» **(Figure 50).**

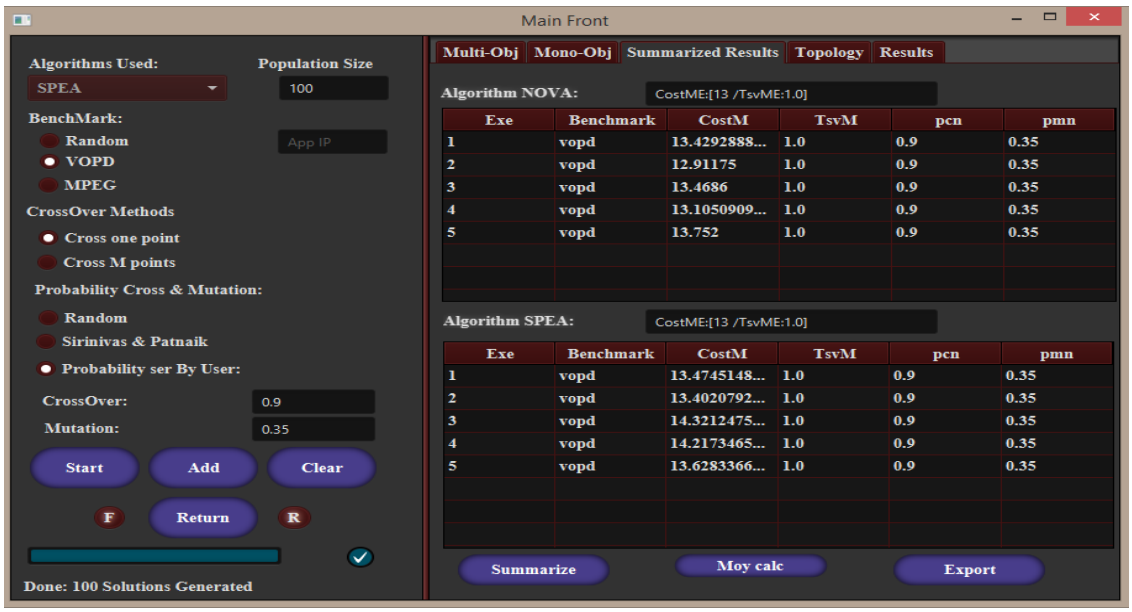

Figure 50: Affichage après le clique sur summarize

La fenêtre « Topolgy » donne la main à l'utilisateur pour voir la topologie de la solution. Il choisit d'abords : l'algorithme, le numéro de l'exécution dans cette algorithme, puis il clique « Display » et une liste de solutions est affichée **(Figure 51)**. Il clique sur « mapping selected » pour choisir la solution, et une fenêtre est apparu après le clique sur « GTpology » **(Figure 52).**

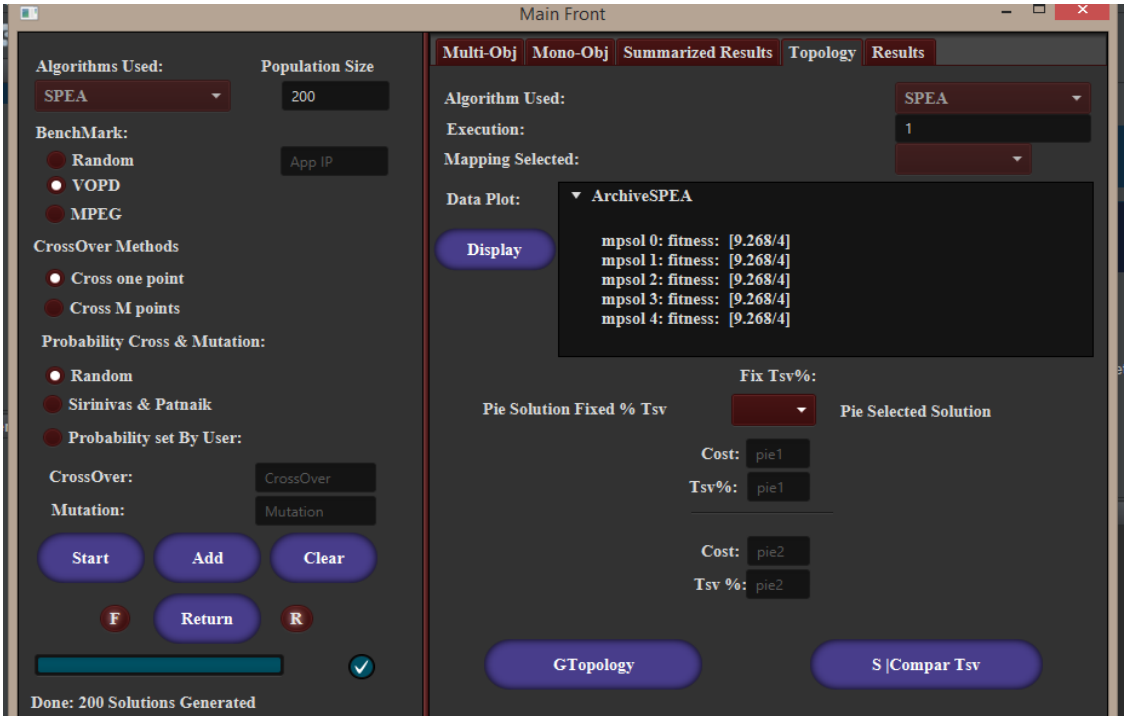

Figure 51 :Affichage après le clique sur bouton Display

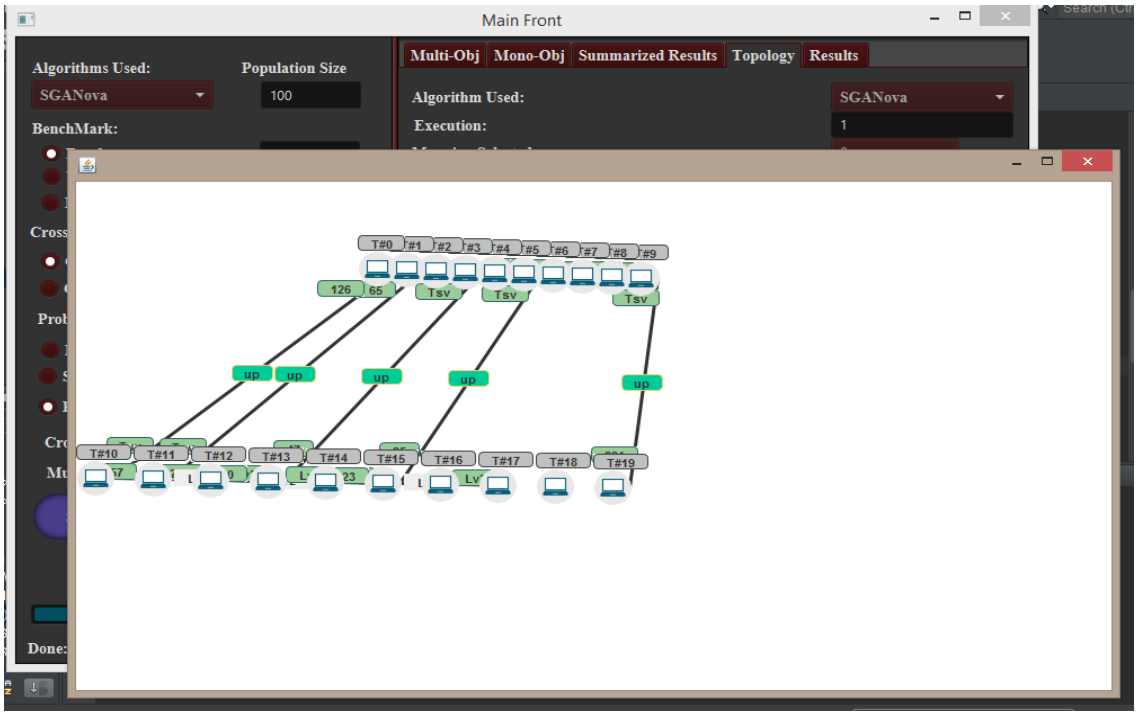

Figure 52 : Génération de topologie avec les différents LV affectés

L'utilisateur peut même faire une comparaison à l'aide du bouton « S|Compar TSV » entre la solution choisi (mappingSol + TSVSol) avec une autre solution (mappingSol + TSV%) qui se diffère par le pourcentage de TSV, en affichant des pie charte (forme graphique) qui indique le

coût de communication et le progrès du nombre LV **(figure 53).**

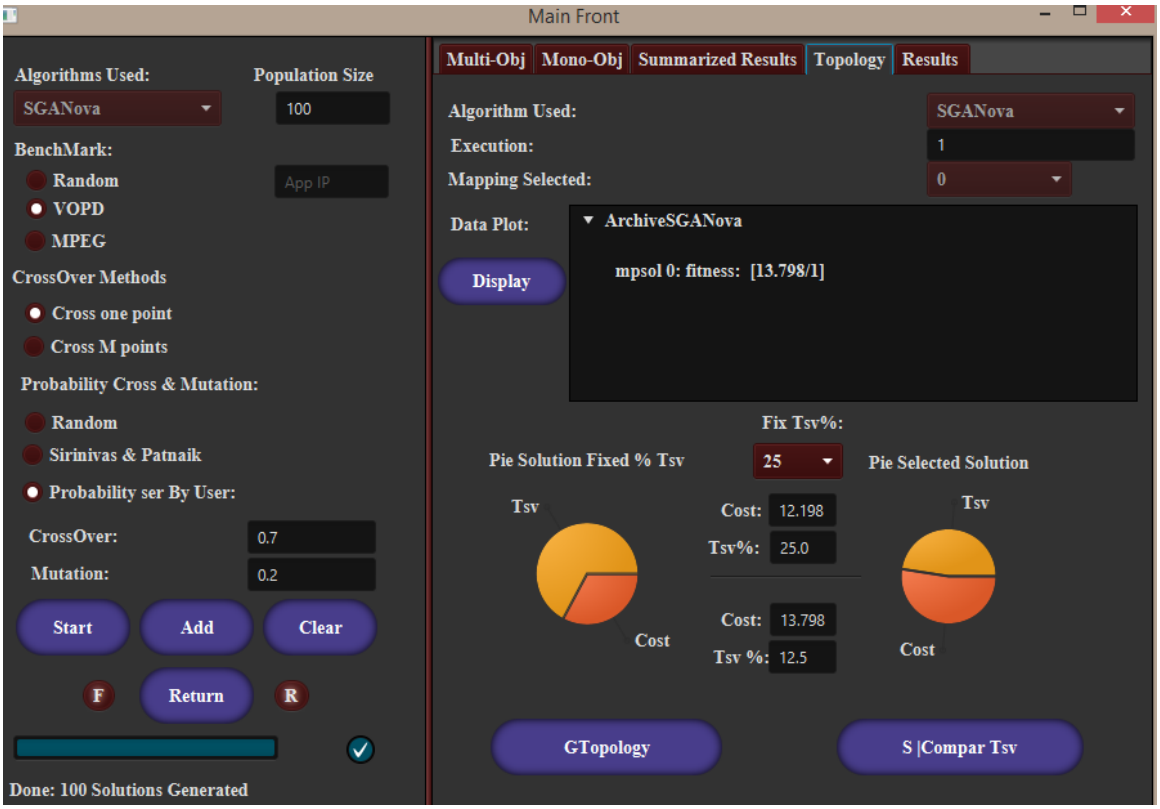

Figure 53: l'utilitaire de comparaison en %TSV des NoC partiellement interconnectés

6. Conclusion

Dans ce chapitre, nous avons présenté les deux métaheuristiques proposées. Nous avons commencé par introduire notre problème en formule mathématique, puis on a présenté notre supposition. Ensuite, on a décrit les algorithmes utilisés {SGANova, SPEA-II Adapté}. Et nous avons terminé cette partie de conception par des tests de validation de notre proposition et des discussions sur chaque paramètre des 02 algorithmes ainsi l'étude comparatives entre ces deux derniers.

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

Conclusion générale

Les travaux étudiés et réalisés dans le cadre de ce projet de fin d'étude ont permis de montrer l'intérêt d'utiliser la structure de communication réseaux sur puce pour résoudre les problèmes de communication dans les systèmes sur puce.

Un nombre important de chercheurs travaillent toujours sur ce nouveau paradigme, en concluant à chaque fois de nouveaux défis de conception. Parmi ces défis, est le mapping, la génération des topologies personnalisées ou l'utilisation de la technologie 3D.

La résolution des problèmes de conception en combinant plusieurs phases au même temps font appel aux méthodes d'optimisation combinatoire multi-objectives. C'est dans ce contexte là que nous avons étudié quelques méthodes heuristiques et métaheuristiques liées aux problèmes des réseaux sur puce. En mettant l'accent sur l'aspect optimisation dans la phase mapping et l'utilisation des topologies personnalisées en utilisant la technologie 3D, vu qu'il n'existe pas des plateformes ou des simulateurs qui automatisent ou prennent en compte toutes ces étapes à la fois.

L'approche choisi ici suit le principe des métaheuristiques, spécialement les algorithmes génétiques. Le choix de cette approche est argumenté par le fait que :

1- Les heuristiques demandent beaucoup de savoir-faire et d'expertise dans le domaine, de ce fait, on a opté pour les méthodes métaheuristiques.
2- Ainsi que, les métaheuristiques travaillent avec l'aspect solutions à population, de ce fait, on a choisi aléatoirement (choix personnel) de travailler avec les AGs.
3- Dans notre cas, on s'est bien étaler dans le domaine des AGs, c'est pour ce là que nous avons opté dans notre étude à aspect multi-objectif.

Afin de mettre en évidence certains aspects de la recherche, nous avons proposés deux algorithmes adapté (SGANova et SPEA-II Adapté). Notre solution donne un ensemble de solutions mapping optimales, tout en minimisant de nombre de liens verticaux. Le choix des meilleures solutions suit l'approche Pareto afin de présenter les solutions de compromis qui minimisent le coût de communication et de surface dans le système.

Dans les expérimentations réalisées, nous avons pu observer l'efficacité de notre méthode en la comparant à celle de la littérature, en prouvant l'intérêt d'utiliser des architectures 3D partiellement connectés verticalement. Néanmoins, La qualité des solutions trouvées par notre solution dépend du choix de paramétrage des métaheuristiques (taille de la population, les probabilités P_c et P_m etc.) et de l'équilibre entre le coût de communication et le coût de surface.

Un des obstacles remarqué pendant notre recherche est le manque de simulateurs gratuits qui prennent en considération les topologies personnalisées. De ce fait, nous nous sommes basé seulement sur des métriques indépendantes de toute technologie : coût de communication pour simuler le délai et la consommation d'énergie qui sont proportionnelles à la longueur des chemins parcourus. Et nombre de liens verticaux afin d'évaluer la surface occupée par les TSVs.

C’est dans nos perspectives, que nous voulons approfondir plus dans ce domaine et améliorer notre contribution, en terme de :

Valider notre contribution par la communauté scientifique.

Réaliser une interface simple utilisateur, en automatisant la phase d’étude de paramètres.

Procéder au reste des phases de conception des réseaux sur puce 3D, tel que le routage et floorplanning.

Améliorer notre solution algorithmique en cherchant d’autres approches de mapping (statiques, dynamique) et d’autres topologies (Torus, BFT, …).

Explorer d’autres méthodes d’optimisation combinatoire multi-objectives liées aux problèmes de mapping, aux topologies personnalisées et à utilisation de la technologies 3D.

Exploiter d’autres solutions métaheuristiques et intégrer des solutions heuristiques dans quelques phases de l’algorithme proposé, tels que : la génération de la population initiale, le choix des meilleurs individus, etc.

Références

[1] Hu, J. et al. (2002). ***System-Level Point-to-Point Communication Synthesis Using Floorplanning Information***. VLSI Design and Design Automation, India.

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

[3] Abderrahim, Chariete. (2014). ***Approches d'optimisation et de personnalisation des réseaux sur puce (NoC : Networks on Chip)***. Université de Technologie de Belfort-Montbéliard (UTBM).

[4] Lemaire, R. (2006). ***Conception et modélisation d'un système de contrôle d'applications de télécommunication avec une architecture de réseau sur puce (NoC).*** Thèse de doctorat, Micro et nanotechnologies / Microélectronique, Institut National Polytechnique de Grenoble – INPG.

[5] Flynn, D. (1997). ***Amba : enabling reusable on-chip designs***. IEEE Micro 17, p. 20–27.

[6] Hofmann, R. et Drerup, B. (2002). ***Next generation CoreConnect™ processor local bus architecture.*** Proc. 15th Annual IEEE International ASIC/SOC Conference, p. 221–225.

[7] Cordan, B. (1999). ***An efficient bus architecture for system-on-chip design***. Proc. IEEE 1999, Custom Integrated Circuits Conference, p. 623–626.

[8] STMicroelectronics (2001). ***STBus communication system : Concept and definitions.*** Tech. report, STMicroelectronics.

[9] Wingard, D. (2001). ***Micronetwork-based integration for SOCs.*** Proc. Design Automation Conference, p. 18–22.

[10] PI-Bus Systems Toolkit,
http://cordis.europa.eu/esprit/src/results/pages/infoind/infind24.html

[11] OpenCores.org (2002). ***Wishbone, revision b.3 specification***. Tech. Report

[12] Slavisa, J. (2009). ***Architecture reconfigurable de système embarqué auto-organisé***. Thèse de Doctorat

[13] Tran, Xuan-Tu. (2019). ***Méthode de Test et Conception en Vue du Test pour les Réseaux sur Puce Asynchrones : Application au Réseau ANOC***.

[14] Kameche, A. H. (2013). ***Approches basées sur la BBO pour le Data Clustering et le NoC Mapping***, Mémoire de magister. Ecole Supérieure d'Informatique ESI, Algérie.

[15] Fernandez-Alonso, Eduard et al. (2012). ***Survey of NoC and Programming Models Proposals for MPSoC***. International Journal of Computer Science Issues. 9.

[16] BOUGUETTAYA, Abdelmalek. (2017). ***Génération d'un réseau sur puce au format VHDL RTL à partir d'une modélisation de haut niveau UML par raffinement.*** Thèse de doctorat, université Badji Mokhtar, Annaba.

[17] DELORME, Julien. (2007). ***Méthodologie de modélisation et d'exploration d'architecture de réseaux sur puce appliquée aux télécommunications***, Thèse de doctorat. Institut national des sciences appliquées de Rennes.

[18] Marescaux, T. et al. (2002). ***Interconnections Networks Enable Fine-Grain Dynamic Multi-tasking on FPGAs***. Field-Programmable Logic and Applications, France.

[19] Karim, F. et al. (2002). ***An interconnect architecture for networking systems on chip***. IEEE Micro, Vol 22, n° 12, 36-45

[20] Karim F., et al. (2001). ***On chip communication architecture for OC-768 network processors***. Design Automation, USA.

[21] Chen, Y. et al. (2013). ***Topology and mapping co-design for complex communication systems on wireless NoC platforms.*** IEEE Conference on Industrial Electronics and Applications, Australia.

[22] Elmiligi, H. et al. (2013). ***Power consumption of 3D networks-onchips: Modeling and Optimization.*** Microprocessors and Microsystems, Vol. 37, n° 10.

[23] Wang, J. et al. (2011). ***Latency-aware mapping for 3D NoC using rank-based multi-objective genetic algorithm.*** International Conference on ASIC, China.

[24] Ying, H. et al. (2012). ***A genetic algorithm based optimization method for low vertical link density 3-dimensional networks-on-chip many core systems.*** NORCHIP, Denmark.

[25] Sotiriou-Xanthopoulos, E. (2014). ***A framework for rapid evaluation of heterogeneous 3-D NoC architectures***. Microprocessors and Microsystems, Vol 38, n°6.

[26] Xu, T. C. et al. (2010). ***A study of Through Silicon Via impact to 3D Network-on- Chip design.*** International Conference On Electronics and Information Engineering, Japan.

[27] Xu, T. C. et al. (2013). ***Optimal placement of vertical connections in 3D Network-on-Chip.*** Journal of Systems Architecture, Vol. 59, n°8.

[28] Manna, K. et al. (2015). ***TSV Placement and Core Mapping for 3D Mesh Based Network-on-Chip Design Using Extended Kernighan-Lin Partitioning.*** IEEE Computer Society Annual Symposium on VLSI, France.

[29] Toubaline, N. (2018). ***Aide à la conception d'un réseau sur puce pour un système intégré***, thèse de doctorat. Université Saad Dahlab Blida, Algérie.

[30] Mohamed Fehmi, Chatmen. (2016). ***Conception d'un réseau sur puce optimisé en latence***. Electronique, Université de Bretagne Sud.

[31] Rantala, Ville et al. (2006). ***Network on Chip Routing Algorithms***. TUCS Technical Report No 779.

[32] Jie, Wu. (2003). ***A fault-tolerant and deadlock-free routing protocol in 2d meshes based on odd-even turn model***. IEEE Transactions on Computers, Vol 52, No 9.

[33] Boppana, R.V. & Chalasani, S. (1995). ***Fault-tolerant wormhole routing algorithms for mesh networks***. IEEE Transactions on Computers, Vol 44, No 7.

[34] Chen, K.H. & Chiu, G.M. (1998). ***Fault-tolerant routing algorithms for meshs without using virtual channels***. Journal of Information Science and Engineering, Vol 14, n°12.

[35] Ali, M. et al. (2005). ***A Dynamic Routing Mechanism for Network on Chip***. 23rd NORCHIP Conference.

[36] Wolkotte, P. T. et al. (2005). ***An energy-efficient reconfigurable circuit switched network-on-chip.*** In Proceedings of the 19th IEEE international parallel and distributed processing symposium.

[37] Kouadri-Mostéfaoui, A. M. (2009). ***Architectures Flexibles pour la Validation et l'Exploration de Réseaux Sur Puce,*** Thèse Doctorat.

[38] Moussa, H. (2009). ***Architectures de réseaux sur puce pour de codeurs canal multiprocesseurs***, Thèse Doctorat.

[39] Quartana, J. (2004). ***Conception de réseaux de communication sur puce asynchrones : Application aux architectures GALS,*** Thèse Doctorat.

[40] Toubaline, N. & al. (2017). ***A Classification and Evaluation Framework for NoC Mapping Strategies***. Journal of Circuits Systems and Computers.

[41] Manna, Kanchan et al. (2014). ***Through silicon via placement and mapping strategy for 3D mesh based Network-On-Chip***

[42] Davis, W. R. et al. (2005). ***Demystifying 3D ICs: the pros and cons of going vertical***. IEEE Design & Test of Computers, vol. 22, pp. 498–510.

[43] Karima, M. (2003). ***L'optimisation multi objectif et l'informatique quantique***, Mémoire Magister. Faculté des sciences de l'Ingénieur Département d'Informatique, Université Mentouri – Constantine. (http://tiny.cc/c2um9y)

[44] Barichard, V. (2003). ***Approches hybrides pour les problèmes multi objectifs***, Phd Thesis. Laboratoire d'Etude et de Recherche en Informatique d'Angers, Ecole doctorale d'Angers. (http://tiny.cc/pqum9y)

[45] Fourman. (1985). ***Compaction of symbolic layout using genetic algorithms***. In Genetic Algorithms and their Applications: Proceedings of the First International Conference on Genetic Algorithm, pages 141–153.

[46] Dréo, J. et al. (2003). ***Métaheuristiques pour l'optimisation difficile***. Eyrolles (Editions).

[47] Back, T. et al. (1997). ***Handbook of Evolutionary Computation***. Institute of Physics Publishing and Oxford University Press.

[48] Holland, J.H. (1975). ***Adaptation in natural and artificial systems***. Ann Arbor: the University of Michigan Press.

[49] Goldberg, D. E. (1989). ***Genetic Algorithm in Search, Optimization, and Machine Learning***. MA: Addison-Welsley.

[50] Schwefel, H-P. (1981). ***Numerical Optimization of Computer Models***. Wiley, Chichester.

[51] Fogel, D. (2000). ***Evolutionary Computation: Toward a New Philosophy of Machine Intelligence (second edition).*** IEEEE Press.

[52] Jozefowiez, N. (2002). ***Une métaheuristique parallèle et hybride pour un problème de tournées de véhicules multicritères.*** ROADEF'2002.

[53] Jaszkiewicz, A. (2002). ***Genetic local search for multiple objective combinatorial optimization***. European Journal of Operational Research.

[54] Czyżżak, P & Jaszkiewicz A. (1998). ***A Pareto simulated annealing - A metaheuristic technique for multiple-objective combinatorial optimization***. Journal of Multi-Criteria Decision Analysis.

[55] Ilhem, B. (2013). ***Perfectionnement de métaheuristiques pour l'optimisation continue.*** MSTIC, Université Paris-Est Créteil. **(http://tiny.cc/wwum9y)**

[56] Darwin, C. (1859). ***On the origins of species by means of natural selection or the Preservation of Favoured Races in the Struggle for Life***. J. Murray, London.

[57] Goldberg, D. E. (1989). ***Genetic Algorithms in Search, Optimization and Machine Learning (1st ed.).*** Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

[58] Goldberg, D. E. (1990). ***Real-coded genetic algorithms, virtual alphabets, and blocking***. Urbana.

[59] Benhama, A. & Benkhelouf, Y. (2015). ***Étude Comparative Des Algorithmes Génétiques Multi-Objectifs***, Mémoire Master. Département d'Automatique, Télécommunication et Electronique, Université Abderrahmane MIRA de Bejaia.

[60] Baker, J. E. (1985). ***Adaptive selection methods for genetic algorithms***. In Proceedings of an International Conference on Genetic Algorithms and their applications.

[61] Goldberg, D. E. & Deb, K. (1991). ***A comparative analysis of selection schemes used in genetic algorithms***. Foundations of genetic algorithms.

[62] Man, K. F. et al. (1996). ***Genetic algorithms: concepts and applications [in engineering design].*** Industrial Electronics, IEEE Transactions on.

[63] Srinivas, N. & Deb, K. (1994). ***Muiltiobjective optimization using nondominated sorting in genetic algorithms***. Evolutionary computation.

[64] Deb, K. et al. (2002). ***A fast and elitist multiobjective genetic algorithm: NSGA-II***. Evolutionary Computation, IEEE Transactions on.

[65] Sareni, B. et Krähenbühl, L. (1998). ***Fitness sharing and niching methods revisited***. *Evolutionary Computation, IEEE Transactions on*, *2*(3), 97-106.

[66] Zitzler, E. et al. (2001). ***SPEA2: Improving the Strength Pareto Evolutionary Algorithm.*** Technical Report 103, Computer Engineering and Communication Networks Lab (Tik), Swiss Federal Institute of Technology (ETH) Zurith.

[67] Zitzler, E. & Thiele, L. (1998). ***An evolutionary algorithm for multiobjective optimization: the strength Pareto approach***. Technical report, Swiss Federal Institute of technology, Zurich.

[68] Silverman, B. W. (1986). ***Density estimation for statistics and data analysis.*** (Vol. 26), London: Chapman and Hall

[69] Kumar Sahu, P. & Chattopadhyay, S. (2013). ***A survey on application mapping strategies for Network-On-Chip design***. Journal of Systems Architecture.

[70] Ascia, G. et al. (2004). ***Multi-objective mapping for mesh-based NoC architectures***. Second IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and Systems Synthesis, Sweden.

[71] Ascia, G. et al. (2006). ***A multi-objective genetic approach to mapping problem on network-on-chip.*** Journal of Universal Computer Science, vol. 12, n°. 4. (http://tiny.cc/rnz4bz)

[72] Ascia, G.et al. (2005). ***Mapping cores on network-on-chip***. International Journal of Computational Intelligence Research, Vol.1, No.2, Research India Publications.

[73] Jena, R. K. et Mahanti, P. K. (2008). ***Design space exploration of network-on-chip: A system level approach***, International Journal of Computing and ICT Research, Vol. 2, n°. 1. (http://tiny.cc/hcy4bz)

[74] Lei, T. et Kumar, S. (2003). ***A two-step genetic algorithm for mapping task graphs to a network on chip architecture***. Euromicro Symposium on Digital Systems Design, Turkey.

[75] Janidarmian, M. & Roshan Fekr, A. (2012). ***A Survey of Meta-Heuristic Solution Methods for Mapping Problem in Network-on-Chips***. Advanced Materials Research. (http://tiny.cc/wqz4bz)

[76] Jena, R. & Ku. Sharma, G. (2007). ***A multi-objective evolutionary algorithm based optimization model for network-on-chip synthesis***. International Journal of Innovative Computing and Applications, International Conference on Information Technology, USA.

[77] Hu, J. et Marculescu, R. (2003). ***Energy-aware mapping for tile-based NoC architectures under performance constraints***. Asia and South Pacific Design Automation Conference, Japan.

[78] Lu, Z. et al. (2008). ***Cluster-based simulated annealing for mapping cores onto 2D mesh networks on chip***. IEEE Workshop Design and Diagnostics of Electronic Circuits and System, USA.

[79] Tavanpour, M. et al. (2009). ***Chain mapping for mesh based network on chip architecture***. IEICE Electronics Express, Vol. 6, n°11

[80] Saeidi, S. et al. (2009). ***Crinkle: A heuristic mapping algorithm for network on chip***. IEICE Electronics Express, Vol. 6, n°12.

[81] Mehran, A. et al. (2007). ***SPIRAL: A heuristic mapping algorithm for network on chip.*** IEICE Electronics Express, Vol. 4, n°8.

[82] Janidarmian, M. et al. (2009). ***Onyx: A new heuristic bandwidth-constrained mapping of cores onto tile-based Network on Chip***.

[83] Hu, J. & Marculescu, R. (2005). ***Energy and performance-aware mapping for regular NoC architectures.*** IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 24, n°4

[84] Carvalho, Ewerson al. (2009). ***Optimal application mapping on NoC infrastructure using NSGA-II and MicroGA***. Proc. Conf. IEEE Intelligent Engineering Systems, pp. 83–88.

[85] Guangyu, C. et al. (2008). ***Application mapping for chip multiprocessor.*** IEEE/ACM Design Automation Conference, USA.

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

[87] Moein-Darbari, F. et al. (2009). ***CGMAP: A new approach to network-on-chip mapping problem.*** IEICE Electronics Express, Vol. 6, n°1.

[88] Ghosal, P. et Karmakar, S. (2012). ***Diametrical mesh of tree (D2D-MoT) architecture: A novel routing solution for NoC.*** ACEEE International journal on Communications, Vol.03, n°3

[89] Kim, J. G. et Kim, Y. D. (2003). ***A linear programming-based algorithm for foorplanning in VLSI design.*** IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 22, n°4.

[90] Srinivasan, K. et al. (2004). ***Linear programming based techniques for synthesis of network-on-chip architectures.*** IEEE International Conference on Computer Design: VLSI in Computers and Processors, USA.

[91] Srinivasan, K. et al. (2006). ***Linear programming based techniques for synthesis of network-on-chip architectures.*** IEEE Transactions on Very Large Scale Integration (VLSI) Systems. Vol. 14, n°4.

[92] Mahdoum, A. (2012). ***A New Design Methodology of Networks On Chip.*** IEEE/ASQED (Asian Symposium on Quality Electronic Design), Malaysia.

[93] Chath, K. S. et al. (2008). ***Automated Techniques for Synthesis of Application-Specific Network-on-Chip Architectures***. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 27, N°. 8.

[94] Chariete, A. et al. (2011). ***Une approche de personnalisation des architectures pour réseaux d'interconnexion sur puce.*** Les Journées Doctorales d'Informatique et Réseaux (JDIR), Belfort, France

[95] Srinivasan, K. et Chatha, K. (2005). ***ISIS: A genetic algorithm based technique for custom on chip interconnection network synthesis***. International Conference on VLSI Design, India.

[96] Li, X. (2009). ***Méthodologie de Conception Automatique pour Multiprocesseur sur puce Hétérogène***. Thèse de doctorat.

[97] Joyner, J. W. et al. (2001). ***Impact of three dimensional architectures on interconnects in gigascale integration.*** IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol. 9, n°12.

[98] Pavlidis, V. F. et Friedman, E. G. (2005). ***Interconnect delay minimization through interlayer via placement in 3-D ICs.*** ACM Great Lakes symposium on VLSI, USA.

[99] Joyner, J. et al. (2001). ***A stochastic global net-length distribution for a three dimensional system-on-a-chip (3d-soc)***. IEEE International ASIC/SOC Conference, USA.

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

[101] Feero, Brett & Pande, Partha. (2009). ***Pande, P.: Networks-on-Chip in a Three-Dimensional Environment: A Performance Evaluation***. IEEE Transactions on Computers 58(1), 32-45.

[102] Jacob, P. et al. (2005). ***Predicting the Performance of a 3D Processor-Memory Stack.*** IEEE Design and Test of Computers, vol. 22, no. 6, pp. 540-547.

[103] Li, F. et al. (2005). ***Design and Management of 3D Chip Multiprocessors Using Network-in-Memory.*** Proc. 33rd Int'l Symp. Computer Architecture (ISCA '06), pp. 130-141.

[104] Toshiba system catalog, "System-in-Package", 2004, disponible en ligne: (Avril 2017) http://bbs.hwrf.com.cn/downebd/33213d1215758969- system_in_package_toshiba_1606.pdf

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

[106] Chang, L. et Lim, S. K. (2012). ***A design tradeoff study with monolithic 3D integration.*** International Symposium on Quality Electronic Design, USA.

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

[108] Pasricha, S. (2009). ***Exploring serial vertical interconnects for 3d ics.*** in Design Automation Conference (DAC), pp. 581 –586.

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

[110] Buckler, M. et al. (2013). ***Low-power Networks-on-Chip: Progress and remaining challenges.*** IEEE International Symposium on Low Power Electronics and Design (ISLPED), pp. 132–134.

[111] CARVALHO, Ewerson et al. (2009). ***Evaluation of Static and Dynamic Task Mapping Algorithms in NoC-Based MOEPCs***. Proceedings of the 11th international conference on System-on-chip.