Republication Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Saad Dahleb Blida

Faculté des sciences

Département d'informatique

Mémoire de fin d'étudie pour l'obtention du diplôme de Master

Spécialité : Systèmes Informatiques et Réseaux

Thème :

Mapping optimal à l'aide de l'algorithme aigle royal

Réalisé par :

Akila Osmani

Habiba Maddi

Soutenu devant un jury constitué de :

Mme. Toubaline Nesrine	Présidente
Mr. Kameche Abdallah Hicham	Examinateur
Mme. Boumahdi Fatima	Promotrice

Année universitaire : 2021/2022

Résumé

Les systèmes sur puce (SoC) intègrent un système complet dans une seule puce. Le nombre de composants intégrés continue d'augmenter pour répondre aux exigences des applications d'aujourd'hui. Ainsi, la conception des SoC se concentre davantage sur les communications que sur les calculs. Le réseau sur puce (NoC) est apparu comme une solution viable aux goulots d'étranglement de la communication sur puce. C'est similaire à un réseau général mais avec des ressources limitées.

Le moyen le plus simple de connecter les ressources et les commutateurs disponibles est de les organiser à l'aide de topologies, la topologie maillée bidimensionnelle est la plus populaire utilisée dans NoC en raison de sa simplicité, mais le Noc bidimensionnel n'a pas été pensé pour répondre aux exigences du système sur puce à bien des égards.Itegrate le wifi dans le NoC, Ou réseau sur puce tridimensionnel (NoC 3D) sont des solutions efficaces au problème de la complexité d'interconnexion des SoC à grande échelle.

But principal de notre projet est de développer une technique de mapping des taches d'une application dans une architecture de réseau sur puce à base de méta-heuristiques qui va minimiser le coût des communications. Nous proposons d'utiliser un nouvel algorithme basé sur l'aigle royal.

Mots-clés : Réseau sur puce bidimensionnel avec fils, Réseau sur puce bidimensionnel sans fils, réseau sur puce tridimensionnel, SoC, Mapping, méta-heuristique, coût de communications, l'algorithme d'optimisation des aigles royal.

Abstract

Systems-on-Chip (SoC) integrate a complete system into a single chip. The number of integrated components continues to increase to meet the requirements of today's applications. Thereby, the design of SoCs focuses in communications more than computations. Network-on-Chip (NoC) emerged as a viable solution to on-chip communication bottlenecks. It's similar to a general network but with limited resources.

The simplest way to connect the available resources and switches is arranging them using topologies, the two-dimensional mesh topology is the most popular used in NoC due to its simplicity however the two-dimensional NoC has not been thought to meet the requirement of System-on-Chip in many aspects. Integrate the WIFI in the NoC, Or Three-dimensional network on chip (3D NoC) are an effective solution to the problem of interconnection complexity of largescale SoC

The main goal of our project is to develop a technique for mapping the tasks of an application in a network-on-chip architecture based on meta-heuristics that will minimize the cost of communications. We propose to use a new algorithm based on the golden eagle.

Keywords: Two-dimensional network on chip,2D Wireless NoC, Three-dimensional network on chip, Mapping, meta heuristic, communications cost, Golden eagle Optimization

الملخص

الأنظمة على الشريحة (SoC) تدمج نظاما كاملا في شريحة واحدة. يستمر عدد المكونات المتكاملة في الزيادة لتلبية متطلبات تطبيقات اليوم. وبالتالي، يركز تصميم SoCs على الاتصالات أكثر من الحسابات. ظهرت الشبكة على الرقاقة (NoC) كحل قابل للتطبيق لاختناقات الاتصالات على الشريحة. إنها تشبه شبكة عامة ولكن بموارد محدودة.

أبسط طريقة لربط الموارد والمفاتيح المتاحة هي ترتيبها باستخدام الطبولوجيات، طوبولوجيا الشبكة ثنائية الأبعاد هي الأكثر استخداما في أكاسيد النيتروجين بسبب بساطتها ولكن لم يعتقد أن شهادة عدم الممانعة ثنائية الأبعاد تلبي متطلبات النظام على الرقاقة في العديد من الجوانب. يعد تكرار WIFI في NoC، أو الشبكة ثلاثية الأبعاد على الشريحة3D-NoC) حلا فعالا لمشكلة تعقيد الربط البيني لشركة SoC واسعة النطاق.

الهدف الرئيسي من مشروعنا هو تطوير تقنية لرسم خرائط لمهام التطبيق في بنية شبكة على رقاقة تعتمد على الاستدلالات الفوقية التي من شأنها تقليل تكلفة الاتصالات. نقترح استخدام خوارزمية جديدة تعتمد على النسر الذهبي.

الكلمات المفتاحية : شبكة ثنائية الأبعاد على رقاقة ، اللاسلكية NoC 2D، شبكة ثلاثية الأبعاد على رقاقة، رسم الخرائط، الاستدلال الفوقي ، تكلفة الاتصالات، تحسين النسر الذهبي .

Remerciement

En premier lieu et avant tout nous remercions Dieu « Allah » le tout-puissant et miséricordieux donné autant de courage, de patience et de volonté pour réaliser ce travail.

Nous exprimons nos sincères remerciements à madame Fatima Boumahdi pour leur aide morale et scientifique, leur conseil, leur gentillesse et leur grande disponibilité ainsi que la confiance qu'ils nous ont toujours inspirée.

Nous tenons à remercier monsieur Maamar Bougherara doctorant à ensk pour son suivi et ses orientations et en particulier pour le temps qu'il a consacré pour développer ce travail.

Nous tenons également à remercier les honorables membres du jury d'avoir accepté de juger et d'évaluer notre travail et d'avoir pris le temps de lire ce mémoire.

Enfin je voudrais remercier toutes les enseignantes qui suivirent dans notre cursus.

Dédicace

Je dédie ce travail, à mes très chers parents dont le sacrifice, la patience, le soutien, l'aide et les encouragements sont le secret de ma réussite. Sans eux je ne serai pas ce que je suis aujourd'hui.

Je dédie ce travail également à mes chers frères Alaa Eddine et Walid et à ma chère sœur Rayane.

À mon cher grand-père Mustafa miséricorde de Dieu.

À mon cher grand-père Amar Dieu vous préserve la santé et longue vie.

À mes chères grandes mères.

À tous les membres de la famille Osmani.

À mon binôme Habiba

À tous mes amis et tous avec ceux qui je partage de bons souvenirs.

Akila.

Dédicace

Je dédie le fruit de ce modeste travail comme un geste de gratitude :

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

Ma très chère mère.

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

Mon très cher père.

À toute ma famille, du plus vieux jusqu'au plus jeune, à mes.

Sœurs Djamila et Fatima Zohra.

À mon binôme Akila et tous les étudiants de Master 2 SIR (2021 /2022) à Université Saad Dahleb Blida.

À tous les professeurs que ce soit du primaire, du

Moyen, du secondaire ou de l'enseignement supérieur.

Habiba.

Content

Introduction générale 1

Plan du mémoire 2

Chapitre 1 : Réseau sur puce 4

1.1 Introduction 4

1.2 Système sur puce 5

1.3 Les interconnexions au sien des SoC 6

1.3.1 Interconnexion point à point 6

1.3.2 Intégration basée sur Bus partagé 7

1.3.3 Intégration par Bus hiérarchique 7

1.3.4 Intégration basée sur le crossbar 8

1.3.5 Intégration basée sur des réseaux à commutation de paquets 9

1.4 Réseau sur puce 10

1.4.1 Composante de réseau sur puce 10

1.4.2 L'architecture en couches 13

1 .4.3 Caractéristiques des réseaux sur puce 15

1.4.3 Réseaux sur puce émergents 23

1.5 Conclusion 27

Chapitre 2 : L'optimisation 28

2.1 Introduction 28

2.2 Définition du problème 28

2.3 Les méthodes de résolution de problèmes d'optimisation 29

2.3.1 Optimisation continue 29
2.3.2 Optimisation combinatoire 30
2.4 Algorithme aigle royal 38
2.5 Conclusion 43
Chapitre 3 : Problème de Mapping 44
3.1 Introduction 44
3.2 Définitions 44
3.2.1 Graphe d'application (GAP) 45
3.2.2 Graphe d'architecture (GAR) 46
3.2.3 Mapping 47
3.2.4 Fonctions objectives 49
3.3 Types de Mapping 50
3.3.1 Mapping statique 50
3.3.2 Mapping dynamique 50
3.4 Problème de Mapping 50
3.5 Algorithmes de Mapping existants 51
3.5.1 Les méthodes de mapping Noc 2-D 51
3.5.2 Les méthodes de mapping Noc 3-D 52
3.5.3 Les méthodes de mapping WNoC 53
3.6 Synthèse entre les différentes méthodes de mapping 53
3.7 Conclusion 56
Chapitre 4 : Solution proposée 57
4.1 Introduction 57
4.2 Formulation du problème 57
4.3 Définition de notre fonction objective 58
4.3.1 Architecture 2D et 3D 58

4.3.2 Architecture WNoC 59
4.4 Représentations des solutions 63
4.5 Représentation de graphe d'application 64
4.6 Le processus de la solution 65
4.6.1 Étape 1{Les entrées du problème} 66
4.6.2 Étape 2 {Traitement de population} 67
4.6.3 Etape 3 {Traitement de Golden Eagle} 69
4.7 Conclusion 72
Chapitre 5 : Tests et résultats 73
5.1 Introduction 73
5.2 Outils d'implémentation utilisés 73
5.2.1 Caractéristiques de machine 73
5.2.2 Langage de programmation Java 74
5.3 Environnement de développement NetBeans 74
5.4 Présentation des benchmarks (graphes utilisés) 75
5.4.1 Benchmark aléatoire 75
5.4.2 Les benchmarks standard 76
5.5 Réalisation des tests 78
5.5.1 Paramétrage 78
5.5.2 Réalisation des tests 79
5.6 Conclusion 106
Conclusion générale 107
Perspectives 108
Référence 109

Liste des figures

Figure 1-1 : Exemple d'une architecture des systèmes sur puce (architecture de Samsung Galaxy S7)(Suovanen, 2020) 5

Figure 1-2 : Évolution des interconnexions sur les SoCs (Tan & Guo, 2009).6

Figure 1-3 : la connexion point à point(DJABALLAH, 2011)..... 6

Figure 1-4: SoC basé sur l'infrastructure de communication en bus partagé (Chariete, 2014). 7

Figure 1-5: Structure d'interconnexion Bus hiérarchique (Delorme, 2007).. 8

Figure 1-6: Structure d'interconnexion Crossbar (BOUGUETTAYA, 2017).8

Figure 1-7: SoC basé sur l'infrastructure de communication en réseau à commutation de paquets (Chariete, 2014). 9

Figure 1-8: La structure du réseau sur puce 4x4 (Chariete, 2014). 11

Figure 1-9: structure d'un router sur puce (Delorme, 2007). 12

Figure 1-10 : Les éléments de Noc et les couches réseau (Tanenbaum & Wetherall, 2010) 13

Figure 1-11 : La structure des différentes topologies (LARIBI, 2010 ; Lemaire et al., 2006 ; MAYACHE, 2018). 17

Figure 1-12: Techniques de commutation dans les réseaux sur puce (Alimi et al., 2021). 18

Figure 1-13: commutation store-and -forward (Tatas et al., 2014). 19

Figure 1-14: commutation Virtual Cut-through (Tatas et al., 2014). 20

Figure 1-15: commutation Wormhole (Tatas et al., 2014). 20

Figure 1-16: Exemple de l'algorithme de routage XY (Sheng Ma, Libo Huang, 2015). 22

Figure 1-17: Architecture de WNOC 10x10 (C. Wang et al., 2011). 24

Figure 1-18: Microarchitecture de routeur Hybride Wired/Wireless (Yin et al., 2012). 25

Figure 1-19: Exemple de la topologie 3D (Gan et al., 2021)..... 26

Figure 2-1: La classification générale des méthodes d'optimisation(Janga Reddy & Nagesh Kumar, 2020). 29

Figure 2-2: Les méthodes de résolution exacte (Janga Reddy & Nagesh Kumar, 2020). 31

Figure 2-3: Taxinomie des méthodes d'optimisation métaheuristiques. 33

Figure 2-4: La structure de l'algorithme génétique.(Okwu & Tartibu, 2020).36

Figure 2-5 : Golden Eagle. (Fennema, 2018). 38

Figure 2-6 : Mouvement en spirale des aigles royaux (Mohammadi-Balani et al., 2021) 40

Figure 3-1: graphe d'application. 45

Figure 3-2: Graphe d'architecture (Xiaodong et al., 2020). 46

Figure 3-3: Graphe d'architecture 3D(Gan et al., 2021). 47

Figure 3-4: Exemple de mapping sur l'architecture 2D NOC..... 48

Figure 3-5: Exemple de mapping sur l'architecture 3D Noc(Gan et al., 2021). 48

Figure 3-6: Exemple de Mapping des routeurs sans fil sur une architecture de WNoC. 49

Figure 3-7: Exemple de Mapping des IPs sur une architecture de WNoC. . 49

Figure 4-1: Exemple de calcul du nombre de sauts dans une topologie Noc 2D 3×3. 59

Figure 4-2: Exemple du calcule de la distance dans le même sous-réseau sur une topologie WiNoc. 60

Figure 4-3 : Exemple de calcul de la distance dans des sous-réseaux différents sur une topologie WiNoc. 63

Figure 4-4: un exemple de la représentation d'une solution. 63

Figure 4-5: Un exemple de la représentation d'une solution. 64

Figure 4-6: Représentation du graphe d'application. 65

Figure 4-7: Processus de la solution. 66

Figure 5-1: Environnement de développement NetBeans. 75

Figure 5-2: Code QR contient les Benchmarks aléatoires utilisés dans les tests 76

Figure 5-3: Benchmark TG10. 76

Figure 5-4 : Benchmark VOPD. 77

Figure 5-5 : Benchmark MWD. 78

Figure 5-6: Comparaison des meilleurs résultats de GEO, GWO-TAB, ABC et FBO dans Noc 2D. 87

Figure 5-7: Comparaison des meilleurs résultats de GEO, GWO-TAB, ABC et FBO dans Noc 3D. 93

Figure 5-8: Comparaison des meilleurs résultats pour les trois topologies.101

Figure 5-9 : comparaison des résultats utilisant le Benchmark VOPD dans 2D NOC. 103

Figure 5-10: comparaison des résultats utilisant le Benchmark MWD dans 2D NOC. 104

Figure 5-11: comparaison des meilleurs résultats utilisant le Benchmark VOPD dans NOC 3D..... 105

Figure 5-12 : comparaison des meilleurs résultats utilisant le Benchmark MWD dans NOC 3D..... 106

Liste des tableaux

Tableau 1-1: La structure des différentes topologies (LARIBI, 2010). 9

Tableau 3-1 : une comparaison entre les algorithmes utilisés dans le mapping. 54

Tableau 5-1 : Résultats obtenus en changeant la taille de population initiale dans Noc 2D. 79

Tableau 5-2 : Résultats obtenus en changeant la taille d'itération dans Noc 2D. 84

Tableau 5-3 : Comparaison GEO avec GWO-TAB, BFO, ABC et sur Noc 2D. 86

Tableau 5-4 : Résultats obtenus en changeant la taille de population initiale dans Noc 3D. 88

Tableau 5-5 : Résultats obtenus en changeant la taille d'itération dans Noc 3D. 90

Tableau 5-6 : Comparaison GEO avec GWO-TAB, BFO, ABC et sur Noc 3D. 92

Tableau 5-7 : Résultats obtenus en changeant la taille de population initiale dans WNoC 2D. 94

Tableau 5-8 : Résultats obtenus en changeant la taille d'itération dans WNoC 2D. 98

Tableau 5-9 : Meilleurs résultats pour les trois topologies. 100

Tableau 5-10 : Les dimensions utilisé dans les tests. 102

Tableau 5-11 : Comparaison entre GEO avec d'autres résultats s sur NoC 2D. 102

Tableau 5-12 : Comparaison des meilleurs résultats utilisant le Benchmark VOPD, MWD dans 3D NOC..... 104

Liste d'abréviations

Abréviations	*Significations*
2D	Two Dimensional
3D	Three Dimensional
ABC	Artificial Bee Colony
ACO	Ant Colony Optimization
AG	Algorithme Génétique
BFO	Bacterial Foraging Optimization
CPU	Central Processing Unit
GEO	Golden Eagle Optimisation
GOW-TAB	Tabo Gray Wolf Optimisation
IP	Intellectual Property
GA	Genetic Algorithm
NI	Network interfaces
Noc	Network On Chip
PSO	Particule Swarm Optimization
RAM	Random Access Memory
ROM	Read Only Memory
SAF	Store-and- Forward
SOC	System On Chip
TG	Task Graph
TGFF	Task Graph For Free
TSV	Through Silicon Via
VCT	Virtual Cut-through
WR	Wireless Router
WNoC	Wireless Network on Chip

Introduction générale

Grâce aux évolutions technologiques, plus d'un billion de transistors peuvent désormais être intégrés dans un seul substrat de silicium (SAMUEL, 2021). Cette intégration a incité les concepteurs de circuits intégrés à incorporer une variété de fonctions, de capacités et de topologies et de circuits supplémentaires. En raison de la rapidité avec laquelle la technique évolue, il est désormais possible de combiner un système complet sur un seul substrat appelé système sur puce (Soc).

Les systèmes sur puces ont joué un rôle de plus en plus essentiel dans notre vie quotidienne afin d'optimiser les applications dédiées. On effet on trouve système sur puces dans nos cartes biométriques, carte chiffra, carte bancaire… Alors qu'il couvre une variété de domaines notamment l'avionique, les transports, les systèmes médicaux et les télécommunications et des appareils commerciaux à usage répandu par exemple : les smartphones, télévision, les avions, satellites, appareils photo, imprimantes …ceci est dû au nombre croissant de transistors qui peuvent être intégrés sur une seule puce qui continue à doubler tous les 18 mois selon la loi de Moore qui a été formulée en 1965 (Moore, 2005).

Cependant, la conception d'une telle application créée un problème au niveau des communications, car un grand volume de données doit être traité. Il n'est pas possible d'utiliser les topologies de communications traditionnelles, car elles ne sont pas adaptées pour supporter l'interconnexion de plusieurs ressources. Il faut plutôt utiliser le concept de topologie de bus avancé tel que le réseau sur puce (Network on chip) (GUÉRARD, 2011). La présence d'un fonctionnel sur l'architecture réseau sur puce est essentielle. En comparaison avec d'autres systèmes, cette structure offre une large bande passante, une variété de chemins, le parallélisme des communications, la flexibilité, l'évolutivité et la réutilisation IP simple. D'autre part, l'augmentation du nombre de cœurs et de transistors dans les modèles 2D s'est traduite par l'augmentation de la taille des circuits et l'extension des câbles de communication.

Technologie de fabrication de circuits intégrés 3D développée pour réduire les délais de propagation sur les fils longs et globaux dans une seule puce. Le NoC 3D est produit en plaçant des couches de réseau 2D verticalement sur la puce. Mais lorsque l'échelle du réseau devient plus grande, le NoC 3D présent des limitations de performances telles qu'une latence et une consommation d'énergie élève.

C'est dans ce contexte que le réseau sans fil sur puce (WNoC) s'est développé. Inspiré du réseau traditionnel sur puce, ce paradigme d'interconnexion fournit une structure de communication évolutive et flexible et propose des solutions efficaces aux problèmes d'interconnexion difficiles des réseaux sur puce.

La phase de mapping (placement physique) est une phase centrale dans la conception de Nocs. Donc le résultat effectue directement les performances du système. Cela implique de placer chaque élément de l'application sur une architecture de réseau sur puce afin de minimiser le coût de communications.

L'objectif de ce travail est de créer une application qui réduit le coût de la communication en utilisant un nouvel algorithme métaheuristique appelé aigle royal (Golden eagle).

Plan du mémoire

Le travail présenté dans ce mémoire a été divisé en 5 chapitres comme suit

Le premier chapitre illustre les concepts de base des réseaux sur puce, ses composantes et ses caractéristiques ainsi que les différents Nocs émergents.

Dans le second chapitre consisté à présenter les techniques d'optimisation, ses différents concepts et ses méthodes sont principalement l'optimisation GEO (Golden Eagle Optimisation).

Dans le troisième chapitre explique le problème du mapping, ses types et aussi les différentes algorithmes méta heuristique utiliser pour résoudre le problème de mapping.

Le quatrième chapitre présente comment adapter l'algorithme de l'aigle royal (GEO) à notre problématique.

Dans le dernier chapitre (cinquième), nous avons présenté les différents résultats obtenus et les comparons avec d'autres études.

Chapitre 1 : Réseau sur puce

1.1 Introduction

Un système sur puce est constitué en particulier par des cœurs de propriété intellectuelle (IP), un IP est en fait un circuit prédéfini permettant de définir une application bien ciblée par exemple : des contrôleurs de mémoire, les processeurs ou des périphériques tels que des contrôleurs de bus PCI. Les cœurs de propriété intellectuelle sont les blocs de base pour la construction de divers systèmes sur puces pour la mise en œuvre des applications de systèmes embarqués plus grands et plus complexes(Chatmen, 2017).

Alors que le nombre d'IPs sur une même surface de silicium augmente, le principal enjeu lors de la conception de systèmes sur puce est d'assurer les interconnexions entre les différents éléments qui communiquent entre eux, car les interconnexions traditionnelles comme point à point, bus partage et hiérarchique donnée des performances réduit en matière de bonde passante et la consommation lors de l'application d'un degré élevé de communication.

Pour surmonter cette limitation, une technique de conception se concentre sur les réseaux de communication sur puce, appelés réseaux sur puce (Network on chip : NOC). À structure de bus traditionnelle est remplacée par une architecture a inspiré par le réseau d'informatique. Le réseau sur puce est considéré comme la solution la plus appropriée pour la communication entre les différents composants d'une puce.

Dans ce chapitre, nous allons présenter l'évolution des différents types de communications traditionnels utilisés dans les systèmes sur puce, jusqu'à l'apparition des réseaux sur puces (Network on chip : NOC) leur composant, les architectures, et aussi les réseaux sur puces émergentes.

1.2 Système sur puce

Est un système complet embarqué sur une seule puce (un circuit intégré) qui combine de différents éléments d'un système informatique en une seule puce de silicium pouvant contenir RAM (Random Access Memory), mémoire morte (Read Only Memory : ROM), des unités centrales de traitement (CPU), des ports d'entrée et de sortie, un contrôleur d'E/S, Carte graphique (GPU) , processeur de signal numérique (DSP)et des structures d'interconnexion, les composants assemblés dans un Soc varient en fonction de l'application réalisée (ATAT, 2007).

Les récents SoCs des principaux fabricants de puces comme Intel, Qualcomm, Apple, et Texas Instruments sont beaucoup plus complexes que ceux montrés dans figure 1.1. Comme il peut être vu, les SoCs contiennent la plupart des blocs fonctionnels essentiels du système pouvoir fonctionner comme le produit prévu(Chakravarthi, 2020).

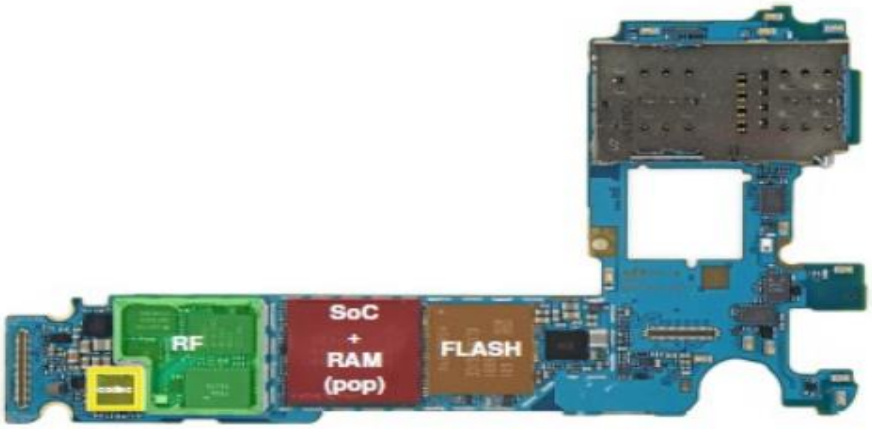


Figure 1-1 : Exemple d'une architecture des systèmes sur puce (architecture de Samsung Galaxy S7)(Suovanen, 2020) .

Les principaux avantages du système sur puce sont sa faible consommation d'énergie, son coût inférieur et sa fiabilité supérieure par rapport aux systèmes multi-puce qu'il remplace (Ngan, 2011). Mais la transition vers la technologie Soc fait face à de nombreux défis. Tout

d'abord, l'évolutivité du système. Réduire un grand système informatique à la taille d'une puce de silicium est en effet une tâche ardue.

1.3 Les interconnexions au sien des SoC

Il existe différents types de systèmes d'interconnexion utilisables dans un Soc pour assurer les communications entre les différents blocs fonctionnels d'un SoC dont le point à point, le bus partagé ou hiérarchique et le réseau sur puce.

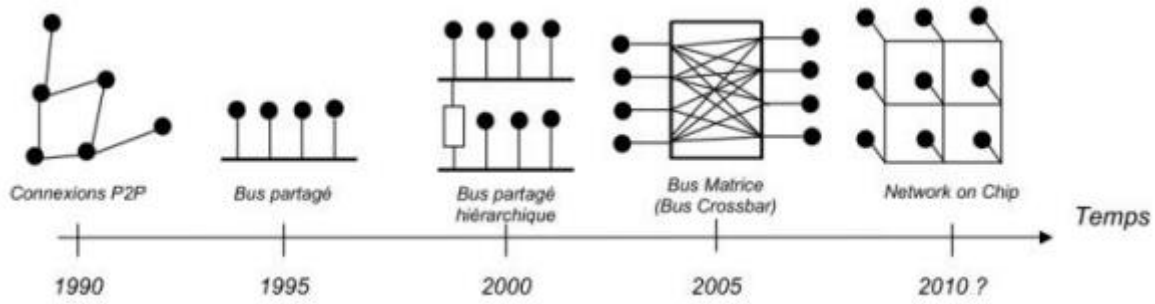


Figure 1-2 : Évolution des interconnexions sur les SoCs (Tan & Guo, 2009).

1.3.1 Interconnexion point à point

Ce type d'interconnexion est le plus simple. Il permet de relier chaque deux IPs directement sans aucun protocole de gestion de communication (BOUGUETTAYA, 2017) . L'avantage de cette connexion est qu'elle a une grande capacité de parallélisation de connexions multiples entre les ressources, ainsi qu'une excellente bande passante et des taux de transfert de données très élevé. Mais il nécessite un nombre important de branches pour chaque IP lorsque le système a une grande taille (SALEEM, 2015), alors ce type d'interconnexions n'offre pas de flexibilité et des performances pour un système avec un grand nombre de composants.

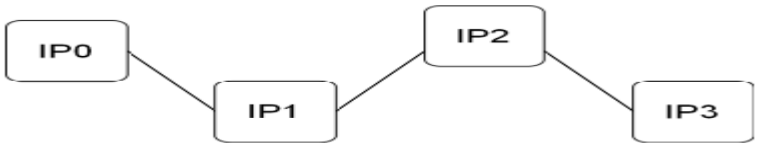


Figure 1-3 : la connexion point à point(DJABALLAH, 2011).

1.3.2 Intégration basée sur Bus partagé

L'interconnexion par bus est la méthode de communication la plus large utilisée dans l'industrie. Dans les systèmes à base de bus partagé, un seul cœur à la fois peut utiliser le médium (bus), un protocole d'arbitrage gère la communication et résout les conflits d'accès (stratégie par priorité)(Séverine Riso, 2005).

C'est le moyen de communication le plus flexible et le plus simple, avec la meilleure réutilisabilité, mais aucune extensibilité, faible et toute forme de parallélisme sont exclues.

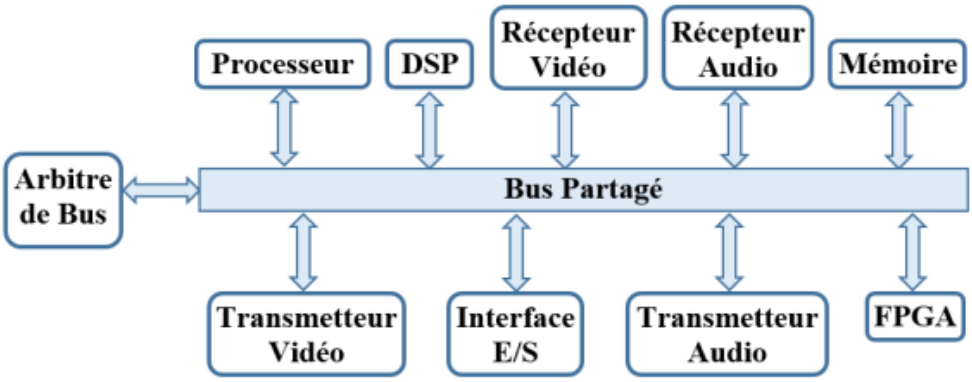


Figure 1-4: SoC basé sur l'infrastructure de communication en bus partagé (Chariete, 2014).

1.3.3 Intégration par Bus hiérarchique

Cette architecture résoudre le problème concerne le parallélisme dans bus partage, car il se compose de plusieurs bus partagés connectés les uns aux autres par des ponts (bridge) (BOUGUETTAYA, 2017). De sorte que les opérations puissent être effectuées en parallèle sur différents segments de bus et avoir une consommation d'énergie minimise et plus petite latence.

Néanmoins, les communications à travers le pont deviennent un goulot d'étranglement, ce qui signifie que la latence augmente lorsque les deux bus veulent communiquer entre eux. Cette architecture ne peut pas non plus être utilisée avec les futurs systèmes.

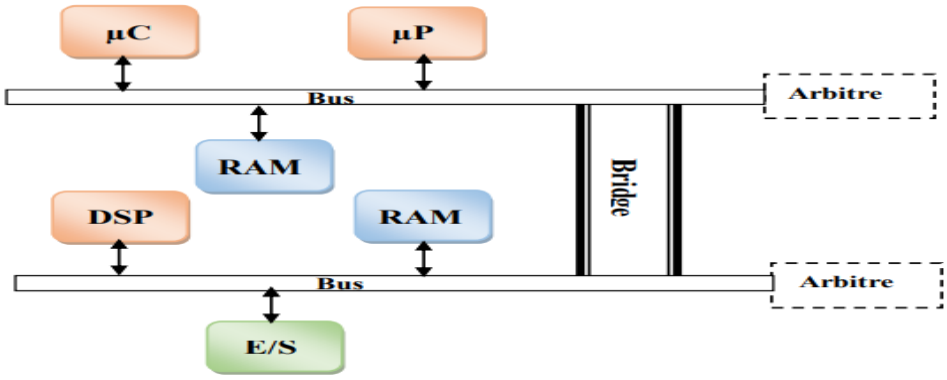


Figure 1-5: Structure d'interconnexion Bus hiérarchique (Delorme, 2007).

1.3.4 Intégration basée sur le crossbar

Ce type d'interconnexion est une matrice de multiplexeur qui permet à tous les composants sur la même surface de communiquer en parallèle, les IPs d'entrée sont connectées directement aux IPs de sortie sans un intermédiaire.

Le crossbar offre une large bande passante. Cependant, le nombre de connexions physiques nécessaires à la mise en place d'un crossbar complet limite grandement les domaines où il peut être appliqué (Chariete, 2014) .

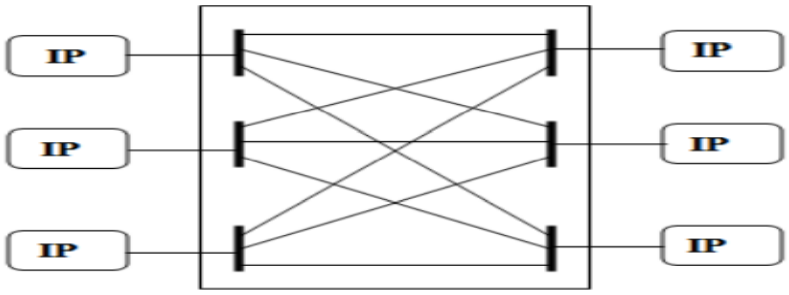


***Figure 1-6: Structure d'interconnexion Crossbar* (BOUGUETTAYA, 2017).**

1.3.5 Intégration basée sur des réseaux à commutation de paquets

En raison d'un certain nombre d'inconvénients d'interconnexions point à point, du bus partagé et du crossbar dont la faible évolutivité, non-adaptabilité aux nouvelles applications et là très peu ou pas de réutilisation. Cette structure est inspirée par le réseau informatique et adaptée pour les systèmes sur puce (Chariete, 2014).

Un réseau sur puce comprend un certain nombre de dispositifs hétérogènes interconnectés (processeurs à usage général ou spécifique, mémoires embarquées, composants spécifiques à l'application…) connectés par un réseau de communication à commutation de paquets sur une seule puce(KAMECHE, 2013) .Réseau est une interconnexion évolutive et une tolérance aux pannes sans dégrader le débit de données avec un grand potentiel pour gérer la complexité croissante des SoC multicœurs actuels et futurs.

Figure 1-7: SoC basé sur l'infrastructure de communication en réseau à commutation de paquets (Chariete, 2014).

❖ Comparaison entre les différents types d'interconnexion de SoC

Tableau 1-1: La structure des différentes topologies (LARIBI, 2010).

Connexion	*Parallélisme*	*Consummation*	*Réutilisation*	*Bande Passante*

Point à point	Très bon	Bon	Très mauvais	Très bon
Bus partagé	Très mauvais	Très mauvais	Très bon	Très mauvais
Bus Hiérarchique	Bon	Mauvais	Mauvais	Bon
Noc	Très Bon	Très Bon	Très Bon	Très Bon

1.4 Réseau sur puce

Le réseau est utilisé pour communiquer entre les IP fondamentaux du système sur une seule puce. L'aspect essentiel du NoC est qu'il utilise la technologie de réseau pour établir l'échange de données au sein de la puce, le NoC se compose d'un ou plusieurs cœurs IP et a pour rôle de transporter les données sous forme de paquets entre plusieurs nœuds (IP, routeur et interface réseau). Les routeurs sont placés selon un schéma régulier pour diriger chaque paquet vers sa destination prévue (c'est-à-dire sa route). Pour transférer physiquement les messages, chaque routeur possède de nombreux ports de sortie et d'entrée reliés à une connexion (NEHNOUH, 2019).

Selon la structure des IP, un NoC peut être classé en deux catégories (Wei,song; Guangda, 2022) : NoCs homogènes et NoCs hétérogènes. Tous Les IP dans un NoC homogène utilisent la même structure interne ; ils ont donc la même taille et la puce est symétrique, alors que les IP dans un NoC hétérogène peuvent utiliser des structures différentes. La taille d'un IP est donc variable et le plancher de copeaux est à peine symétrique.

1.4.1 Composante de réseau sur puce

Un réseau sur puce typique est composé de quatre éléments principaux (Chariete, 2014): des routeurs, des liens de communication, les IPs (Intellectuel Property) et d'interfaces réseaux.

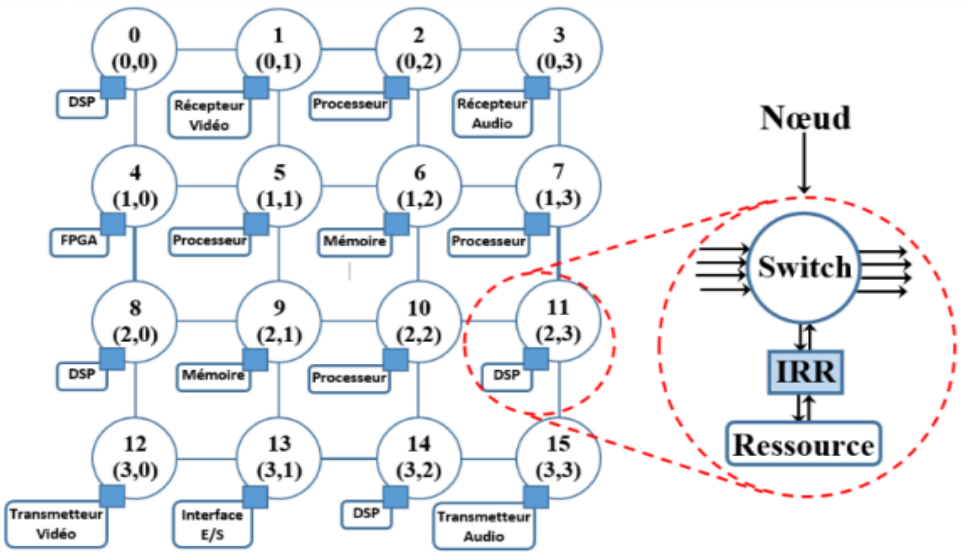


Figure 1-8: La structure du réseau sur puce 4x4 (Chariete, 2014).

1.4.1.1 Routeur (commutateur)

Routeur est élément clé dans la conception et la communication du réseau sur puce, la fonction d'un routeur est d'acheminer les paquets entrants d'une ressource source vers une ressource destination selon un protocole de routage choisi.

Chaque routeur comprend :

- Plusieurs ports d'entrée / sortie pour la communication.
- Crossbar (une matrice de commutation) pour connecter les ports d'entrée à des ports de sortie.
- Des files d'attente pour stocker les données qui ne peuvent pas être transmises directement.
- Une unité de routage et d'arbitrage sert d'arbitre, résolvant les conflits entre les paquets.

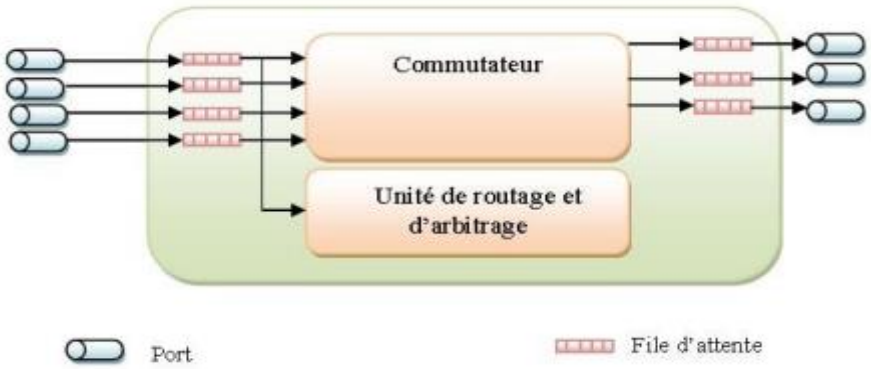


Figure 1-9: structure d'un router sur puce (Delorme, 2007).

1.4.1.2 Les ressources IPs (Intellectuel Property)

Les ressources sont les différentes unités de traitement et de stockage d'une puce, qui peuvent être des cartes graphiques, des processeurs à usage général (processeurs à usage général pouvant effectuer tout type de calcul) ou des processeurs à usage spécial (processeurs spécialisés qui effectuent certaines tâches spécifiques), et différents types de mémoire volatile ou non volatile, contrôleur E/S. Les ressources devraient être mises en œuvre en utilisant la même technologie que celle du commutateur de réseau (Luo-Feng et al., 2008).

1.4.1.3 Interface réseau

Cet élément est un intermédiaire entre une ressource IP responsable des traitements et le routeur des réseaux responsables de la communication, le rôle de l'interface réseau dans le réseau sur puce est le même que celui de la carte réseau dans le réseau informatique (Jantsch& Tenhunen, 2003).

L'interface réseau est composée de deux parties, une partie indépendante des ressources où elle est liée au router, la deuxième est la partie dépendante des ressources consistées à connecter la partie frontale des ressources de traitement, en tant qu'adaptateur entre le protocole de l'unité de traitement et le protocole de la structure de communication (NEHNOUH, 2019).

1.4.1.4 Les liens

Les liens sont les connexions logiques qui permet de transférer des données entre deux composants d'un réseau sur puce (entre deux routeurs, interface réseau et routeur, une ressource IPs et interface réseau), ils fournir la bande passante pour les communications entre une source et une destination. Les liens peuvent être unidirectionnels ou bidirectionnels comme ils peuvent se composer de plusieurs canaux physiques ou logiques afin de réduire le taux de congestion lors des communications (Leroy, 2007) .

1.4.2 L'architecture en couches

Comme dans les réseaux informatiques de communications, le NoC utilise la communication en couches. Dans ce type de systèmes, le protocole de communication spécifie les principes de transmission, c'est un ensemble de règles et de méthodes qui sont nécessaires pour transférer des données de l'émetteur au récepteur (Chariete, 2014).

Le modèle OSI décrit le protocole de communication d'un réseau sous forme de sept couches (physique, liaison de donnée, réseau, transport, session, présentation, application) (Zimmermann, 1980), cette représentation OSI n'est pas adaptée à la description d'un NoC, car celui-ci n'intègre pas toutes ces couches, seules les quatre premières couches sont implantées, les couches supérieures sont en général plus spécifiques à l'application et prises en charge par les ressources (IP, CPU) connectées au réseau(DJABALLAH, 2011; Lemaire et al., 2006).

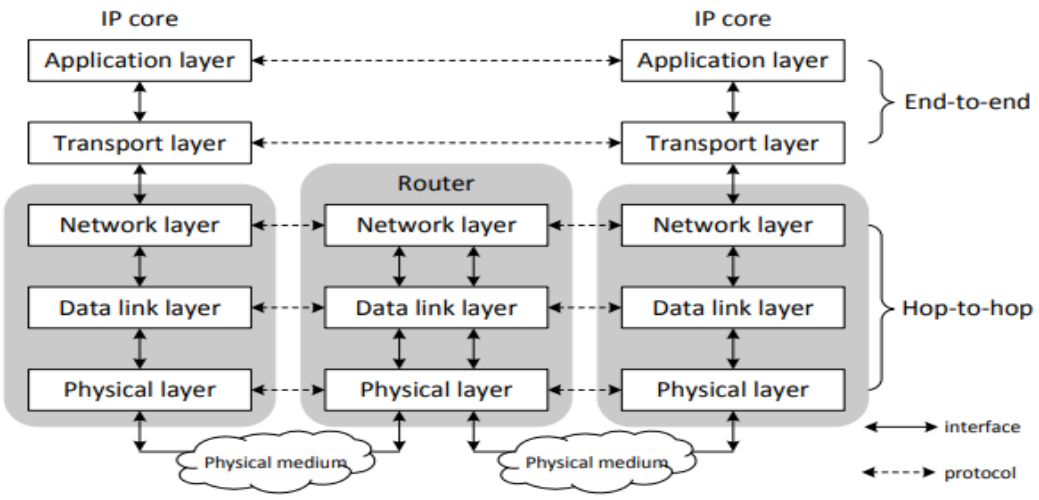


Figure 1-10 : Les éléments de Noc et les couches réseau (Tanenbaum & Wetherall, 2010) .

- **Couche application**

C’est la couche qui contient les ressources (IPs) ou chaque ressource (IP) peut effectuer plusieurs tâches différentes (par exemple, un microprocesseur peut exécuter plusieurs processus simultanément)(Tatas et al., 2014).

- **Couche transport**

La couche de transport est l’entité responsable de la bonne transmission des messages(ALLARDBERNIER, 2011). Elle obtient les messages à transmettre de la couche logicielle, les fragmente en paquets plus petits si nécessaire et les transmet à la couche réseau. Elle assure le contrôle de flux de bout en bout afin d’éviter que des données ne soient envoyées vers une destination qui n’a plus de place disponible pour recevoir. De plus elle gère l’adaptation de protocole entre NOC et l’IP via interface réseau.

- **Couche réseau**

La couche réseau est responsable de la livraison des paquets depuis la ressource source à travers le réseau de commutateurs jusqu'à la ressource de destination, elle donne la possibilité d’utiliser un algorithme de routage dynamique ainsi que la mise en mémoire tampon des paquets dans les commutateurs(Jantsch& Tenhunen, 2003).

- **Couche liaison de donnée**

Cette couche regroupe les flux de bits en petits blocs pour assurer une transmission fiable de saut en saut sur le support physique médium. Elle définit les méthodes de contrôle des flux, qui décident de l'allocation des ressources du réseau (Dally, Willliam james; Towlrd, 2004).

- **Couche physique**

Définis la structure physique et les protocoles nécessaires (type d'interconnexion et bande passante des trames de données) à l'établissement de la communication au niveau de la liaison entre les divers routeurs.

1 .4.3 Caractéristiques des réseaux sur puce

Chaque réseau sur puce est caractérisé par une topologie, une technique de commutation, un algorithme de routage et un contrôle de flux, qualité de service (Chatmen, 2016).

1.4.3.1 Topologie

La topologie des réseaux sur puce est très similaire à des réseaux informatiques classiques, un réseau dans lequel chaque routeur est relié à un processeur (ressource IP), cette paire appelle un nœud (Érika et al., 2011). La topologie montre comment les nœuds sont connectés les uns avec les autres via des liens de communication. La topologie peut être "logique" et/ou "physique". La topologie logique illustre comment le flux de données entre les nœuds du réseau alors que la topologie physique indique le placement des différents nœuds sur la surface de la puce(Ouled-khaoua & Terranti, 2020).

D'un point de vue topologique, un réseau peut être direct ou indirect. Dans un réseau direct, chaque nœud dispose de deux fonctions. Il produit ou consomme des paquets et il agit également comme un interrupteur. Dans un réseau indirect, le nœud n'a qu'une seule des deux fonctions(KAMECHE, 2013).

1.4.3.1.1 Topologie directe

Un nœud a des liens directs vers un sous-ensemble d'autres nœuds du système, appelés nœuds voisins. Les topologies directes ont une plus grande disponibilité de bande passante de communication, mais elles nécessitent une augmentation du nombre de nœuds dans le système(Tatas et al., 2014). Quelques exemples des topologies directes :

- **2D maillée :** est considérée comme la topologie de réseau sur puce la plus populaire en raison de sa mise en œuvre simple et de sa structure régulière (Delorme, 2007). Il est facile à implanter sur la technologie silicium, car tous les maillons ont la même longueur(Tatas et al., 2014). Le réseau maillé présente également une grande diversité de chemins, ce qui contribue à réduire la dissonance dans le NoC. La surface de maillage (nombre de routeurs et d'interfaces réseau) augmente linéairement avec le nombre des IPs. Le maillage a une distance moyenne relativement grande entre les interfaces réseau,

ce qui affecte négativement la consommation d'énergie. De plus, leur bande passante limitée peut dégrader les performances en cas de charge élevée (Luca & Giovanni, 2006).

- **3D maillée :** le développement de la technologie 3D dans les circuits intégrés a donné naissance aux réseaux 3D sur puce. Selon cela, le réseau sur puce traditionnelle (2D) peut être étendu à trois dimensions en ajoutant des routeurs 3D, fournissant une communication verticale via l'interconnexion Throught Silocon via (TSV), 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.
- **En anneau :** chaque routeur dans une topologie en anneau est connecté à deux routeurs adjacents et à une ressource IP locale, formant une boucle comme la figure 1.11[c]. Parmi les avantages de cette topologie, elle est facile à mettre en œuvre du fait de sa faible connectivité (TOUATI, 2012). D'autre part, pour les grands réseaux. Mais, cette topologie souffre d'un inconvénient majeur ; car pour un réseau de grande taille, le message doit traverser un nombre important de routeurs, engendrant inévitablement une limite de bande passante(Séverine, 2005).
- **Torus (ou trou) :** la topologie du trou très similaire au maillage, mais en un sens complète les liens du réseau en connectant le dernier nœud au premier sur chaque axe, comme il est présenté à la figure 1.11[b] .Elle offre ainsi une bande passante supérieure à celle de la 2D maillée, mais elle est plus complexe à implémenter et très difficile à tester (Delorme, 2007).

1.4.3.1.2 Topologie indirecte

Dans les topologies indirectes, pas tous les routeurs ne sont connectés aux unités de traitement comme dans le modèle direct, seuls les routeurs des nœuds terminaux qui sont connectés à une ressource IP. Les nœuds terminaux sont connectés via un ou plusieurs étages intermédiaires de nœuds de commutation. Seuls les nœuds terminaux sont des sources et des destinations de trafic, les nœuds intermédiaires commutent simplement le trafic vers et depuis les nœuds terminaux(Érika et al., 2011; Jerger et al., 2017).

- **En arbre :** Cette topologie est sous forme d'arborescence, où les nœuds sont des routeurs/commutateurs, et les feuilles sont des ressources de calcul que le routeur enfant est connecté à son routeur parent comme nous présente dans la figure 1.11 [d], l'avantage

de cette topologie est que le réseau est évolutif de manière récursive et facile à partitionner, mais des niveaux d'arborescence plus élevés peuvent provoquer des goulots d'étranglement. Où des niveaux plus élevés sont utilisés pour le routage entre les neurones.

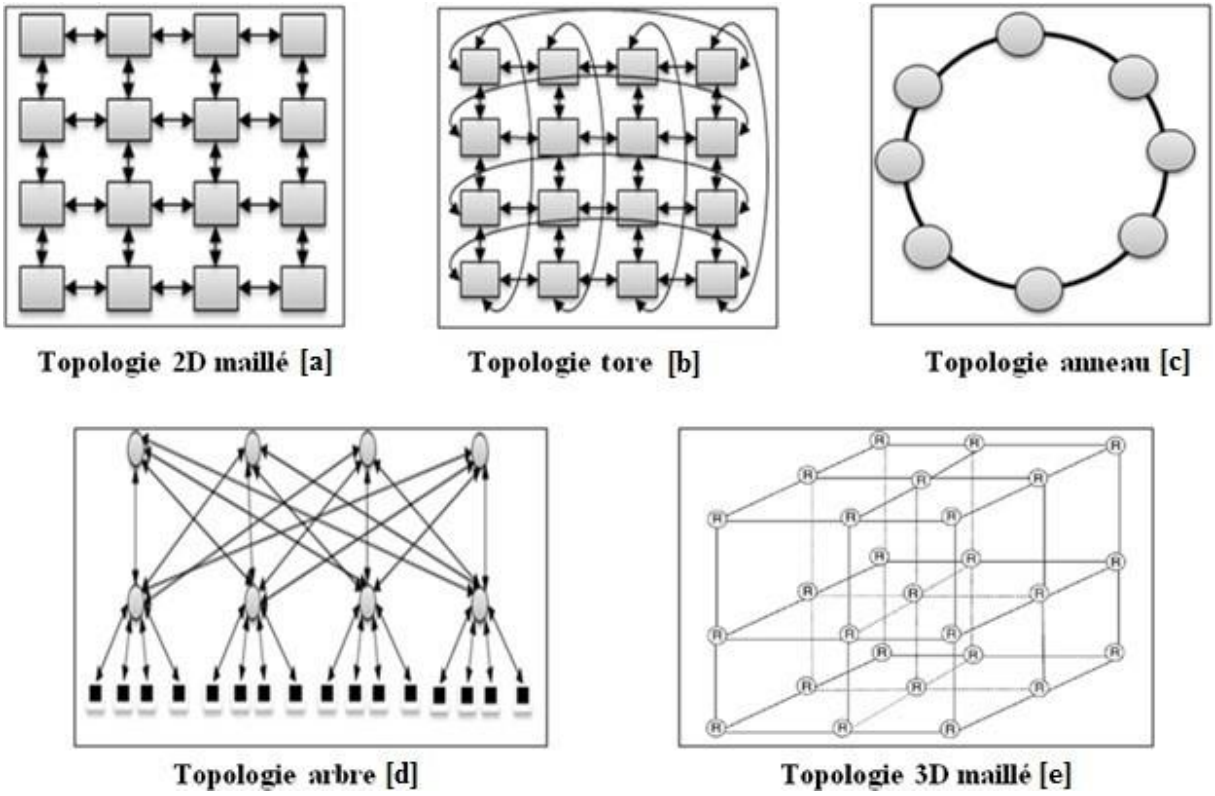


Figure 1-11 : La structure des différentes topologies (LARIBI, 2010 ; Lemaire et al., 2006 ; MAYACHE, 2018).

1.4.3.2 Techniques de commutation

Les techniques de commutation sont utilisées pour définir la façon dont les données sont transférées d'une ressource source à une ressource de destination dans le réseau sur puce, sélectionné par un algorithme de routage (Tatas et al., 2014). Il existe deux types de commutation de données, la commutation de circuits et la commutation de paquets.

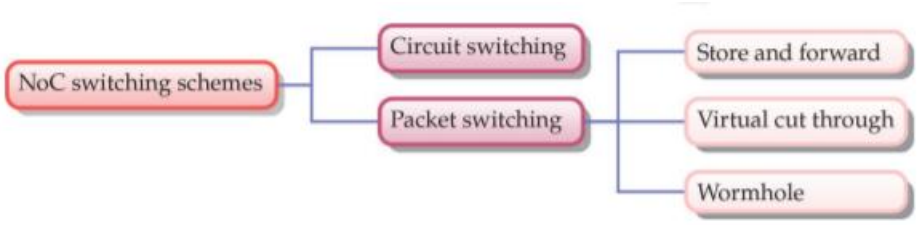


***Figure 1-12: Techniques de commutation dans les réseaux sur puce* (Alimi et al., 2021).**

1.4.3.2.1 Commutation de circuits

La commutation de circuits est basée sur l'établissement d'un chemin physique réservé (une série de liens et de routeurs) entre la source et la destination pendant la communication, une fois la transmission terminée, le chemin qui a été réservé avant de commencer à envoyer des données doit être déconnecté (Ben Abdallah, 2017). L'avantage de cette technique est d'offre une bande passante élevée et une latence limitée, est approprié lorsque les données qui sont envoyées très importants ou lorsque le modèle de communication entre les expéditeurs et les destinataires est relativement statique. Mais une fois qu'une connexion est établie, les routeurs et les liens qui composent le chemin ne peuvent plus être utilisés par d'autres communications (TOUATI, 2012). Cette technique de commutation est inefficace, car elle crée une congestion excessive, qui à son tour affecte la bande passante du réseau et entraîne des retards de communication excessive.

1.4.3.2.2 Commutation des paquets

Cette technique de commutation est plus courante et largement utilisée et fonctionne en divisant un message en un ensemble de paquets de données indépendantes qui sont transportées individuellement sur le réseau(Alimi et al., 2021). Les routeurs de commutation de paquets envoient des données dans des paquets appelés flits (unité de contrôle de flux). Sur la base des informations de routage contenues dans le paquet, le paquet est livré à sa destination, et les paquets qui suivent des chemins indépendants sont transmis sans réserver le chemin entier (SALEEM, 2015). La commutation de paquets permet d'obtenir les meilleures performances réseaux possibles, car toutes les communications partagent des ressources et la transmission d'un paquet donné ne doit pas bloquer les autres communications sur le réseau. Il existe trois modes bases sur la commutation des paquets :

- **Store-and- Forward (SAF) :** est la forme la plus simple de commutation de paquets (Alimi et al., 2021). Un paquet est envoyé d'un routeur à un autre uniquement si le routeur récepteur dispose d'un espace tampon pour l'ensemble du paquet (Isabirye et al., 2012). Chaque nœud attend que le paquet entier soit entièrement reçu dans le tampon du port d'entrée avant de transmettre une partie du paquet au nœud suivant, cette approche nécessite de l'espace tampon pour un paquet complet.

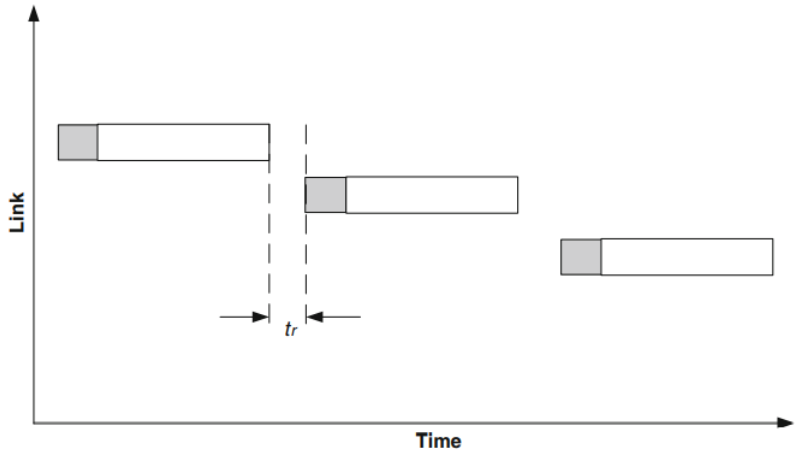


Figure 1-13: commutation store-and -forward (Tatas et al., 2014).

- **Virtual Cut-through :** la technique de commutation VCT a été introduite pour réduire la latence du modèle SAF, cette technique est basée sur la transmission d'un paquet dès que le prochain routeur garantit qu'il peut stocker le paquet dans sa totalité. En cas d'absence de garantie, le routeur doit être en mesure de stocker l'intégralité du paquet. Mais cette approche nécessite de l'espace tampon pour un paquet la même que pour le mode de commutation Store and Forward (Tatas et al., 2014).

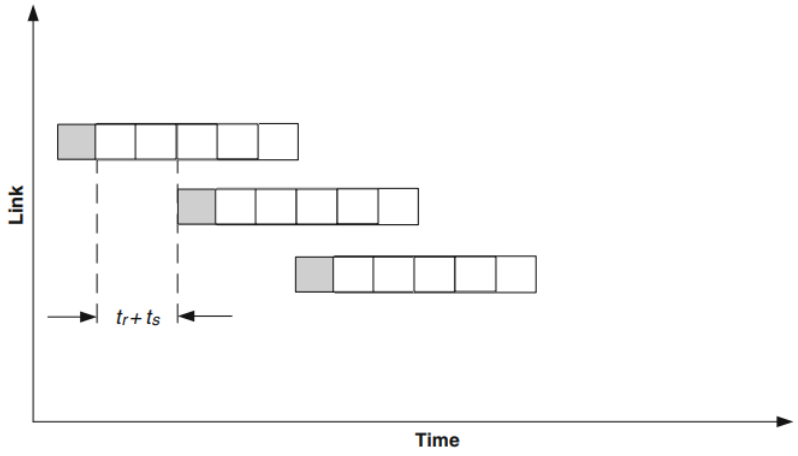


Figure 1-14: commutation Virtual Cut-through (Tatas et al., 2014).

- **Wormhole :** dans cette méthode, chaque paquet est divisé en flits (plus petite unité constituante d'un paquet),le premier flit d'en-tête possède toutes les informations nécessaires au routage (SALEEM, 2015). Celui- ci est transmis de routeur en routeur, tant qu'il y a assez d'espace pour le stocker. Le passage de l'en-tête dans un routeur réserve ce dernier pour la réception et la transmission du reste des flits qui compose le paquet. Ce mode de commutation avoir une meilleure performance en termes de latence et ne doit pas stocker un paquet complet comme les deux modes qui précèdent (Tatas et al., 2014).

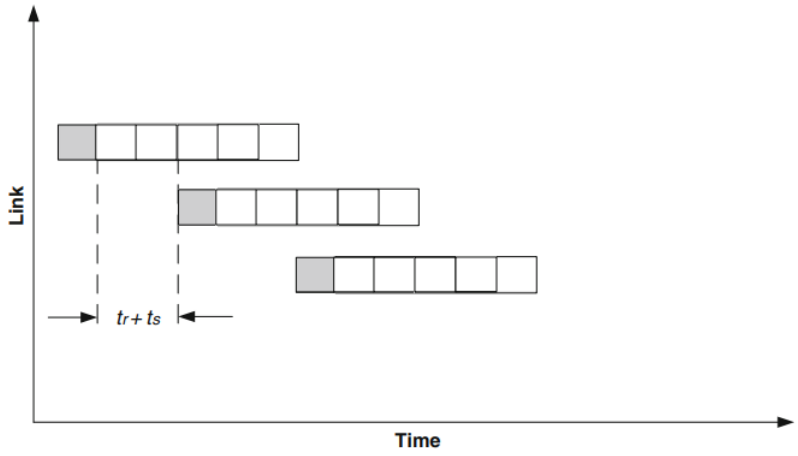


Figure 1-15: commutation Wormhole (Tatas et al., 2014).

1.4.3.3 Algorithme de Routage

L'algorithme est le processus qui permet de sélectionner un port de sortie pour transférer les paquets arrivant à un port d'entrée du routeur. Ce port est choisi en fonction des informations de routage disponibles dans l'en-tête du paquet. Le routage joue un rôle important dans l'architecture de communication, donc un meilleur algorithme de routage offre les meilleures performances. Par conséquent, l'algorithme de routage est soigneusement conçu par le concepteur. Il existe plusieurs algorithmes de routage possibles qui peuvent être utilisés dans un NoC, chacun d'entre eux aboutissant à un compromis diffèrent entre performances et coûts (Érika et al., 2011).

Il existe plusieurs algorithmes de routage qui utilisent dans la conception des réseaux sur puce. Le router peut sélectionner un chemine par le routage déterministe (statique) ou adaptatif (dynamique).

1.4.3.3.1 Routage adaptif

La route empruntée par un message n'est pas connue a priori, mais est déterminée au fur et à mesure qu'il parcourt le réseau. Les décisions de routage sont basées sur l'état actuel du réseau, en tenant compte de facteurs tels que la disponibilité du système et les conditions de charge de la liaison (Kundu & Chattopadhyay, 2014). Par conséquent, à mesure que les besoins des applications et les conditions de trafic changent, le chemin entre la source et la destination peut changer en conséquence. Dans le routage dynamique, le trafic peut être réparti plus efficacement sur différents routeurs. De plus, il peut profiter de chemins alternatifs en cas de congestion du réseau sur certaines liaisons NoC. Ils sont facilement adaptables à différentes topologies (Isabirye et al., 2012). Mais, cette technique de routage est compliquée à mettre en place et peut créer des conflits. En fait, les paquets peuvent boucler sur le réseau tout en consommant des ressources sans trouver leur destination, ce qui entraîne une incertitude sur les temps de transit(Isabirye et al., 2012).

1.4.3.3.1 Routage déterministe

Dans le routage déterministe, un paquet toujours utilise le même chemin entre les nœuds sources et destination, le chemin déterminé par les adresses de l'émetteur et du récepteur(Isabirye et al., 2012) . Il est facile à réaliser en termes de logique et d'interaction entre les routeurs. Le routage statique est plus adapté lorsque les échanges de données sont prévisibles et stables.

Le routage déterministe présent plusieurs algorithmes par exemple :

- **Algorithme de routage XY :** est appliqué dans les topologies 2D Mesh (Chatmen, 2016). Le paquet de données est toujours acheminé le long de la direction "X" en premier, jusqu'à ce qu'il atteigne le nœud avec la même coordonnée "X" que le nœud de destination. Ensuite, il est acheminé dans la direction des « Y » jusqu'à atteindre le nœud de destination. Le routage XY ne parvient pas à transférer les paquets si la liaison et/ou le routeur échoue.

L'algorithme XY est un algorithme déterministe et minimal, qui évite les interblocages puisqu'il est conçu par l'élimination de tous les tours qui peuvent former un cycle.

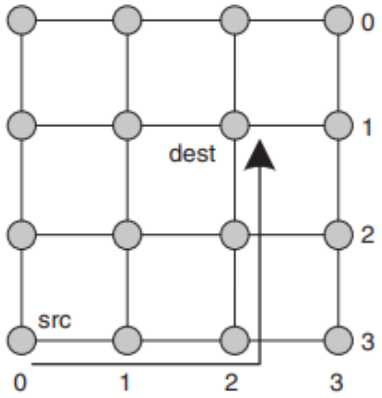


Figure 1-16: Exemple de l'algorithme de routage XY (Sheng Ma, Libo Huang, 2015).

1.4.3.4 Contrôle de flux

Le contrôle de flux est un mécanisme qui détermine le mouvement des paquets et gère l'allocation des tampons réseau et des liaisons sur les chemins réseau (Bjerregaard &

Mahadevan, 2006). Ce mécanisme n'est utilisé qu'en mode commutation par paquets. Au fur et à mesure que les paquets voyagent sur le réseau, ils sont temporairement stockés dans des tampons et attendent d'être traités à l'étape suivante. En ce sens, le contrôle de flux est nécessaire pour s'assurer que ces tampons ne débordent pas. Les schémas traditionnels de contrôle de flux de bout en bout incluent les schémas basés sur le crédit et les schémas basés sur ACK.

1.4.3.5 Qualité de service

Les applications au sein d'un NOC peuvent avoir besoin d'une certaine qualité de service pour assurer des contraintes sévères en débit et / ou en latence. Les autres services qui sont considérés comme essentiels sont l'intégrité des données, l'absence de perte des données et L'ordonnancement des données.

- ***Intégrité des données*** les données ne sont pas modifiées.

- ***L'absence de perte des données*** aucun paquet n'est perdu durant la transmission des données. La perte peut être due à une mauvaise gestion du trafic ou espace de stockage dans le nœud.

- ***L'ordonnancement des données*** les données doivent être reçu dans l'ordre dans lequel les émissions ont été reçues.

1.4.3 Réseaux sur puce émergents

Avec l'amélioration de la technologie, le nombre de noyaux intégrés à un SoC atteint plusieurs centaines ou même des milliers. Basée sur la technologie 90 nm, la latence de connexion de fils qui sont plus et plus n'est plus négligeable. D'autre part, la probabilité d'erreur sur les données échangées à travers le réseau est plus élevée à cause de plusieurs paramètres, entre autres, les interférences électromagnétiques (le bruit du cross talk) et les problèmes de synchronisation(Zerioul, 2016) .

De nouveaux paradigmes d'interconnexion sur puce ont émergé pour répondre aux limites traditionnelles de l'NOC. Les principaux aspects de recherche sont les interconnexions 3D et les interconnexions sans fil.

1.4.3.1 Réseau sur puce sans fils

Winnoc (Wireless Network on Chip) est une nouvelle technologie d'interconnexion système sur puce, qui vise à résoudre les limitations de performances telles que la latence élevée et la consommation d'énergie du NoC lorsque l'échelle du réseau augmente. Dans le WNoC, les PE sont divisés en plusieurs sous-réseaux qui ont un WR responsable de la communication sans fil pour les PE du même sous-réseau. L'allocation des routeurs sans fils WRs dans les sous-réseaux est une question importante pour décider des performances du système. La figure 1.17 illustre un exemple de WNoC avec 10x10 PEs divisés en 4 sous-réseaux rectangulaires et les routeurs sans fils WRs sont situés au centre de chaque sous-réseau.

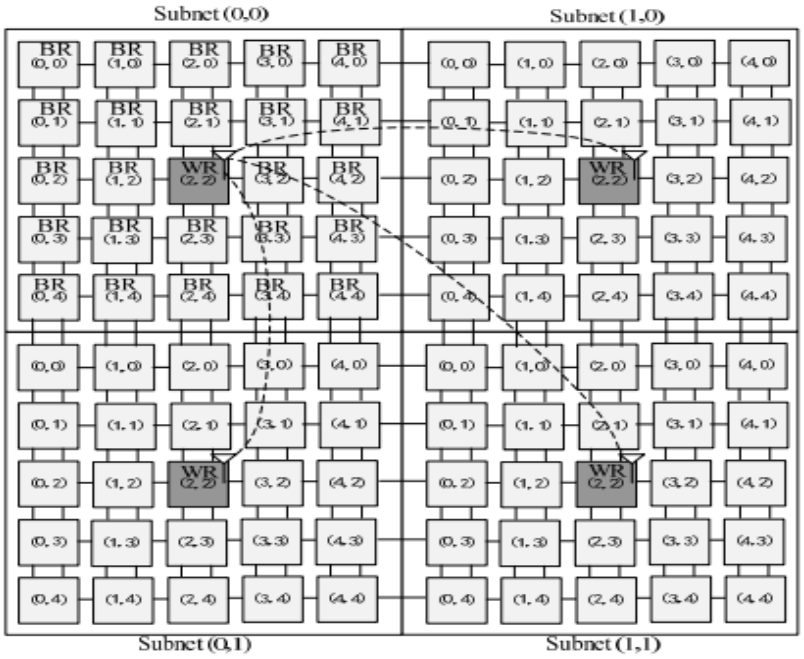


Figure 1-17: Architecture de WNOC 10x10 (C. Wang et al., 2011).

1.4.3.1.1 Composante d'un WNOC

Le WNoC est construit en remplaçant certains des routeurs de base du NOC par des routeurs sans fil (WR), qui ont des liens sans fil avec d'autres routeurs en plus des liens câblés d'origine. Par conséquent, les WR sont capables de transférer des paquets à la fois via des signaux câblés et des canaux sans fil (C. Wang et al., 2011).Le principe d'utilisation des routeurs sans fil entre les sous-réseaux est de réduire la latence du réseau, en particulier pour ces nœuds longue distance (S. Wang & Jin, 2014).

Le routeur sans fil en 2D maillé est comprenant de 5 ports bidirectionnels nommés North (N), South(S), Est(E), West(W), Local(L), crossbar switch et unité de contrôle logique, une unité de communication sans fil. Une unité de contrôle logique est composée par une logique de routage, un allocateur VC (Virtual Channel) et un arbitre de commutation. L'unité de communication sans fil est connectée au commutateur crossbar de sorte que les flits provenant des ports câblés peuvent être transmis à la liaison sans fil et vice versa. Mais il est interdit de transmettre les flits de l'entrée sans fil à la sortie sans fil.

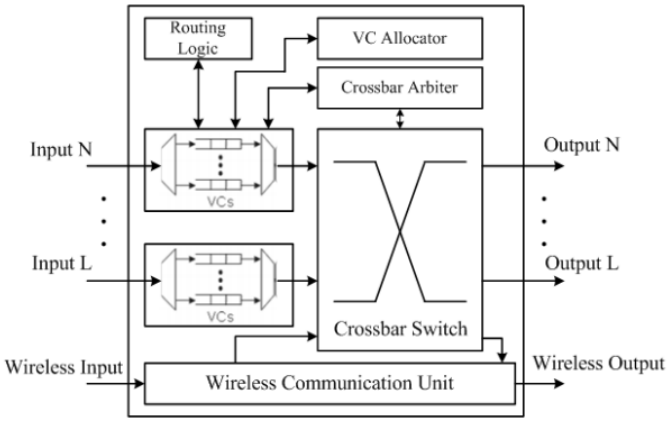


Figure 1-18: Microarchitecture de routeur Hybride Wired/Wireless (Yin et al., 2012).

1.4.3.2 Topologie 3D

Se définissent par plusieurs couches de silicium empilées ensemble et communicantes au moyen du lien d'interconnexion verticale qui est habituellement appelé TSV (Through Silicon Via). Le Noc 3D présente un certain nombre d'avantages, notamment la possibilité de placer davantage de routeurs sur une surface plus réduite grâce aux niveaux supplémentaires.

Par rapport à son équivalent en 2D, la densité du Noc est accrue, mais la latence et la distance entre les nœuds sont réduites.

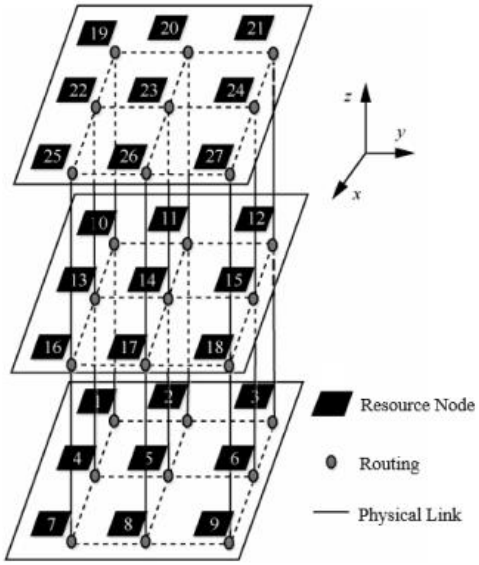


Figure 1-19: Exemple de la topologie 3D (Gan et al., 2021).

1.4.3.3 Composante d'un NOC-3D

Le NoC 3D comprendra les composants suivants(Vasic, 2014) :

- **Routeur** Ce routeur sera le cœur du NoC, et aura 7 ports (nord, sud, est, ouest, local, haut, bas). Il sera capable d'effectuer des connexions horizontales comme verticales, et de former un maillage connecté en 3D comme nous montre la figure 1-19.
- **Sérialiseur** Sert à réduire le nombre de signaux verticaux qui passent par les TSV vers une autre pile, ce qui signifie que le nombre de TSV nécessaire va diminuer.
- **Dé-sérialiseur** Permet de récupérer les signaux sérialisés provenant des TSV d'une autre pile afin de les utiliser dans le nouveau routeur 3D.
- **L'unité de synchronisation** est composée de deux synchroniseurs et de codeurs. Il synchronise les pointeurs de lecture et d'écriture de la FIFO, permettant ainsi un fonctionnement asynchrone du routeur 3D.
- **Interface réseau** est l'interface qui assurera la connexion entre le CPU et le routeur via le port local.

- **Unité de calcul** l'unité de calcul peut être constituée de tous les types de CPU, de DSP, etc. Elle reçoit des données du NoC ou envoie des données au NoC.

1.5 Conclusion

Ce premier chapitre nous a permis d'offrir une description générale sur les réseaux sur puce, nous avons expliqué pourquoi les bus et les connexions point à point ainsi le crossbar sont limités dans les SoC actuellement. Ces contraintes sont liées au faible parallélisme de la communication, mais aussi aux contraintes en termes de consommation énergétique en conséquence, les NoCs proposent des réponses aux contraintes limitatives de ces topologies de bus.

En effet, nous avons défini le concept de réseau sur puce et ses composants, puis étudions l'architecture OSI dans le noc, et détaillaient les caractéristiques du NoC à partir d'exemples concrets de topologies de réseaux sur puce connue, de techniques de commutation existantes, et de différents types d'algorithmes de routage utilisés dans les NoC. Outre la gestion des flux et la qualité de service sont également décrites. Les réseaux sur puce émergent comme des solutions efficaces pour surmonter les problèmes traditionnels NoC.

Un réseau sur puce est une technologie de communication qui utilise l'idée de réseau informatique pour relier plusieurs tâches. Dans le chapitre suivant, nous allons définir l'optimisation et les algorithmes de métaheuristique comme une solution de mapping.

Chapitre 2 : L’optimisation

2.1 Introduction

L’optimisation occupe actuellement une place importante dans le domaine informatiques tels que le mapping, l'assignation, etc., sont des problèmes d'optimisation au cours des phases de conception du réseau basé sur Noc. Elle consiste à explorer un espace de recherche ; afin de maximiser ou minimiser une fonction objective.

En outre, les problèmes d'optimisation sont résolus par des méthodes exactes, mais il est difficile de trouver une solution optimale à problème d'optimisation dans un temps limité. Il nécessaire disposer de techniques comme les heuristiques et les métaheuristiques qui tente d'offrir une solution proche de l'optimal dans un temps.

Dans ce chapitre, nous allons donner une idée générale sur les méthodes d'optimisation en particulier l'optimisation combinatoire, et une description de la méthode d’optimisation par l’algorithme des aigles royaux afin de résoudre notre problème de Mapping.

2.2 Définition du problème

L'optimisation est l'art de comprendre un problème réel et de pouvoir le transformer en un modèle mathématique qui peut être étudié pour extraire ses propriétés structurelles et décrire une solution au problème. Enfin, exploiter cette fonctionnalité pour déterminer les algorithmes qui les calculent est un art, tout en mettant en évidence les limites de l'efficience et de l'efficacité de ces algorithmes(MAHBOUBI, 2019; Saïd, 2013).

Un problème d'optimisation est défini par un espace de recherche, une ou plusieurs fonctions objectives et un ensemble de contraintes, pour le but du minimum ou maximum (optimum) d’une fonction donnée. On peut aussi trouver des problèmes d’optimisation pour

lesquels les variables de la fonction à optimiser sont soumises à des contraintes. Dans ce cas, on a une forme particulière de ce que l’on appelle un problème d’optimisation sous contraintes(MESSAOUDI, 2019).

2.3 Les méthodes de résolution de problèmes d’optimisation

Il existe de nombreuses méthodes d'optimisation ont été conçues pour rechercher la solution optimale, cependant une classification a été faite selon deux catégories sont : l’optimisation continue et combinatoire. Puisque nous utilisons l'algorithme métaheuristique comme une solution du problème de mapping, nous nous concentrerons sur l'optimisation combinatoire.

La figure 2-1 montre la classification générale des méthodes d'optimisation.

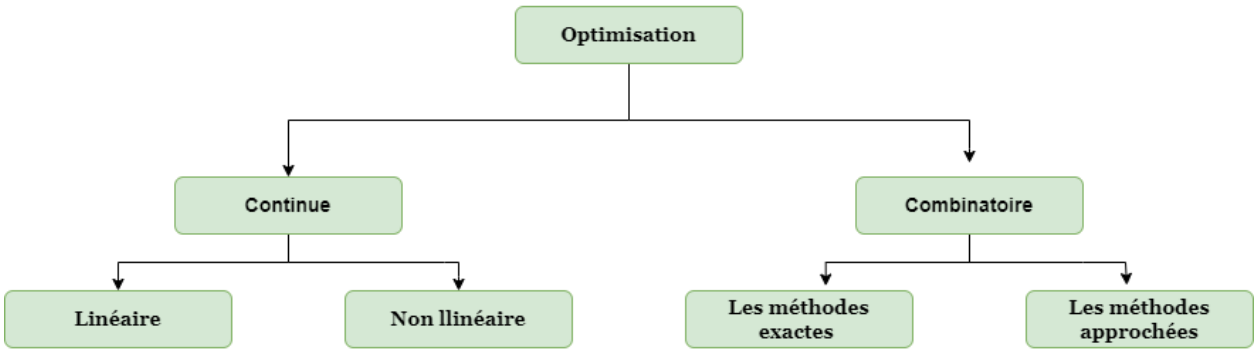


Figure 2-1: La classification générale des méthodes d'optimisation(Janga Reddy & Nagesh Kumar, 2020).

2.3.1 Optimisation continue

L'optimisation continue est caractérisée par une fonction objective continue dans laquelle les variables à optimiser peuvent prendre tout un continuum de valeurs. Il existe une distinction entre les problèmes linéaires (utilisant la programmation linéaire) où la fonction objective est linéaire, et les problèmes non linéaires (fonction objectif non linéaire et souvent inconnue).Pour les problèmes non linéaires, on trouve des méthodes globales (méta heuristique ou méthode classique) et locales (avec gradients ou sans gradients)(TOUATI, 2012).

2.3.2 Optimisation combinatoire

L'optimisation combinatoire occupe une place très importante en recherche opérationnelle, en mathématiques discrètes et en informatique. Son importance réside d'une part dans l'énorme difficulté des problèmes d'optimisation et d'autre part dans les nombreuses applications pratiques qui peuvent être formulées sous la forme de problèmes d'optimisation combinatoire. Bien que les problèmes d'optimisation combinatoire soient souvent faciles à définir, ils sont souvent difficiles à résoudre. En effet, la plupart de ces problèmes sont NP-difficiles(coût exponentiel), il n'existe donc actuellement aucune solution algorithmique efficace qui fonctionne sur toutes les données(Hao et al., 1999).

Les problèmes combinatoires sont des problèmes discrets, tels que le problème du voyageur de commerce ou le problème d'ordonnancement des tâches, qui consistent à trouver le meilleur élément parmi un ensemble fini de possibilités(TOUATI, 2012). Il s'agit de toute situation où l'on cherche une solution tout en respectant l'existence d'un ensemble de contraintes. La solution c’est une combinaison de ces contraintes d’une manière qu’on maximise quelques-uns et on minimise les autres, ces contraintes ont une caractéristique primordiale, c’est que chaque contrainte influe sur les autres soit quand on minimise sa valeur ou on la maximise. La résolution de problèmes d'optimisation combinatoire consiste à identifier des solutions qui optimisent une fonction objective donnée.

L'optimisation combinatoire peut être classée en deux grandes catégories : les méthodes exactes et les méthodes approchées.

2.3.2.1 Les méthodes exactes

L’intérêt des méthodes exactes est qu'elles garantissent l'obtention d'une solution optimale au problème traité. En effet, ils permettent de parcourir tout l'espace de recherche de manière à garantir l'obtention de toutes les solutions possibles meilleures que la solution optimale trouvée lors de la recherche. Cependant, les méthodes exactes sont très gourmandes en termes de temps de calcul, et de l’espace mémoire nécessaire, c’est la raison pour laquelle elles sont généralement préconisées pour résoudre les problèmes faciles et de petites tailles pour lesquels le nombre de combinaisons possibles est suffisamment faible pour pouvoir explorer l'espace de solutions dans un temps raisonnable(Salim & Selma, 2020).

Il existe de nombreux algorithmes exacts, y compris l'algorithme du simplexe, la programmation dynamique, l'algorithme A*, les algorithmes de séparation et évaluation (Branch and Bound, Branch and Cut, Branch and Price), les algorithmes de retour arrière (Backtracking).

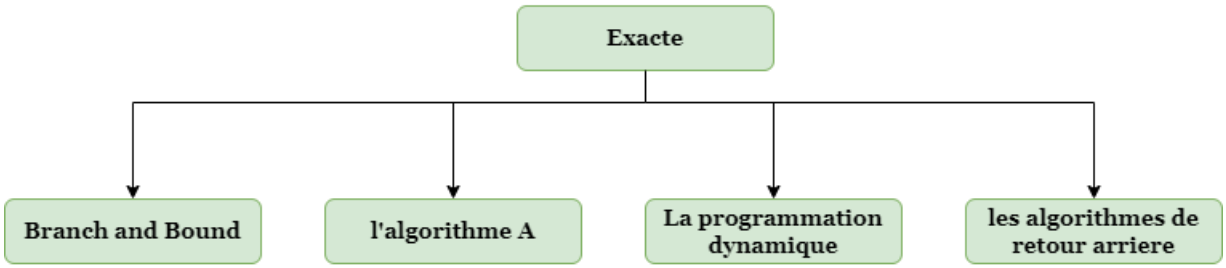


Figure 2-2: Les méthodes de résolution exacte (Janga Reddy & Nagesh Kumar, 2020).

- ***Branch and Bound***

C'est un algorithme d'optimisation déterministe appliqué sur le placement d'IPs sur une topologie de 2 D ayant pour objectifs de minimiser le cout de communication, son but est de trouver dans l'arbre du graphe la feuille ayant un cout d'énergie le plus faible possible(Delorme, 2007).

Le principe de cette méthode est de découper (branch) l'espace initial de recherche en domaines de plus en plus restreints afin d'isoler l'optimum global (principe de séparation). L'algorithme de recherche forme ainsi un arbre dont chaque nœud représente une partie de l'espace.

Le Branch and Bound est basé sur 3 axes principaux(Aroussi, 2016) :

- **L'évaluation** permet de réduire l'espace de recherche en éliminant quelques sous-ensembles qui ne contiennent pas la solution optimale.
- **Séparation** consiste à diviser le problème en sous-problèmes. Ainsi, en résolvant tous les sous-problèmes et en gardant la meilleure solution trouvée, on est assuré d'avoir résolu le problème initial. Cela revient à construire un arbre permettant d'énumérer toutes les solutions.
- **La stratégie de parcours** Il existe trois parcours possibles de l'arbre :

La profondeur d’abord Cette stratégie avantage les sommets les plus éloignés de la racine en appliquant plus de séparations au problème initial. Cette voie mène rapidement à une solution optimale en économisant la mémoire.

Le meilleur d’abord Cette stratégie consiste à explorer des sous-problèmes possédant la meilleure borne. Elle permet aussi d’éviter l’exploration de tous les sous-problèmes qui possèdent une mauvaise évaluation par rapport à la valeur optimale.

La largeur d’abord Cette stratégie favorise les sommets les plus proches de la racine en faisant moins de séparations du problème initial. Elle est moins efficace que les deux autres stratégies présentées.

2.3.2.2 Les méthodes approchées

Contrairement aux méthodes exactes, qui garantissent de fournir une solution optimale au problème, les méthodes approchées sont des alternatives très intéressantes pour traiter les problèmes d'optimisation de grande taille et ne fournissent pas nécessairement la solution optimale, mais seulement une bonne solution (de qualité raisonnable) dans un temps raisonnable(Martí & Reinelt, 2022), il s'agit donc ici de trouver la solution optimale ou la solution la plus proche de la solution optimale. Ces méthodes sont souvent classées en deux catégories : des méthodes heuristiques et des méthodes métaheuristiques.

2.3.2.2.1 Heuristique

Une heuristique d'optimisation est une méthode approchée se voulant simple, rapide et adaptée à un problème donné. Sa capacité à optimiser un problème avec un minimum d'informations est contrebalancée par le fait qu'elle n'offre aucune garantie quant à l'optimalité de la meilleure solution trouvée. Du point de vue de la recherche, ce défaut n'est pas toujours un problème, tout spécialement quand seule une approximation de la solution optimale est recherchée. Les heuristiques sont basées sur l'expérience et les résultats déjà obtenus, qui nécessitent une compréhension du domaine du problème traité(Ouardia, 2011).

2.3.2.2.2 Méta-heuristique

Métaheuristiques sont basées sur des idées générales inspirées par des systèmes naturels peuvent être appliquées à une large gamme de problèmes (N.Toubaline, 2018). Elles sont généralement conçues au départ pour des problèmes discrets, mais peuvent faire l'objet

d'adaptations pour des problèmes continus. On peut dire que les métaheuristiques sont des heuristiques modernes de plus haut niveau, spécialisées dans la résolution de problèmes d'optimisation. Leur objectif est d'atteindre des optima globaux tout en s'échappant des optima locaux (Hao et al., 1999).

On peut diviser les méthodes d'optimisation métaheuristiques en deux grandes classes :

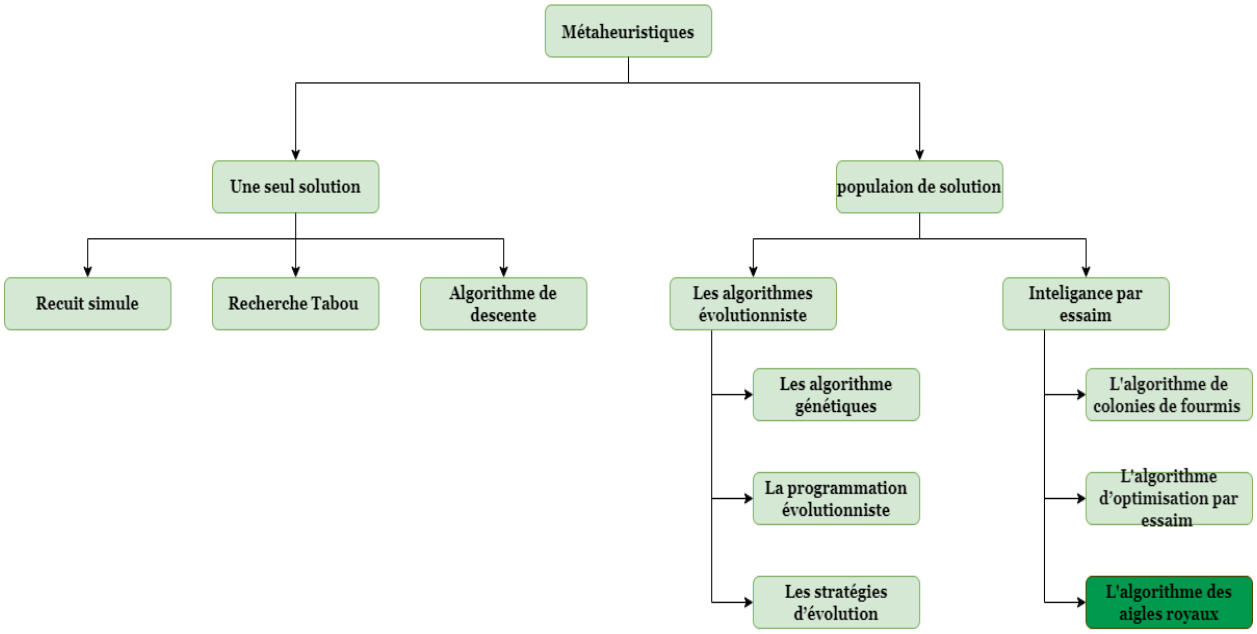


Figure 2-3: Taxinomie des méthodes d’optimisation métaheuristiques.

- ***Les métaheuristiques à solution unique***

Les métaheuristiques à solution unique ont plusieurs noms : elles peuvent être appelées, méthodes de recherche locale ou bien méthodes de trajectoire(Ghoumari, 2018). Elles se basent sur la notion du voisinage pour améliorer la qualité de la solution actuelle. Leur mécanisme consiste à démarrer la recherche avec une seule solution initiale, puis la solution initiale subit une série de modifications en fonction de son voisinage. Le but de ces modifications locales est d'explorer le voisinage de la solution actuelle afin d'améliorer progressivement sa qualité dans différentes itérations. La qualité de la solution finale dépend notamment des modifications apportées par les opérateurs de voisinages. En effet, une mauvaise transformation de la solution initiale conduit la recherche vers une vallée optimale locale dans un voisinage donné (éventuellement mauvais), ce qui bloque la recherche en fournissant une solution de qualité insuffisante(Gherboudj, 2013).

De nombreuses méthodes à base de solution unique ont été proposées dans la littérature. Nous présentons les méthodes les plus connues qui sont : le recuit simulé, la recherche taboue et l'algorithme de descente.

- ***Le recuit simulé (simulated annealing SA)***

L’algorithme de recuit simulé (RS) a été proposé par Kirk Patrick, Gellat et Vechi en 1983(Kirkpatrick et al., n.d.).L'algorithme SA simule le recuit est utilisé en métallurgie pour améliorer la qualité d'un solide. C’est ainsi nommé, car il est similaire au processus de recuit physique d'un solide, dans lequel le solide est chauffé puis refroidi lentement jusqu'à ce qu'il atteigne l’arrangement de réseau le plus régulier sans défauts cristallins. Le recuit dans la métallurgie et la science des matériaux définit un processus dans lequel la chaleur modifie les caractéristiques physiques et parfois chimiques d'une substance pour augmenter sa ductilité et réduire sa dureté(Bozorg-Haddad et al., 2017).Ce processus métallurgique a été transposé à une technique l'optimisation et a donné une méthode simple et efficace.

- ***La recherche taboue***

La recherche taboue a été proposée par Golver en 1986(Glover, 1986).elle est une méta-heuristique à base d’une solution unique, et une méthode de recherche locale avancée, fait appel à un ensemble de règles et de mécanismes généraux pour guider la recherche de manière intelligente au travers de l'espace des solutions(Bozorg-Haddad et al., 2017).La liste taboue est une mémoire flexible qui garde une certaine trace des dernières opérations passées. On peut y stocker des informations pertinentes à certaines étapes de la recherche pour en profiter ultérieurement. Cette liste peut être statique ou dynamique et permet d'empêcher les blocages dans les minima locaux en interdisant de passer à nouveau sur des configurations de l'espace de recherche précédemment visitées (problème du cycle).

- ***Algorithme de descente***

La recherche par descente est un très ancien algorithme amélioré. Le principe est d'explorer le voisinage de la solution actuelle pour améliorer progressivement sa qualité. L'algorithme sélectionne d'abord au hasard une solution, puis à chaque itération sélectionne la meilleure solution proche de la solution actuelle. Lorsque aucune autre amélioration n'est possible, l'algorithme s'arrête. Pour choisir la meilleure solution du voisinage, il existe plusieurs stratégies différentes, l'algorithme peut choisir celle qui a le meilleur fitness par rapport à toutes

les autres solutions du voisinage, ou choisir la première solution du voisinage qui améliore le fitness (Ghoumari, 2018).

- ***Les métaheuristiques à population de solutions***

Contrairement aux méthodes à solution unique, ces méthodes d'optimisation sont basées sur une population de solutions qu'il faut améliorer au fur et à mesure des itérations. L'idée d'utiliser un ensemble de solutions au lieu d'une seule solution renforce la diversité de la recherche et augmente la possibilité d'émergence de solutions de bonne qualité. Une grande variété des méthodes basées sur une population de solutions a été proposée dans la littérature, commençant par les algorithmes évolutionnaires, passant par les algorithmes génétiques et arrivant aux algorithmes à base d'intelligence par essaims (l'algorithme d'optimisation par essaim de particules, l'algorithme de colonies de fourmis, l'algorithme de colonies d'abeilles, la recherche coucou, …) (Gherboudj, 2013; Ghoumari, 2018).

- ***Les algorithmes génétiques***

L'algorithme génétique est utilisé pour minimiser ou maximiser la fonction objective en imitant le processus évolutif naturel. Les algorithmes génétiques en tant que technique d'optimisation a leur origine dans la nature, et les termes souvent utilisés ou apparentés sont biologiques. Les composants de base de l'AG comprennent la fonction de fitness pour l'optimisation une population d'individus, dont chacun, est une solution candidate du problème, cette représentation est appelée chromosome ; trois phases pour la rechercher de la solution sont le croisement, mutation et la sélection(Okwu & Tartibu, 2020).

La figure ci-dessous représentée la structure de l'algorithme génétique :

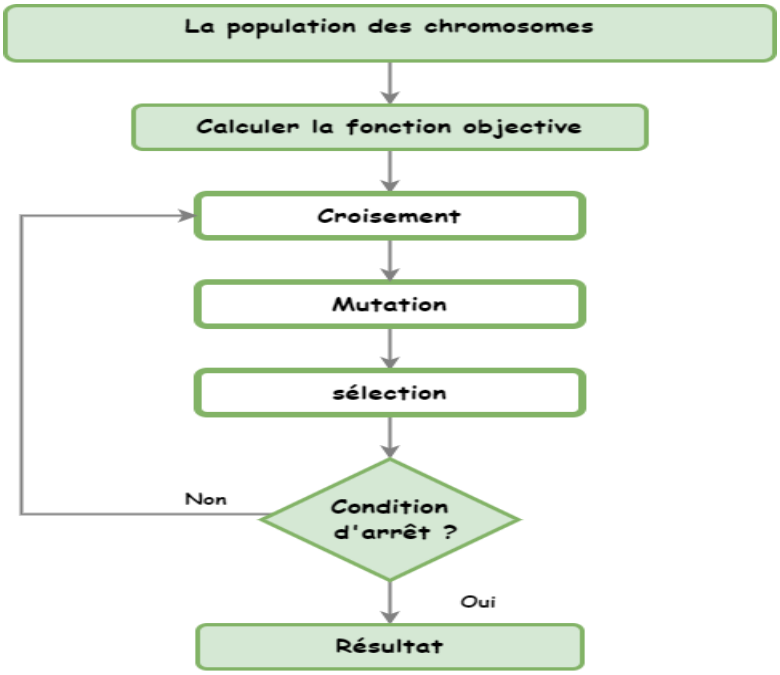


Figure 2-4: La structure de l'algorithme génétique.(Okwu & Tartibu, 2020).

- ***L'algorithme d’optimisation par essaim (PSO)***

PSO est une technique stochastique basée sur la population, inspirée du comportement social des oiseaux ou des poissons. L’optimisation par essaim fonctionne avec une population (appelée essaim) de solutions candidates. Chaque solution appelée une particule, évolue dans l'espace du problème en fonction de sa propre expérience ainsi que de l'expérience de ses voisins. Les particules évoluent pendant l'exploration de l'espace de recherche en utilisant leur propre meilleure position (meilleur local) et la meilleure position connue (meilleure globale) de l'ensemble de l'essaim. Si la particule nouvellement générée a une meilleure valeur de fitness par rapport à ses propres prédécesseurs, son meilleur local est mis à jour. Dans le cas de la meilleure solution globale, toutes les nouvelles particules générées pendant la phase d'évolution sont comparées et la particule avec la meilleure valeur de fitness est choisie comme étant la meilleure globale. Le processus est répété jusqu'à ce que certains critères de terminaison soient remplis(Chatterjee et al., 2018).

- ***L'algorithme de colonies de fourmis (ACO)***

L'optimisation basée sur les colonies de fourmis est utilisée dans plusieurs domaines. Il modélise le mouvement ou le comportement réel des fourmis pour résoudre le problème d'optimisation. Puisque les fourmis sont aveugles, elles se déplacent au hasard d'un endroit à un autre, c'est-à-dire qu'elles choisissent un chemin basé sur la probabilité(Okwu & Tartibu, 2020).

Les fourmis déposent au passage sur le sol une substance odorante appelée phéromone, ce qui a pour effet de créer une piste chimique. Les fourmis peuvent sentir ces phéromones qui ont un rôle de marqueur de chemin. Quand les fourmis choisissent leur chemin, elles ont tendance à choisir la piste qui porte la plus forte concentration de phéromones. Cela leur permet de retrouver le chemin vers leur nid lors du retour. D'autre part, les odeurs peuvent être utilisées par d'autres fourmis pour retrouver les sources de nourriture détectées par leurs consœurs (Ouardia, 2011).

Optimisation par l'algorithme aigle royal

Les aigles sont l'un des plus grands oiseaux de proie. Les aigles se caractérisent par la vitesse à laquelle ils atteignent leur vol, leurs muscles de grande force et leur excellente vision et une grande taille. Il existe différents types tels que Greater spotted Eagle, Philippine Eagle et Golden Eagle.

Golden Eagle possède des plumes brun sombre sur le corps et plutôt brun doré à brun roux sur le camail, c'est-à-dire cette sorte de collier qui orne son cou et sa tête. Son bec crochu de 6,5 cm de longueur est grisâtre et il est surmonté d'un opercule de couleur jaune que l'on appelle la cire. Ce bec est puissant et sa forme, propre aux rapaces, lui permet de déchiqueter ses proies qu'il maintient bien fermement grâce à ses serres impressionnantes et à l'avillon de son pouce. C'est d'ailleurs avec cette griffe affûtée de 7 cm de longueur qu'il transperce le corps de ses victimes. Ce majestueux rapace jouit d'une vue perçante, de l'ordre de huit fois plus développée que celle de l'Homme. Il est le grand prédateur des marmottes, des lapins, des belettes ou encore des renards et même des agneaux. L'aigle royal peut en effet enlever tout représentant de la faune alpine pesant jusqu'à deux fois son propre poids. L'espérance de vie de l'aigle royal est d'environ 25 ans (*L'aigle Royal, Le plus Majestueux Des Rapaces*, n.d.).

Figure 2-5 : Golden Eagle. (Fennema, 2018).

2.4 Algorithme aigle royal

Algorithme aigle royal est une nouvelle méthode méta-heuristique récemment introduite pour résoudre des problèmes d'optimisation globale. L'algorithme GEO s'est inspiré de l'intelligence de l'aigle royal et l'a modélisé mathématiquement en fonction de la vitesse qui contrôle sa trajectoire en spirale.

L'aigle royale est un type particulier d'essaim, leur comportement est régi par deux forces : la propension à attaquer et la propension à croiser. L'aigle royal sait que s'il est attaqué rapidement, il peut attraper de petites proies qui ne peuvent même pas compenser l'énergie consommée par la chasse. D'un autre côté, s'ils chassent sans cesse des proies plus grosses, ils peuvent manquer d'énergie et ne rien attraper. Les aigles royaux créent intelligemment un équilibre entre ces deux désirs d'attraper la meilleure proie dans un temps raisonnable et avec une quantité d'énergie raisonnable (Mohammadi-Balani et al., 2021).

Chaque GE (Golden Eagle) se souvient du meilleur poste qu'il a occupé jusqu'à présent. Au fur et à mesure que le nombre d'itérations augmente, le **GE**$_i$ choisit la proie d'un

autre $\mathbf{GE_j}$ et tourne autour de la meilleure position de $\mathbf{GE_j}$. Il peut également choisir de tourner autour de la meilleure position de mémoire (Amor et al., 2022).

Cela dit que les principales caractéristiques du processus de chasse des aigles royaux peuvent être résumées comme suit.

- Ils suivent une trajectoire en spirale pour la recherche et une trajectoire rectiligne pour l'attaque.
- Ils montrent une plus grande propension à croiser dans les premières étapes de la chasse et une plus grande propension à attaquer dans la dernière étape.
- Ils conservent la tendance à la fois à la croisière et à l'attaque à chaque instant du vol.
- Ils recherchent les informations sur les proies dans les mémoires des autres aigles.

❖ ***Le modèle mathématique***

Les formulations mathématiques des aigles royaux pour imiter les mouvements de recherche des proies sont principalement décrites par :

- **Le mouvement en spirale des aigles royaux**

L'aigle royal basé sur le mouvement en spirale et à une attirance pour la croisière et pour attaquer simultanément la proie à la recherche d'une meilleure nourriture. La figure 2-6 représente les vecteurs de croisière et d'attaque dans l'espace 2D.

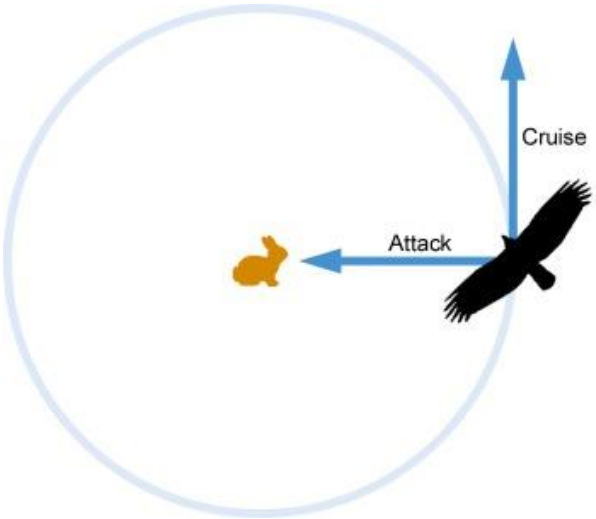


Figure 2-6 : Mouvement en spirale des aigles royaux **(Mohammadi-Balani et al., 2021) .**

Chaque aigle royal se souvient du meilleur endroit où il a été jusqu'à présent, donc à chaque itération, chaque aigle royal ***j*** peut choisir au hasard une proie qui a été capturée par un autre aigle royal ***i*** et tourne autour de la meilleure position visitée par l'aigle royal ***i*** jusqu'à présent. L'aigle royal ***j*** a aussi la particularité de sélectionner pour encercler sa propre mémoire ; ainsi, nous avons $\boldsymbol{i} \in \{1, 2, \cdots, N_{GE}\}$, où $\boldsymbol{N_{GE}}$ représente le nombre d'aigles royaux (Joshi & Biradar, 2021).

- **Sélection des proies**

Comme nous avons mentionné précédemment à chaque itération, chaque aigle royal doit sélectionner une proie (Prey) pour effectuer les opérations de croisière et d'attaque. De plus, chaque aigle royal choisit la proie désirée dans la mémoire de la population.

Par conséquent, les vecteurs de croisière et d'attaque sont calculés en fonction de la proie choisie. Ensuite, il vérifia sa mémoire. Si le nouvel emplacement est meilleur que l'emplacement précédent, la mémoire est mise à jour avec les nouvelles découvertes (Joshi & Biradar, 2021).

- **Attaque**

L’attaque peut être modélisée via un vecteur qui commence à l'emplacement actuel de l'aigle royal et se termine à l'emplacement de la proie dans la mémoire de l'aigle. Le vecteur d'attaque de l'aigle royal peut être calculé par l’équation (Mohammadi-Balani et al., 2021).

$$\overrightarrow{A_\iota} = \overrightarrow{X_f^*} - \overrightarrow{X_\iota} \qquad (2\text{-}1)$$

Où :

$\overrightarrow{A_\iota}$ Représente le vecteur d'attaque de l'aigle royal ***i.***

$\overrightarrow{X_f^*}$ Représente la meilleure position visitée par proie ***f*** jusqu'à présent.

$\overrightarrow{X_\iota}$ Représente la position actuelle de l'aigle ***i***.

- **Croisière**

Le vecteur de croisière est le vecteur perpendiculaire au vecteur d'attaque et tangent au cercle, et il est calculé en fonction du vecteur d'attaque. Elle est également connue sous le nom de vitesse linéaire à laquelle l'aigle royal attaque sa proie. Le point de destination sur le vecteur de croisière est donné ci-dessous :

$$C_k = \frac{d - \sum_{i,i\neq k} a_i}{a_k} \qquad (2\text{-}2)$$

(Amor et al., 2022).

Où :

$a_i \in \overrightarrow{A_\iota}$, où $\overrightarrow{A_\iota} = \{a_1, a_2, a_3, \ldots, a_n\}$.

d Représente l'équation de l'hyperplan dans l'espace à n dimensions.

$$d = \sum_{i=1}^{ni} a_i x_i \qquad (2\text{-}3)$$

- **Passage à une nouvelle position**

Le déplacement vers de nouvelles positions des aigles royaux dépend principalement des vecteurs d'attaque et de croisière. Par conséquent, le vecteur de pas (Step vector) de l'aigle royal ***i*** à l'itération ***t*** est présenté par l'équation suivante (Mohammadi-Balani et al., 2021) mais dans ce cas nous modifions l'équation (2-4):

$$\Delta x_i = \overrightarrow{r_1} p_a \frac{\overrightarrow{A_\iota}}{\|\overrightarrow{A_\iota}\|} + \overrightarrow{r_2} p_c \frac{\overrightarrow{C_\iota}}{\|\overrightarrow{C_\iota}\|} \qquad (2\text{-}4)$$

Où :

p_a et p_c Sont respectivement les coefficients d'attaque et de croisière.

$\overrightarrow{r_1}$ et $\overrightarrow{r_2}$ Sont des vecteurs aléatoires entre 0 et 1.

$\|\overrightarrow{A_\iota}\|$ et $\|\overrightarrow{C_\iota}\|$ Sont la norme euclidienne des vecteurs d'attaque et de croisière et sont calculée en utilisant les équations (Amor et al., 2022).

$$\|\overrightarrow{A_\iota}\| = \sqrt{\sum_{i=1}^{n} a_i^2} \qquad (2\text{-}5)$$

$$\|\overrightarrow{C_\iota}\| = \sqrt{\sum_{i=1}^{n} c_i^2} \qquad (2\text{-}6)$$

La nouvelle position de l'aigle royal est alors donnée par (Amor et al., 2022) :

$$x^{t+1} = x^t + \Delta x_i^t \qquad (2\text{-}7)$$

Si le coût de communication de la nouvelle position de l'aigle royal est meilleur que la position dans sa mémoire, la mémoire de cet aigle ***i*** est mise à jour avec la nouvelle position.

Sinon, la mémoire reste telle qu'elle est, mais l'aigle résidera dans la nouvelle position.

Transition de l'exploration à l'exploitation

L'algorithme GEO utilise le coefficient d'attaque p_a et le coefficient de croisière p_c pour passer de l'état d'exploration à l'état de d'exploitation.

L'algorithme commence par p_a bas et p_c haut. Au fur et à mesure que l'itération progresse, p_a augmente progressivement, tandis que p_c diminue progressivement [GEO]. p_a et p_c peuvent être calculés à l'aide des expressions linéaires suivantes (Mohammadi-Balani et al., 2021):

$$p_a = p_a^0 + \frac{t}{T}\left|p_a^T - p_a^0\right| \qquad (2\text{-}8)$$

$$p_c = p_c^0 + \frac{t}{T}\left|p_c^T - p_c^0\right| \qquad (2\text{-}9)$$

Tel que :

p_a^0 et p_c^0 Sont les valeurs initiales respectivement de propension à attaquer p_a et de propension à croiser p_c .

p_a^T et p_c^T Sont les valeurs finales respectivement de propension à attaquer p_a et de propension à croiser p_c .

2.5 Conclusion

Les méthodes d'optimisation sont très nombreuses, elles s'appuient sur des principes complètement différents, chacune explore et exploite l'espace de recherche selon des techniques qui lui sont propres.

Dans ce chapitre nous avons présenté un aperçu général sur les métaheuristiques en présentant d'abord quelques notions préliminaires sur les principaux algorithmes d'optimisation, en les divisant en deux classes : les méthodes qui font évoluer une seule solution et les méthodes à base de population de solutions, nous avons étudié les différentes métaheuristiques à population (colonie de fourmis, essaims particulaire) et nous avons focalisé nos recherches sur les algorithmes aigle royal afin de résoudre notre problème de mapping.

Dans le chapitre suivant, nous allons présenter comment mapper les différentes tâches du réseau sur puce, ainsi que les différents algorithmes de méta heuristique utilisées pour optimiser les performances du système.

.

Chapitre 3 : Problème de Mapping

3.1 Introduction

Réseau sur puce est une architecture qui permet de relier plusieurs composants d'un système sur silicium. Bien que sa conception soit délicate et complexe, il permet de remédier aux limites des systèmes de communication classiques. Pour y remédier, il faut automatiser les différentes procédures en utilisant des techniques. De bons algorithmes sont proposés pour obtenir une performance maximum pour une application donnée.

Parmi les défis de la conception des réseaux sur puce, on trouve le problème du placement (Mapping). Ce problème implique de trouver l'emplacement des tâches d'application sur les éléments de calcul pour une topologie Noc donnée. En effet, un mauvais placement des composants logiciels d'une application peut réduire considérablement les performances globales du système final(Belkacemi, 2020).

C'est pourquoi le mapping constitue une phase importante et pertinente dans le flot de conception des réseaux sur puce. Nous décrirons dans ce chapitre certains algorithmes de mapping existant pour les topologies Noc 3D, 2D et WNoC. Mais d'abord, on va définir le mapping, problème du mapping, les types de Mapping statique et dynamique.

3.2 Définitions

Le mapping est une phase de conception de réseau sur puce de complexité NP difficile qui est directement liée à l'implémentation de l'application sur une architecture spécialisée, et permet de placer chaque tâche d'une application sur une ressource IP d'une architecture dédiée (affectation de tâches à des éléments de calcul), tout en respectant les contraintes de temps réel

de l'application, en termes de la consommation et le temps d'exécution offrent les meilleures performances(Delorme & Houzet, 2006).

Lors de la spécification d'un Noc, le mapping participe directement à la réalisation des objectifs souhaités, tels que la réduction de la consommation d'énergie et la maximisation du temps de performance ou de l'équilibrage de la charge mémoire(Boumaaza & Benyamina, 2012). Cette phase de conception nécessite des entrées qui sont :

- Un graphe d'application.
- Un graphe d'architecture.
- Un ensemble de fonctions objectives à optimiser.
- Un ensemble de contraintes à considérer.

3.2.1 Graphe d'application (GAP)

L'application est spécifiée comme étant un ensemble de blocs communicants est généralement modélisée par un graphe de tâche (Task graph).

GAP (T, E) est un graphe orienté acyclique, avec chaque sommet $\boldsymbol{t_i} \in \boldsymbol{T}$ représente une tâche d'application et chaque arc directionnel $\boldsymbol{e_{i,j}} \in \boldsymbol{E}$ représente une communication de la tâche $\boldsymbol{t_i}$ au tâche $\boldsymbol{t_j}$(Kundu & Chattopadhyay, 2014).

$\boldsymbol{v_{i\,j}}$est arc de volume du nœud $\boldsymbol{t_i}$ au $\boldsymbol{t_j}$, qui représente le volume de la communication (bits) de $\boldsymbol{c_i}$à $\boldsymbol{c_j}$(Delorme & Houzet, 2006).

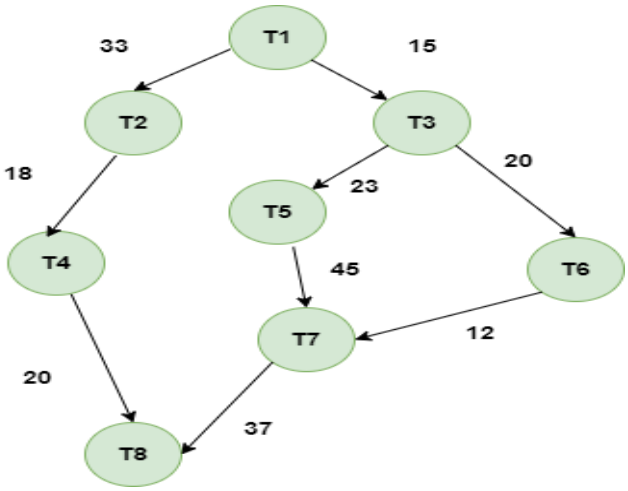


Figure 3-1: graphe d'application.

3.2.2 Graphe d'architecture (GAR)

Une topologie réseau sur puce peut être modélisée par un graphe, appelle un graphe d'architecture (Topology graph).

- **La topologie maillée 2Det WNoC**

GAR (U, L) est un graphe orienté acyclique, avec chaque sommet de graphe $\boldsymbol{u_i} \in \boldsymbol{U}$ représente un nœud dans la topologie Noc et chaque arc directionnel $\boldsymbol{l_{ij}} \in \boldsymbol{L}$est un lien physique de la topologie Noc et représente une communication directe entre les nœuds $\boldsymbol{u_i}$ et $\boldsymbol{u_i}$ (Kundu & Chattopadhyay, 2014).

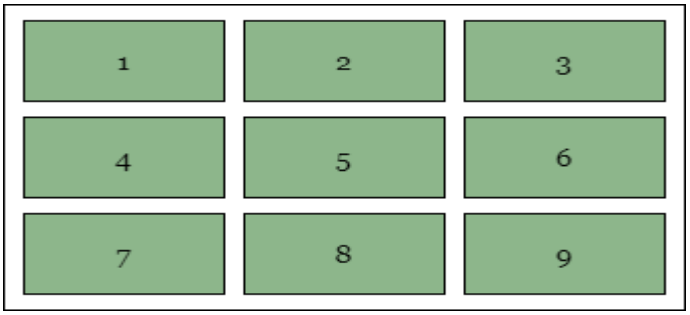


Figure 3-2: Graphe d'architecture (Xiaodong et al., 2020).

- **La topologie maillée 3D**

GAR (U, L) est un graphe orienté acyclique, où $|U| = d_1 \times d_2 \times d_3$ et chaque sommet $\boldsymbol{u_{i_1,i_2,i_3}} \in \boldsymbol{U}$ (à condition que $\boldsymbol{i_k} \in \{\boldsymbol{1, 2, \ldots, d_k}\}$**:** $\boldsymbol{1 \leq k \leq 3}$) Représente un nœud de la topologie, et l'arc $\boldsymbol{l_{ij}} \in \boldsymbol{L}$indique la présence d'un lien de communication directe entre les deux éléments $\boldsymbol{u_{i_1,i_2,i_3}}$ et $\boldsymbol{u_{j_1,j_2,j_3}}$ qu'est défini selon (Fang et al., 2019):

$$\boldsymbol{u_{ij}} = \begin{cases} \boldsymbol{1},\ \sum_{k=1}^{3} |\boldsymbol{i_k} - \boldsymbol{j_k}| = \boldsymbol{1} \\ \boldsymbol{0}, \boldsymbol{Sinon} \end{cases} \qquad \textit{(3-1)} \text{ (MESSAOUDI, 2019)}$$

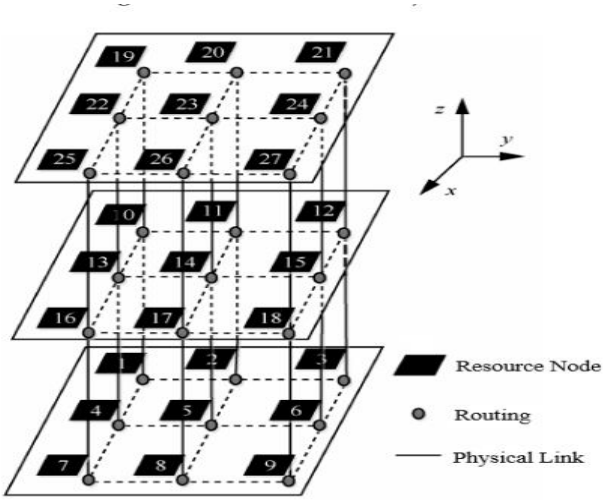


Figure 3-3: Graphe d'architecture 3D(Gan et al., 2021).

3.2.3 Mapping

Le mapping permet d'effectuer les T tâches de l'application à une architecture (topologie Noc) de $\boldsymbol{U}$ processeurs, pour applique le Mapping en utilisant les deux graphes définis ci-dessus, graphe d'application ***GAP*** et le graphe d'architecture ***GAR***, tel que :

Les différentes tâches de l'application sont mappées dans des processeurs différents. (Ne peut pas avoir plusieurs tâches mappées sur un seul processeur).

$$\forall\, \boldsymbol{t_i} \in \boldsymbol{T}, \boldsymbol{Map}(\boldsymbol{t_i}) \in \boldsymbol{U} \qquad \textbf{(3-2)}\ \text{(Delorme \& Houzet, 2006)}$$

$$\forall\, \boldsymbol{t_i} \neq \boldsymbol{t_j} \in \boldsymbol{T}, \boldsymbol{Map}(\boldsymbol{t_i}) \neq \boldsymbol{Map}(\boldsymbol{t_j}) \qquad \textbf{(3-3)}\ \text{(Delorme \& Houzet, 2006)}$$

Les figures 3-4 et 3-5 représentées des exemples de mapping sur les topologies 2D et 3D.

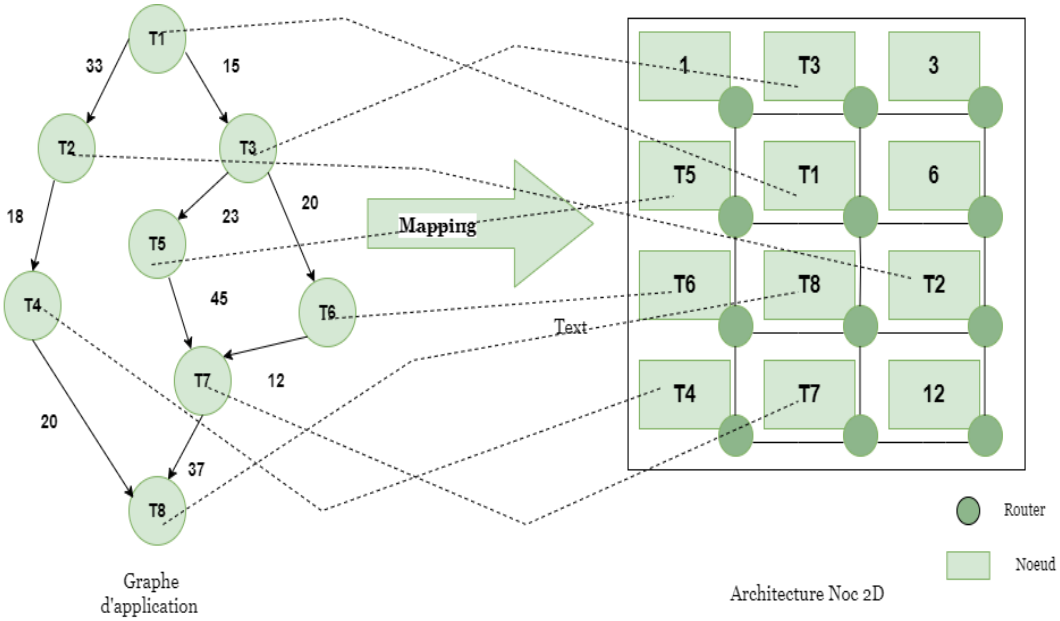


Figure 3-4: Exemple de mapping sur l'architecture 2D NOC.

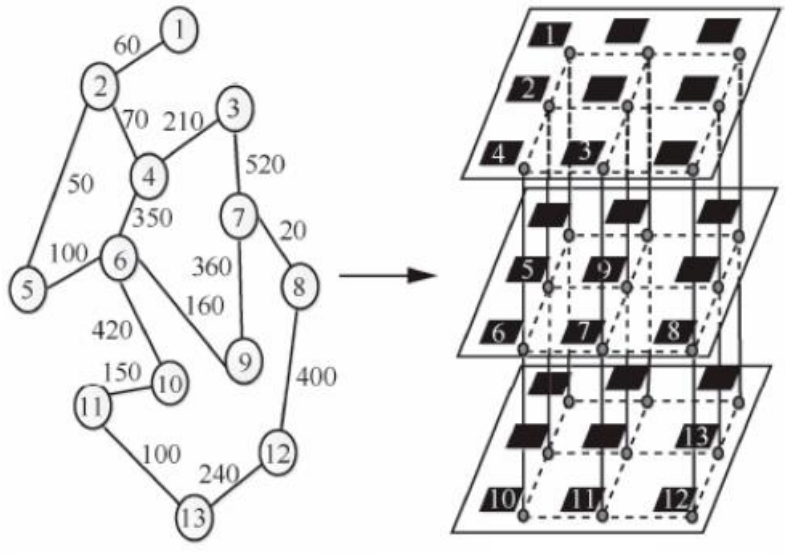


Figure 3-5: Exemple de mapping sur l'architecture 3D Noc(Gan et al., 2021).

- **Mapping sur l'architecture du WNoC**

En WNoCs, il y a deux types de mapping :

le mapping des routeurs WR dans le réseau (placement des WR dans le réseau) qui fournit une communication de données sans fil express pour les PE dans les mêmes ou différents sous-réseau(Hu et al., 2012).

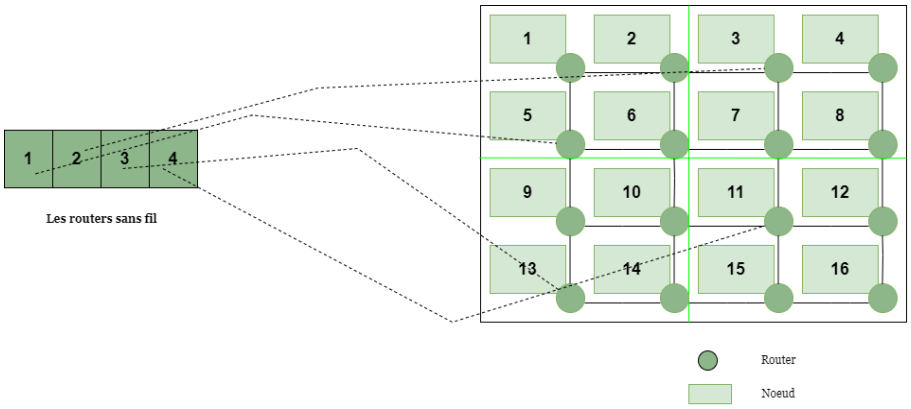


Figure 3-6: Exemple de Mapping des routeurs sans fil sur une architecture de WNoC.

Le mapping les IPs de l'application à une architecture réseau sur puce sans fil, comme le montre la figure 3 -7.

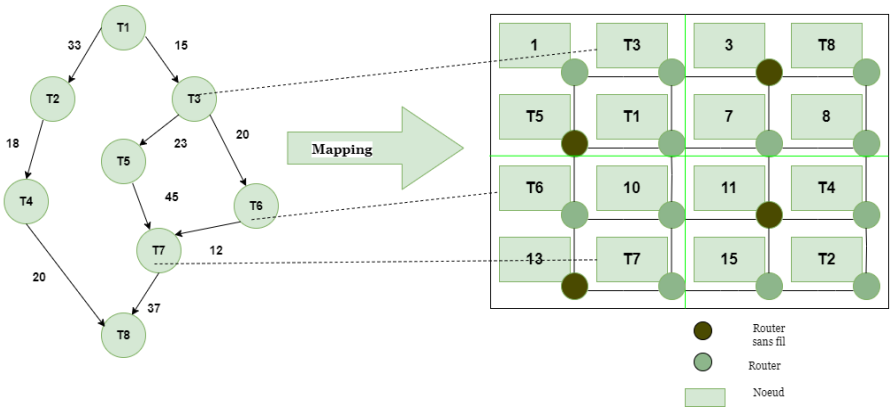


Figure 3-7: Exemple de Mapping des IPs sur une architecture de WNoC.

3.2.4 Fonctions objectives

Fonctions objectives jouent un rôle très important dans la phase de Mapping, elle permet d'évaluer les performances définies par les concepteurs, comme les contraintes de temps, les contraintes de taille mémoire, etc. Ainsi que s'efforcer à optimiser l'objectif (ou les objectives) que la réduction du coût de la communication, de la consommation de temps d'exécution …etc(Belkacemi, 2020; MESSAOUDI, 2019).

Un coût de communication inférieur réduit considérablement la consommation d'énergie et la latence en raison de l'utilisation d'un plus petit nombre de sauts entre les nœuds.

$$\boldsymbol{Cost\ C} = \sum_{\boldsymbol{H_{count}}} \times \boldsymbol{BW}(\boldsymbol{p_i}, \boldsymbol{p_j}). \qquad \textbf{(3-4)}$$

Tel que :

$\boldsymbol{Cost\ C}$: représente le coût de communication entre les nœuds.

$\boldsymbol{BW}$: est la valeur qui indique l'exigence de bande passante pour chaque arête.

$\boldsymbol{H_{count}}$: est le nombre de sauts représenté dans le chemin de communication.

3.3 Types de Mapping

Les techniques de Mapping peuvent être classées comme Mapping statique ou Mapping dynamique.

3.3.1 Mapping statique

Un mapping est dit statique (appelé static ou off-line mapping) si la ressource de l'architecture sur laquelle la tâche de graphe doit être effectuée est finalise avant l'exécution de l'application et toutes les tâches sont mappées aux nœuds l'architecture Noc au moment de la conception et non modifiées par la suite(Sahu & Chattopadhyay, 2013) .

3.3.2 Mapping dynamique

Un mapping est dit dynamique (appelé Dynamic ou on-line mapping) si la ressource de l'architecture sur laquelle la tâche de graphe doit être effectuée est déterminée au moment de son exécution. Ainsi le placement des tâches sur Noc peut être modifié pendant l'exécution de l'application (Sahu & Chattopadhyay, 2013).

3.4 Problème de Mapping

Le mapping d'application est un problème NP-difficile (non-polynomial hard), car pour placer m IPs sur n nœuds de réseaux. Le nombre de solutions possibles devient ***n!***, donc

l'espace de recherche augmente graphiquement avec l'augmentation des cœurs(Xiaodong et al., 2020).

3.5 Algorithmes de Mapping existants

En raison de la croissance rapide des Noc, le problème de mapping a attiré l'attention de nombreux chercheurs, c'est un problème qui nécessite des algorithmes heuristiques ou métaheuristiques efficaces pour trouver des solutions et vise à atteindre différents objectifs. Par exemple, réduire la consommation d'énergie, les coûts de communication, les délais...

Comme nous l'avons mentionné au chapitre 1, il existe de nombreuses architectures Noc, nous mentionnerons quelques études de mapping sur les topologies 2D, 3D et WNoC.

3.5.1 Les méthodes de mapping Noc 2-D

- ***Sailfish optimization algorithm***

Un algorithme d'optimisation sailfish (SFOA) inspire de la nature est utilisée pour le mapping optimisé du graphe de tâches de l'application sur un Noc bidimensionnel avec une topologie maillée, via l'utilisation de l'approche de clustering partagé K-plus proche voisin dans le but de minimiser le temps de calcul, la consommation d'énergie et le coût de communication (Sikandar et al., 2021).

- ***Ant Colony Algorithm Based on Task Scale***

Dans une autre étude, la méthode de mapping basé sur la taille des tâches utilise une version améliorée de l'optimisation des colonies de fourmis après avoir divisé les tâches en fonction de la taille de la tâche. Lorsque la taille de la tâche est petite, une sous structure Noc donnée est sélectionnée pour mapper la tâche afin que les tâches puissent être centralisées ; lorsque l'échelle des tâches est importante, les tâches sont organisées en clusters et divisent certaines tâches avec des dépendances au même nœud de ressource. Cette approche est idéale pour réduire la consommation d'énergie par nœud et la latence dans la structure Noc (Fang et al., 2019).

- ***MBA : Modified bat algorithm***

L'algorithme métaheuristique de chauve-souris modifié (MBA) a été proposé en 2018 (Alagarsamy & Gopalakrishnan, 2018) comme une méthode de mapping base sur la bande passante et l'énergie pour les Nocs 2D, afin de réduire la consommation d'énergie du réseau et le coût de communication.

3.5.2 Les méthodes de mapping Noc 3-D

- ***ILP formulation and heuristic method***

Cet article (Nalci et al., 2021) propose une nouvelle méthode de mapping d'une application sur des topologies Noc 3D-Mesh basée sur un algorithme heuristique appelé CastNet3D et la formulation ILP (Integer linear Programming), dont l'objectif principal est de minimiser la consommation d'énergie dynamique de la communication de données entre les nœuds d'application (minimiser la consommation d'énergie), l'algorithme CastNet3D obtient des résultats proches de l'optimal.

- ***Chicken swarm optimization algorithm (SCSO)***

Le travail présente dans cet article (Alagarsamy et al., 2019) utilise une technique d'optimisation des essaims de poulets (scso) comme une solution pour mapper une application a une topologie Noc 3D et visait à réduire le coût de communication, et pour gérer le mapping initial cet algorithme est utilisé la technique de regroupement des k plus proches voisins.

- ***Improved Simulated Annealing Genetic Algorithm ISAGA***

ISAGA est une méthode de mapping à faible puissance basée sur deux algorithmes métaheuristiques, l'algorithme génétique (GA) et recuit simulé (SA). Cette méthode combine le parallélisme de l'algorithme génétique (GA) qui sélectionne la population initiale pour obtenir le schéma de mapping à faible consommation d'énergie et la capacité de recherche locale du recuit simulé (SA)(He et al., 2018).

- ***V-CastNet3D***

Un autre algorithme de recherche heuristique proposée appelé V-CastNet3D qui traite une architecture de 3DMeshdonnée comme une série d'architecture 2DMeshconnectées verticalement. Il regroupe les nœuds en fonction du nombre de couches 3D. Il utilise ensuite lemappage2D pour mapper chaque cluster aux couches adjacentes. Pour réduire le délai et la consommation d'énergie du réseau et le coût de communication (Dageleh & Jamali, 2018).

3.5.3 Les méthodes de mapping WNoC

- ***Round Rotary Mapping***

Une nouvelle approche de mapping des tâches pour HW Noc a été proposée nommée Round Rotary Mapping (RRM), visant à résoudre deux problèmes critiques dans HW Noc. D'équilibrer d'abord l'utilisation des liaisons sans fil en évitant la congestion sur les routeurs sans fil et en deuxième lieu la propagation de la température sur l'ensemble de la puce en utilisant du silicium noir (Rezaei et al., 2017).

- ***Dynamic Application Mapping Algorithm (DAMA)***

Un algorithme de mapping dynamique des applications (DAMA) sur l'architecture hiérarchique de réseau sans fil sur puce sans fil qui est basée sur un Noc maillé 2D a été proposé afin de réduire la congestion interne et externe. L'algorithme comporte trois étapes clés, la première étape promettre de trouver le premier nœud à mapper ensuite choisir la première tâche qui a le grand nombre des arcs à mapper sur le premier nœud et la dernière étape attribuée les tâches restantes aux nœuds restants (Rezaei et al., 2015).

- ***Genetic algorithm (GA)***

une approche basée sur un algorithme génétique pour trouver des solutions optimales de cartographie des tâches au moment de la conception, pour les systèmes embarqués fonctionnant sur un WNoC, dans le but de minimiser la consommation d'énergie, la bande passante et l'accélération(Sacanamboy-Franco et al., 2017).

3.6 Synthèse entre les différentes méthodes de mapping

Plusieurs travaux de recherche que nous avons cités dans ce chapitre ont indiqués clairement les différentes méthodes utilisées pour résoudre le problème d'optimisation (les différentes techniques étudiées dans le chapitre précédent) sur les topologies 2D, 3D et WNoc.

Les travaux déjà cités utilisent la topologie 2D-Mesh pour le but d'améliorer leur performance. Ceci est dû à la facilité de son implantation sur la technologie silicium et sa capacité de s'adapter à une taille de réseau variable. Cependant, lorsque la taille des tâches est importante, en vue

comme un inconvénient qui influence la performance du système (Fang et al., 2019),tel que la latence, la consommation d'énergie et le coût de communication ...

D'autre part, l'augmentation du nombre de cœurs et du nombre de transistors dans les topologies 2D a conduit à l'augmentation des dimensions des circuits et à l'allongement des fils de communication. La technologie de fabrication de circuits intégrés 3D a été récemment développée et vise à réduire le retard des longs fils globaux à travers une puce (Dageleh & Jamali, 2018).

D'autres travaux déjà cités qui sont basés sur la topologie 3D, est une topologie offre une plus grande performance que la topologie 2D. Dans l'exemple (puce (Dageleh & Jamali, 2018) et (He et al., 2018) sont traites une architecture de 3D Mesh donnée comme une série d'architectures 2D Mesh connectées verticalement.

D'autres idées ont été proposées d'utilisation la communication sans fil dans les réseaux sur puce, l'exemple de l'architecture Hybride (Rezaei et al., 2015)qui est utilisée une architecture hybride : au lieu d'utiliser toutes les liaisons sans fil pour transférer des données entre différents nœuds de l'architecture, la conception hybride utilise des liaisons filaires et sans fil à la fois pour construire les réseaux sur puce hybride. Le mapping dans les WNocs peut avoir des étapes différentes comme le mapping des IPs ou les routeurs sans fil sur l'architecture du WNoc.

Afin de choisir l'algorithme de mapping dans notre projet, nous présentons une comparaison entre les algorithmes utilisés dans la littérature, qui est illustrée par le tableau 3-1.

Tableau 3-1 : une comparaison entre les algorithmes utilisés dans le mapping.

Les références	***Architecture Noc***	***Algorithme***	***Objectif***	***Résultat***
(Sikandar et al., 2021)	2D-Mesh	Une méta-heuristique base sur l'algorithme d'optimisation Sailfish	Minimiser le temps de calcul, la consommation d'énergie et le coût de communication	Une amélioration moyenne de la minimisation de l'énergie de la topologie, le temps de calcul et la latence moyenne de réseau par rapport à d'autres algorithmes comme Bat, Aco et ILP.

(Fang et al., 2019)	2D-Mesh	L'algorithme Colonie de Fourmies (Ant Colony Algorithm)	Réduire la consommation d'énergie et la latence	La méthode proposée est très efficace pour réduire la puissance de communication et la latence du réseau lors du mapping NoC
(Alagarsamy & Gopalakrishnan, 2018)	2D-Mesh	Méta-heuristique de chauve-souris modifiée (MBA)	Réduire consommation d'énergie du réseau et le coût de communication	Une minimisation des coûts de communication et d'énergie
(Nalci et al., 2021)	3D-Mesh	Un algorithme heuristique CastNet3D et la formulation ILP	Minimiser la consommation d'énergie	L'algorithme CastNet3D obtient des résultats proches de l'optimal
(He et al., 2018)	3D-Mesh	L'algorithme génétique (GA) et recuit simulé (SA)	Réduire la consommation d'énergie	Les résultats montrent que par rapport à la GA, ISAGA a une bonne convergence et peut rechercher rapidement la solution optimale, ce qui peut réduire efficacement la consommation d'énergie du système
(Dageleh & Jamali, 2018)	3D-Mesh	Un algorithme heuristique V-CastNet3D	Réduire le délai et la consommation d'énergie du réseau, le coût de communication	L'algorithme proposé a indiqué que l'algorithme de mappage V-CastNet était capable de réduire le délai et la consommation d'énergie du réseau mieux que d'autres algorithmes associés
(Alagarsamy et al., 2019)	3D-Mesh	D'optimisation des essaims de poulets (scso)	Réduire le coût de communication	Une minimisation des coûts de communication et l'd'énergie
(Rezaei et al., 2017)	WNoc	Round Rotary Mapping (RRM)	/	Les résultats de la simulation montrent une amélioration significative à la fois de la congestion et du

				contrôle de la température du système
(Rezaei et al., 2015)	WNoc	Mapping dynamique des applications (DAMA)	Réduire la congestion interne et externe	Le DAMA permet de réduire la probabilité de congestion interne et externe
(Sacanamboy-Franco et al., 2017)	WNoc	Algorithme génétique (GA)	Minimiser la consommation d'énergie, la bande passante et l'accélération.	L'algorithme proposé présente de bonnes performances (consommation d'énergie, la bande passante et l'accélération.)

3.7 Conclusion

Le mapping est considéré comme la phase la plus importante dans la conception des réseaux sur puce, qui sert à placer les tâches d'application sur une architecture Noc, cette phase peut réduire les performances du système.

En effet dans ce chapitre nous avons défini le problème de mapping et ses types et également détaillé des différents algorithmes de méta heuristique utilise dans le mapping.

Dans l'objectif de maximiser (optimiser) les performances du système pour réaliser la phase de mapping, plusieurs approches sont introduites comme les techniques d'optimisation Métaheuristique inspirée de la nature. Dans le chapitre suivant, nous allons présenter en détail l'adaptation et l'implémentation de l'algorithme GEO dans le problème de Mapping.

Chapitre 4 : Solution proposée

4.1 Introduction

Le réseau sur puce Noc est considéré comme un modèle pour établir la communication au sein de systèmes implémentés sur une puce (Soc). Le développement et les progrès continus de l'informatique soulignent l'importance du Noc. Dans la conception du réseau sur puce la phase de mapping aura un impact significatif sur l'efficacité de la puce et sa consommation d'énergie (Dageleh & Jamali, 2018), donc cette phase nécessite des outils adaptés pour résoudre, parmi les algorithmes actuels les plus utilisé sous les algorithmes Méta-heuristique inspiré de la nature bases sur l'intelligence en essaim tel que les chats, les chauves-souris, les fourmis…etc.

L'essaim des Golden Eagles (GE) est l'objet de notre étude, en mettant l'accent sur l'algorithme des Golden Eagles dans ce chapitre, nous allons essayer de l'adapter à résoudre notre problème du mapping.

4.2 Formulation du problème

Le problème du mapping (placement) d'une application sur un réseau sur puce peut définir comme une fonction appelle "***Map***", qui implique de placer chaque tâche $\boldsymbol{v_i}$ du graphe d'application (**GAP**) à un et un seul processeur (tuile) $\boldsymbol{t_i}$ du graphe d'architecture (**GAR**) (Delorme & Houzet, 2006; Gan et al., 2021).

$$Map : \begin{cases} \boldsymbol{V \rightarrow T} \\ \boldsymbol{|V| \leq |T|} \\ \boldsymbol{v_i \rightarrow t_i} \end{cases} \qquad \textbf{(4-1)}$$ (Alagarsamy & Gopalakrishnan, 2016).

Avec

$\boldsymbol{v_i}$: Une tâche de l'application et $\boldsymbol{t_i}$: une tuile de l'architecture Noc.

4.3 Définition de notre fonction objective

Notre but consiste à minimiser le coût de communication entre les différentes tâches d'une application (**GAP**) placé sur une architecture réseau sur puce 2D, 3D et WiNoc. L'architecture est choisie par l'utilisateur.

$$\boldsymbol{f_{objective}(GAP) = Coût\ Com} \qquad \textbf{(4-2)}$$

$$\boldsymbol{Coût\ de\ Com} = \sum_{K=1}^{|E|}(\boldsymbol{vl(C^k)}) \times \boldsymbol{nombre\ de\ saut}\left(\boldsymbol{source(C^k)}, \left(\boldsymbol{destination(C^k)}\right)\right) \qquad \textbf{(4-3)}$$

(Janidarmian & Fekr, 2012).

Où :

$\boldsymbol{C^k}$ Représente un arc dans le graphe d'application.

$\boldsymbol{vl(C^k)}$ Représente la valeur qui indique le volume de communication pour chaque arc (entre 2 tâches).

$\boldsymbol{source(C^k)}$ La tuile source.

$\boldsymbol{destination(C^k)}$ La tuile destination.

$\boldsymbol{nombre\ de\ saut}$ $(\boldsymbol{source(C^k)}, \left(\boldsymbol{destination(C^k)}\right)$ Représente la distance entre la tuile source et la destination.

4.3.1 Architecture 2D et 3D

Dans le cas d'une architecture Noc 2D ou 3D le nombre de sauts sera calculé par la distance de Manhattan.

$$\boldsymbol{nombre\ de\ saut(i,j)} = |x_i - x_j| + |y_i - y_j| \qquad \textbf{(4-4)}$$

(Fang et al., 2019)

$$\boldsymbol{nombre\ de\ saut}(\boldsymbol{i},\boldsymbol{j}) = |\boldsymbol{x_i} - \boldsymbol{x_j}| + |\boldsymbol{y_i} - \boldsymbol{y_j}| + |\boldsymbol{z_i} - \boldsymbol{z_j}| \quad \textbf{(4-5)}$$ (Gan et al., 2021)

Où :

$\boldsymbol{x_i}, \boldsymbol{x_j}$ Désigne la ligne de la position des tuiles ***i*** et ***j***.

$\boldsymbol{y_i}, \boldsymbol{y_j}$ Désigne la colonne de la position des tuiles ***i*** et ***j***.

$\boldsymbol{z_i}, \boldsymbol{z_j}$ Désigne le niveau de la position des tuiles ***i*** et ***j***.

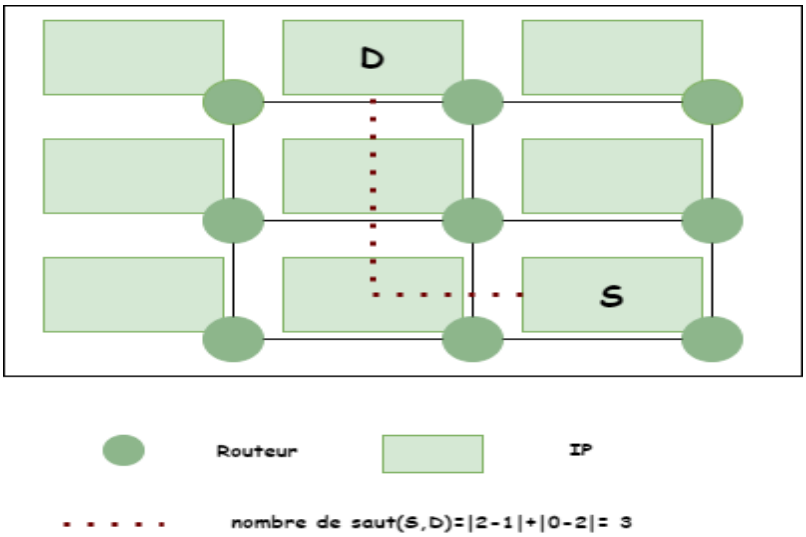


Figure 4-1: Exemple de calcul du nombre de sauts dans une topologie Noc 2D 3×3.

4.3.2 Architecture WNoC

L'architecture WiNoc se compose des réseaux filaires et sans fil, de sorte que les transferts de paquets peuvent se produire sur des liaisons filaires, des liaisons sans fil ou une combinaison des deux (Hu et al., 2012). Afin d'atteindre notre objectif de minimiser le coût de communication entre les tâches d'une application mappées sur l'architecture WiNoc, nous devons choisir la transmission optimale des paquets, c'est-à-dire le plus petit chemin entre la source et la destination dans ce cas la distance est basée sur la localisation de la source et la destination.

- ***Le même sous réseau***

Le même sous réseau est considéré comme une architecture Noc 2D alors le nombre de sauts est calculé par la distance de Manhattan.

$$\boldsymbol{nombre\ de\ saut}(i,j) = |x_i - x_j| + |y_i - y_j| \qquad \textbf{(4-6)}$$ (Hu et al., 2012).

Où :

x_i, x_j Désigne la ligne de la position des tuiles ***i*** et ***j*** dans le même sous-réseau.

y_i, y_j Désigne la colonne de la position des tuiles ***i*** et ***j*** dans le même sous-réseau.

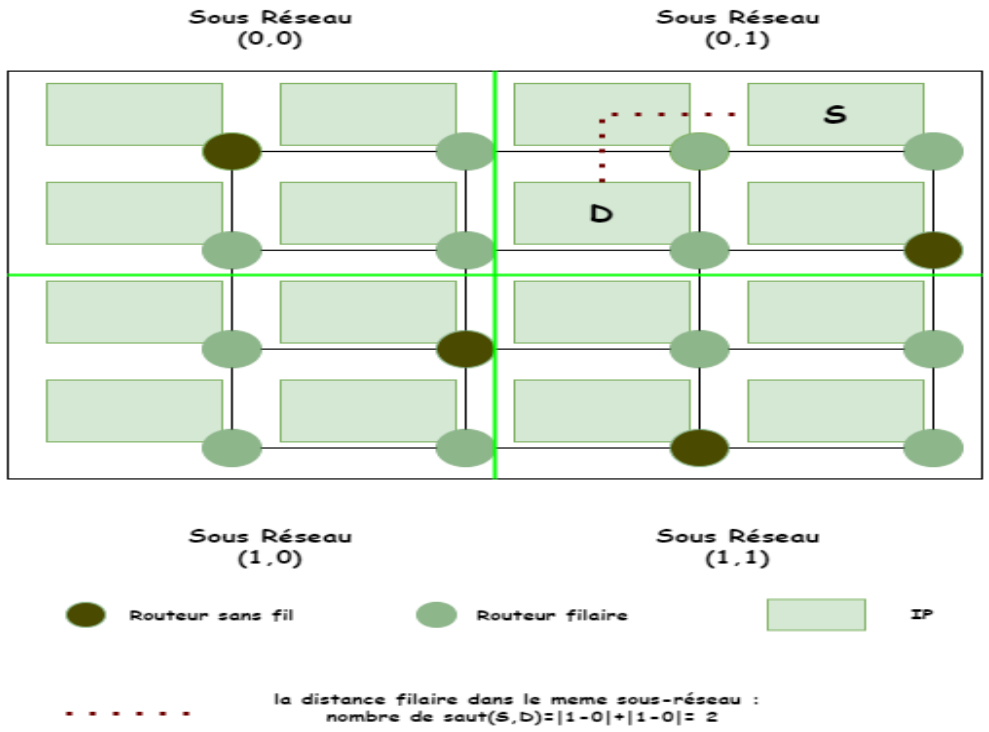


Figure 4-2: Exemple du calcule de la distance dans le même sous-réseau sur une topologie WiNoc.

- ***Sous réseau différent***

Si la communication entre la tuile source et la tuile de destination dans des sous-réseaux différents, le nombre des sauts est calculé par deux méthodes. Ensuite nous choisissons le meilleur (le minimum entre les deux).

- ***Communication filaire***

Dans ce cas la communication est considérée comme le nombre de sauts, elle est calculée par la distance de Manhattan.

$$\boldsymbol{nombre\ de\ saut}(i,j) = |x_i - x_j| + |y_i - y_j| \qquad \textbf{(4-7)}$$ (Hu et al., 2012).

Où :

x_i, x_j Désigne la ligne de la position des tuiles ***i*** et ***j*** dans le réseau total.

y_i, y_j Désigne la colonne de la position des tuiles ***i*** et ***j*** dans le réseau total.

- ***Communication sans fil (Wireless)***

Le nombre de sauts dans la communication sans fil est la somme de trois distances (Hu et al., 2012).

- Entre la tuile source et le routeur sans fil source $|x_i - x_{RW\,i}| + |y_i - y_{RW\,i}|$.
- Entre les deux RW source et destination $\sqrt{(x_{RW\,i} - x_{RW\,j})^2 + (y_{RW\,i} - y_{RW\,j})^2}$.

(La distance euclidienne entre les routeurs WR source et destination dans le réseau total)

- Entre le routeur sans fil destination de la tuile destination $|x_{RW\,j} - x_j| + |y_{RW\,j} - y_j|$.

$$\boldsymbol{nombre\ de\ saut(i,j)} = |x_i - x_{RW\,i}| + |y_i - y_{RW\,i}|$$

$$+\sqrt{(\boldsymbol{x_{RW\,i\,t} - x_{RW\,j\,t}})^2 + (\boldsymbol{y_{RW\,i\,t} - y_{RW\,j\,t}})^2} \quad + |\boldsymbol{x_{RW\,j} - x_j}| + |\boldsymbol{y_{RW\,j} - y_j}| \qquad \textbf{(4-8)}$$

Nous pouvons résumer le calcule de nombre de sauts entre les différentes IPS qui communique dans une architecture WNoC par un algorithme « décision de chemin ».

Algorithme : Décision de chemin

Entrées : *S* //la source de la communication.

***D* //la destination de la communication.**

Sortie : Nb_{saut} // le nombre de sauts entre la source et la destination.

- **Début**
  - **Si S et D sont dans le même sous-réseau alors**
    - **Calculer le nombre de saut Nb_{saut} salon (6).**
  - **Sinon**
    - **Calculer le nombre de saut Nb_{saut1} salon (7).**
    - **Calculer le nombre de saut Nb_{saut2} salon (8).**
    - **Choisie Nb_{saut} le minimum entre Nb_{saut1} et Nb_{saut1}**
  - **Finsi**
- **Fin**

La figure 4-3 est un exemple qui montre comment calculer la distance dans des sous-réseaux différents sur une topologie WiNoc.

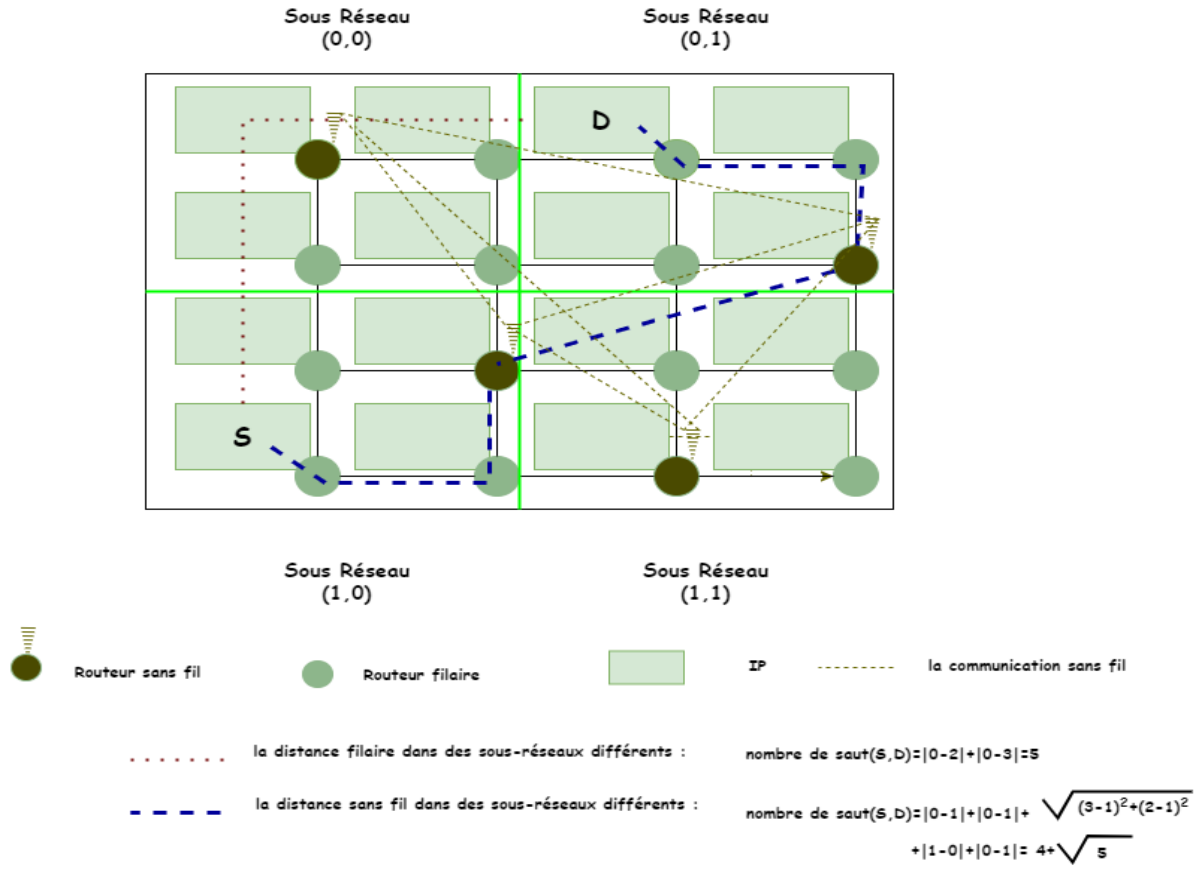


Figure 4-3 : Exemple de calcul de la distance dans des sous-réseaux différents sur une topologie WiNoc.

4.4 Représentations des solutions

Nous représentons une solution de mapping d'une application sur une architecture 2D ou bien 3D comme un tableau $\boldsymbol{T_{map}}$ tel que la taille ($\boldsymbol{T_{map}}$)= la taille (**GAP**), où l'indice du tableau représente une tâche d'application et la valeur de la case représente un processeur de l'architecture Noc où sont affectées les tâches. La figure 4 un exemple du représentons d'une solution.

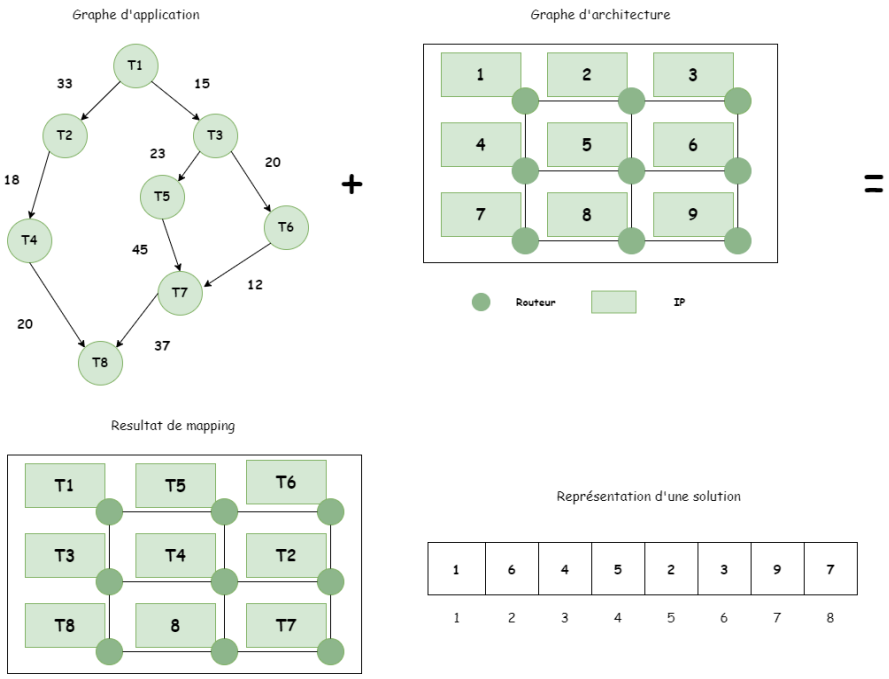


Figure 4-4: un exemple de la représentation d'une solution.

Dans le cas WiNoc une solution de mapping est représentée sous forme de deux tableaux :

$\boldsymbol{T1_{map\ ip}}$ Représente une solution de mapping d'une application sur une architecture WiNoc, où l'indice du tableau représente une tâche d'application et la valeur de la case représente une tuile de l'architecture Noc où est affectée la tâche.

$T2_{map\ wr}$ Représente une solution de mapping des routeurs sans fil sur une architecture WiNoc, où l'indice du tableau représente un routeur sans fil et la valeur de la case signifiée la position de routeur sans fil dans l'architecture WiNoc.

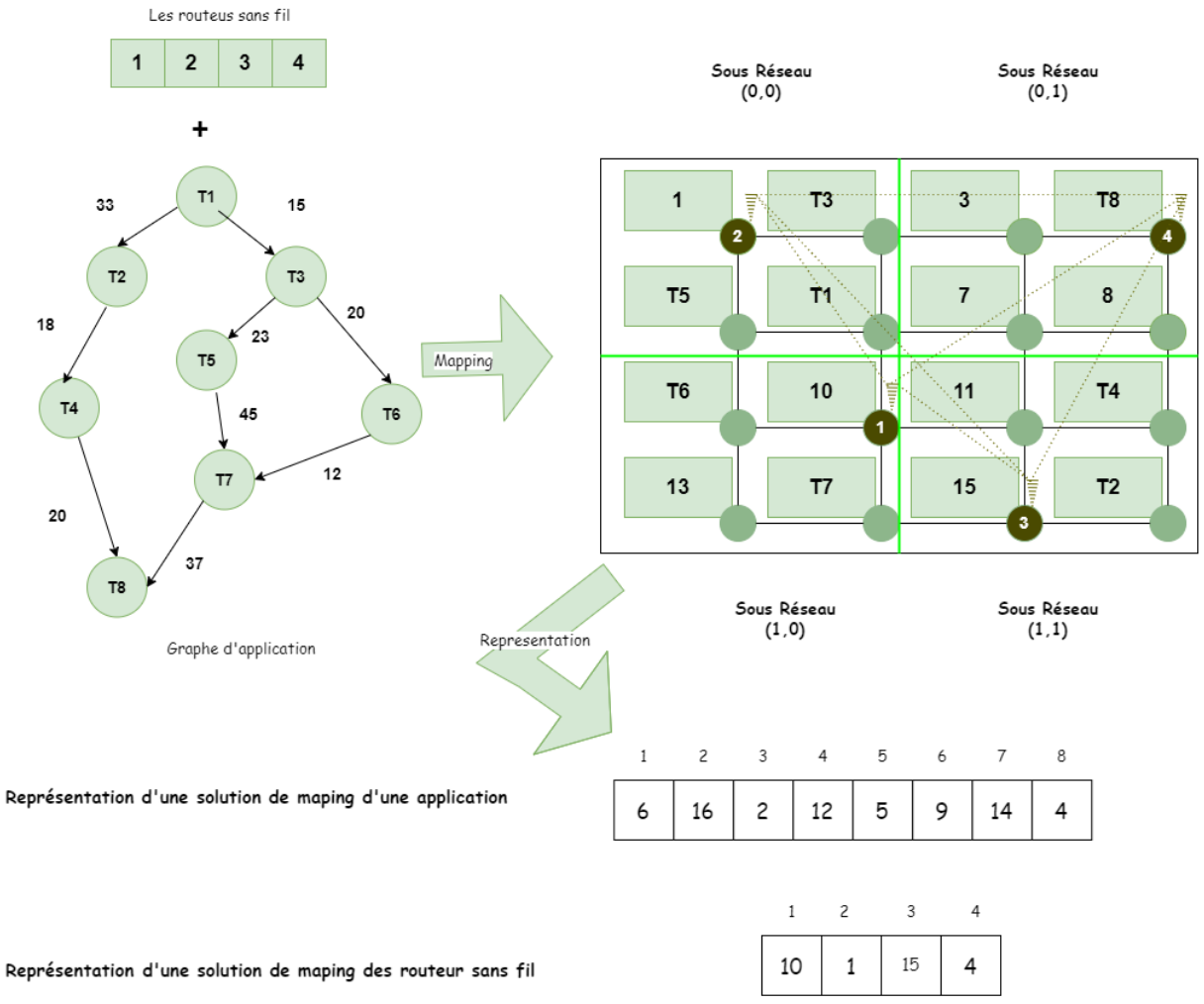
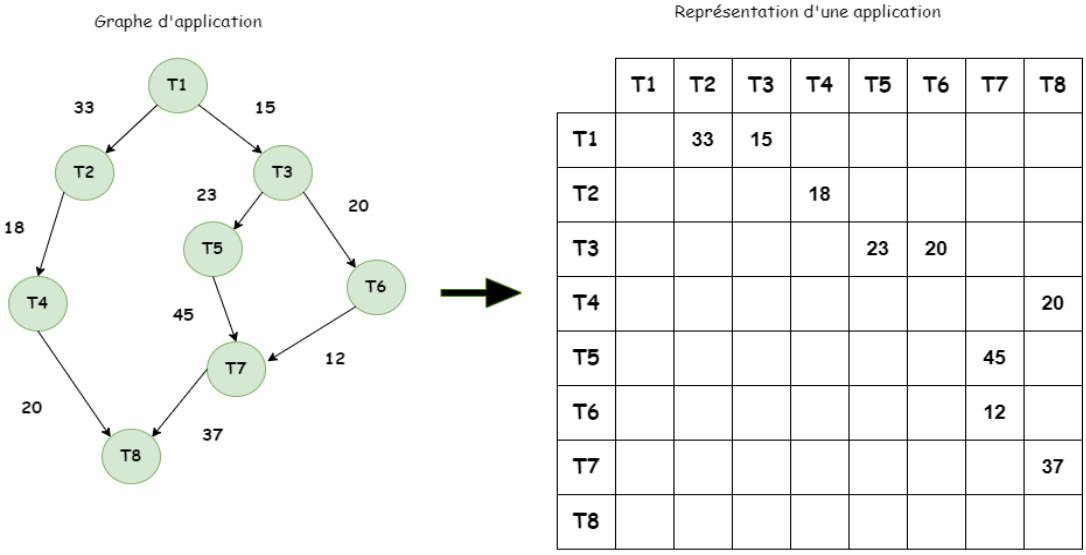


Figure 4-5: Un exemple de la représentation d'une solution.

4.5 Représentation de graphe d'application

Dans notre mémoire, nous allons représenter un graphe d'application sous forme d'une matrice carrée de taille $n \times n$, où n est le nombre de tâches du graphe d'application (GAP).

Les indices de la ligne i et de la colonne j représentent IP, et les cases de la matrice correspondant aux deux indices représentent les poids $w_{i,j}$.



	T1	T2	T3	T4	T5	T6	T7	T8
T1		33	15					
T2				18				
T3					23	20		
T4								20
T5							45	
T6							12	
T7								37
T8								

Figure 4-6: Représentation du graphe d'application.

4.6 Le processus de la solution

L'Algorithme GEO appliqué au problème de mapping est résumé dans la figure 4.7, nous pouvons diviser le processus de solution en trois étapes :

Les entrées du problème.

Traitement de population.

Traitement de Golden Eagle.

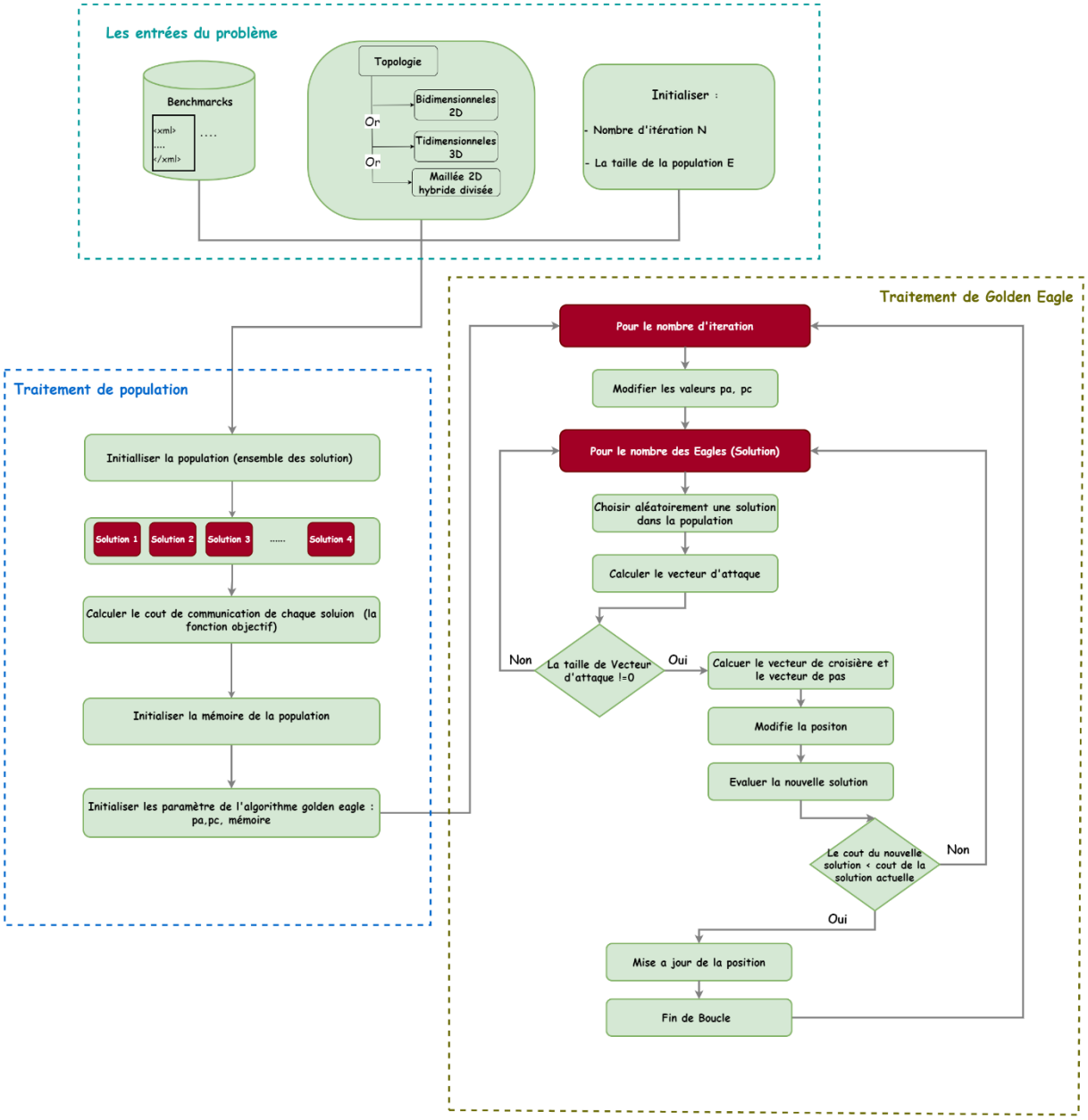


Figure 4-7: Processus de la solution.

4.6.1 Étape 1{Les entrées du problème}

Cette étape permet à initialiser les paramétrés de programme tel que, la taille de la population, le nombre d'itérations, la topologie (Noc 2D, Noc 3D ou WNoC 2D) et le graphe d'application utilisé.

4.6.2 Étape 2 {Traitement de population}

❖ ***Génération d'une population***

La population initiale de GE est générée aléatoirement de taille $\boldsymbol{E}$. Elle consiste à créer un nombre $\boldsymbol{E}$ des solutions, ce nombre est un paramètre choisi par l'utilisateur.

On crée :

Un vecteur $\boldsymbol{T_{Pos\_act}}$ qui représente la position actuelle de chaque eagle $\boldsymbol{i}$, la taille de vecteur est égale le nombre des tâches dans l'application choisie.

Un vecteur $\boldsymbol{T_{Pos\_best}}$ qui représente la meilleure solution locale occupée jusqu'à présent pour chaque eagle $\boldsymbol{i}$, la taille de vecteur est égale le nombre des tâches dans l'application choisie.

Une valeur $\boldsymbol{C_{act}}$ Contient le coût de communication locale eagle $\boldsymbol{i}$.

Une valeur $\boldsymbol{C_{best}}$ Contient le meilleur coût de communication locale eagle $\boldsymbol{i}$.

L'algorithme de génération d'une population « GE » : crée la population initiale aléatoirement de taille $\boldsymbol{E}$.

Algorithme : Génération d'une population

Entrées : $\boldsymbol{M_{app}}$ //une matrice qui représente le graphe d'application.

$\boldsymbol{E}$ // la taille de la population.

$\boldsymbol{S}$ // la taille de Noc.

Sortie : $\boldsymbol{T_{POP}}$ // un vecteur de la population.

- **Tant que $\boldsymbol{i < E}$**

- **Générer aléatoirement une solution (vecteur)** T_{map} **selon la fonction Map (1)**
- **Évaluer le coût de communication** $Cost$ **selon (3)**
- **Initialiser la mémoire de la population** T_{Pos_act}, T_{Pos_best} , C_{act} et C_{best}

// Tel que $T_{Pos_{act}} = T_{Pos\_best} = T_{map}$ **et** $C_{act} = C_{best} = Cost$

- **Insérer dans le vecteur de la population** T_{POP}

- **Fin Tant que**

❖ ***Calculer le coût de communication***

Pour chaque solution générée, nous calculons son coût de communication selon l'algorithme de coût de communication.

Algorithme : coût de communication

Entrées : T_{map} **//un vecteur qui représente une solution de mapping.**

Sortie : $Cost$ **//une valeur de coût de communication calcule.**

- **Pour chaque IP du vecteur** T_{map}
  - **Repérer les autres IP avec qu'il communique**
- **Pour chaque IP repéré qui communique avec l'IP courant**
  - **Calculer le nombre de sauts entre les deux dans l'architecture**
  - **Actualiser la valeur du coût de communication** $Cost$ **selon (3)**

- **Fin pour**
- **Fin pour**

4.6.3 Etape 3 {Traitement de Golden Eagle}

Après la génération d'un ensemble des solutions et calculer le coût de communication pour chaque solution, on calcule le vecteur d'attaque.

❖ **Vecteur d'attaque**

Chaque solution dans la population a un aigle choisi à l'aléatoire que nous traitons comme une proie. Pour déterminer le vecteur d'attaque, nous soustrayons la position actuelle de l'aigle $\boldsymbol{T_{Pos\_act}}$ de la meilleure position de la proie $\boldsymbol{T_{Pos\_best}}$.et voici un exemple comment calculer le vecteur d'attaque.

Positon actuelle d'aigle T_{Pos_act}

11	**7**	**9**	**8**	**12**	**5**

Meilleure position de proie T_{Pos_best}

6	**0**	**4**	**10**	**9**	**15**

Vecteur d'attaque A_i

5	**7**	**5**	**2**	**3**	**10**

❖ **Vecteur de croisière**

Le vecteur de croisière $\boldsymbol{C_i}$ est calculé en fonction du vecteur d'attaque, Il est un vecteur tangent en cercle et perpendiculaire. Ce vecteur de taille **n** contient des valeurs aléatoires entre 0 et 1 sauf dans **k** -ème élément (c'est une valeur aléatoire entre 0 et n-1).

$$C_k = \frac{\sum_{i=0}^{n-1} a_i x_i - \sum_{i,i\neq k} a_i}{a_k} \qquad (4\text{-}9)$$

Voici un exemple pour vecteur de croisière calculer à base de vecteur d'attaque A_i .

$$C_6 = \frac{(6\times 5 + 0\times 7 + 4\times 5 + 10\times 2 + 9\times 3 + 15\times 10) - (5+7+5+2+3)}{10} = 20.5$$

Vecteur de croisière C_i

0.78	**0.65**	**0.88**	**0.39**	**0.57**	**20.5**

❖ **Vecteur de pas**

Ce vecteur permit de déplacer la position d'aigle vers un autre à la base de vecteur d'attaque A_i et vecteur de croisière C_i selon l'équation **(2-4)**.

Pour les paramètres d'algorithme on prend par exemple

r1=0.08 r2 = 0.2 $p_a = 0.9$ $p_c = 0.6$

$\overrightarrow{r_1} p_a \frac{\overrightarrow{A_i}}{\|\overrightarrow{A_i}\|}$

0.024	**0.034**	**0.024**	**0.00989**	**0.014**	**0.049**

$\overrightarrow{r_2} p_c \frac{\overrightarrow{C_i}}{\|\overrightarrow{C_i}\|}$

0.00455	**0.00379**	**0.00513**	**0.00227**	**0.00332**	**0.12**

Vecteur de pas Δx_i

Comme nous l'avons vu, la somme des deux vecteurs précédents, qui fournit des valeurs entre 0 et 1, ne change pas la position de l'aigle. C'est pourquoi nous multiplions le résultat par 100.

2.85	**3.77**	**2.91**	**1.21**	**1.73**	**16.9**

❖ L'algorithme d'optimisation de Golden Eagle

L'algorithme GEO est expliqué dans le pseudo-code Optimisation Golden Eagle.

Algorithme : Optimisation Golden Eagle

Initialiser la population des aigles royaux

Évaluer le coût de communication

Initialiser la mémoire de la population

Initialiser p_a **et** p_c

- **Pour chaque itération** *t*
  - **Mettre à jour** p_a **et** p_c

- **Pour chaque golden eagle** *i*
  - **Sélectionner au hasard une proie dans la population**
  - **Calculer le vecteur d'attaque** $\vec{A}$ **selon (3)**
  - **Si** $\overrightarrow{A_i} \neq 0$
    - ❖ **Calculer vecteur de croisière** $\vec{C}$
    - ❖ **Calculer vecteur de pas** $\overrightarrow{\Delta_x}$
    - ❖ **Mettre à jour la position**
    - ❖ **Évaluer le coût de communication pour la nouvelle position**
    - ❖ **Si le coût de communication calculée est meilleur que celui dans la mémoire de l'aigle** *i*
      - ✓ **Remplacer la position qui est dans la mémoire de l'aigle** *i* **par la nouvelle position.**
    - ❖ **Fin si**
  - **Fin si**
- **Fin pour**
- **Fin pour**

4.7 Conclusion

Ce chapitre a montré la manière que nous avons adaptée à l'algorithme Golden Eagle Optimization (GEO) que nous avons utilisés par la suite pour résoudre le problème de mapping.

Pour tester notre solution, des graphes d'application ont été créés à l'aide de TGFF, qui est la plus utilisée dans la littérature. Dans le chapitre suivant, nous allons présenter l'ensemble des graphes d'application, les outils de développement, les tests ainsi que les résultats obtenus.

Chapitre 5 : Tests et résultats

5.1 Introduction

Comme nous avons mentionné précédemment, l'idée principale de ce travail est de résoudre le problème de placement d'une application (Mapping) sur le réseau sur puce pour le but de réduire le coût de communication. Plusieurs métaheuristiques basées sur l'intelligence en essaim sont introduites, notamment celles inspirées par les comportements sociaux (bactéries, abeilles, chauves-souris, poissons, etc.) qui ont évolué dans des essaims d'animaux et d'insectes. La méta-heuristique base sur l'intelligence en essaim des aigles royaux fait l'objet de nos recherches. Le problème du mapping nécessite des outils adaptés à résoudre, et dans ce chapitre, nous commençons par introduire des outils tels que l'environnement et différents benchmarks utilisés dans ce travail, puis présentons les résultats obtenus en les comparants avec d'autres travaux.

Nous avons choisi d'utiliser NetBeans pour développer notre application sous le système d'exploitation Windows, et la plate-forme de test utilisée a été conçue en Java.

5.2 Outils d'implémentation utilisés

5.2.1 Caractéristiques de machine

Les tests sur le problème du mapping proposés sont réalisés sur une machine équipée : Processeur Intel (R) Core (TM) i5, 2.40 GHZ, et un RAM de 8Go sous le système d'exploitation Windows 10 Pro 64 bit.

5.2.2 Langage de programmation Java

Java est un langage de programmation multiplateforme créé en 1995(*Java*, n.d.) par James Gosling chez Sun Microsystems (qui apparie aujourd'hui à l'entreprise Oracle). Une des particularités principales qui différencie le langage Java des autres langages comme le C ou le C++ est la manière dont il est exécuté et compilé sur une machine (*[Guide Ultime] Apprenez La Programmation Java Par La Pratique*, n.d.).

Le compilateur Java, nommé javac, ne traduit pas directement le code source en langage machine comme les autres compilateurs. Il le traduit en un langage intermédiaire appelé bytecode. Ce bytecode est ensuite interprété par un autre programme : la machine virtuelle java ou JVM (pour Java Virtual Machine en anglais). Ainsi, un programme java sera toujours représenté par deux types de fichiers : les fichiers sources en java (extension. Java) et le résultat de leur compilation en byte code (fichier d'extension .class).La JVM est dépendante de la plate-forme, c'est-à-dire que sa mise en œuvre diffère d'une plate-forme à l'autre (comme Windows, Linux, Mac, etc.…), mais toutes les JVM peuvent exécuter le même bytecode java : c'est l'approche « write once and run anywhere », en français « exécuter une fois et écrire partout » (*[Guide Ultime] Apprenez La Programmation Java Par La Pratique*, n.d.).

Grâce à ce système, Java peut être développé sur n'importe quel appareil, n'importe quel système d'exploitation (OS), compilé en un bytecode standard et fonctionner sur n'importe quel appareil équipé d'une JVM.

5.3 Environnement de développement NetBeans

NetBeans est un environnement de développement intégré (EDI), placé en open source par Sun en juin 2000 (Famenne, 2012), sous licence CDDL et GPLv2 (Common Development and Distribution License).

NetBeans est un IDE populaire et open source écrit en Java. Il s'agit d'un logiciel multiplateforme et peut fonctionner sur n'importe quel système d'exploitation. NetBeans permet également aux développeurs tiers d'améliorer ses fonctionnalités. Il est livré avec des fonctionnalités telles qu'un système de projet basé sur Ant, une refactorisation et un contrôle de versions. Parallèlement à cela, il peut également fonctionner avec des applications cloud.

NetBeans est devenu une partie du projet Apache et a gagné en popularité. Cependant, il prend en charge d'autres langages tels que PHP, C/C++ et HTML5(*6 Best Open Source IDEs for Java Programming Language*, 2019).

Figure 5-1: Environnement de développement NetBeans.

5.4 Présentation des benchmarks (graphes utilisés)

Les benchmarks sont des graphes de tâches qui sont utilisés pour effectuer les tests, le graphique de tâche pour le logiciel libre - TGFF a été utilisé pour créer l'ensemble d'applications à tester.

5.4.1 Benchmark aléatoire

TGFF a été développé à l'origine en 1998(Vallerio, 2008) pour fournir une référence randomisée standardisée pour la recherche sur la planification et l'allocation, TGFF convient à de nombreuses applications nécessitant la génération de graphes pseudo-aléatoires, d'usage général et contrôlable par l'utilisateur (Vallerio, 2008).

Le paramètre principal pour la création de graphiques de test est le nombre desIPs (le nombre des tâches dans l'application), pour la bande passante de communication entre différentes IP, elle était aléatoire et comprise entre 1000 et 1400 bits de variation, ces connexions sont celles que l'on cherche à réaliser au moindre coût de communication.

TG1, TG2, TG3, TG4, TG5, TG6, TG7, TG8, TG9, TG10 sont dix graphiques créés par la TGFF, pour effectuer les tests de notre problème.

Figure 5-2: Code QR contient les Benchmarks aléatoires utilisés dans les tests .

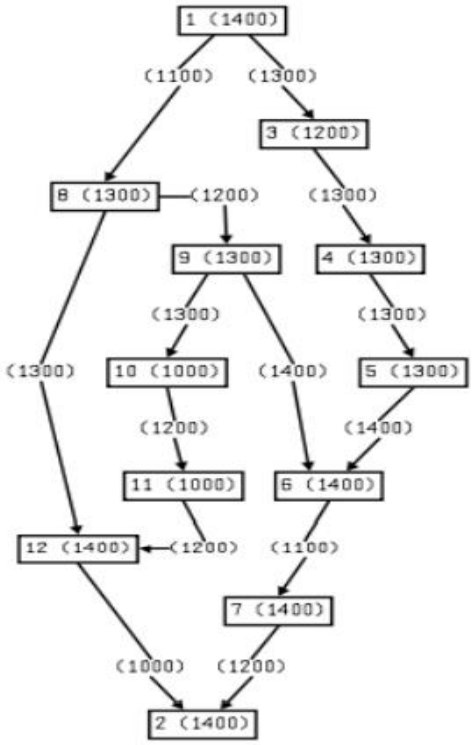


Figure 5-3: Benchmark TG10.

5.4.2 Les benchmarks standard

Dans notre expérimentation on a utilisé 2 benchmarks (Task Graph) standards qui sont VOPD, MWD.

- **Benchmark VAPD**

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

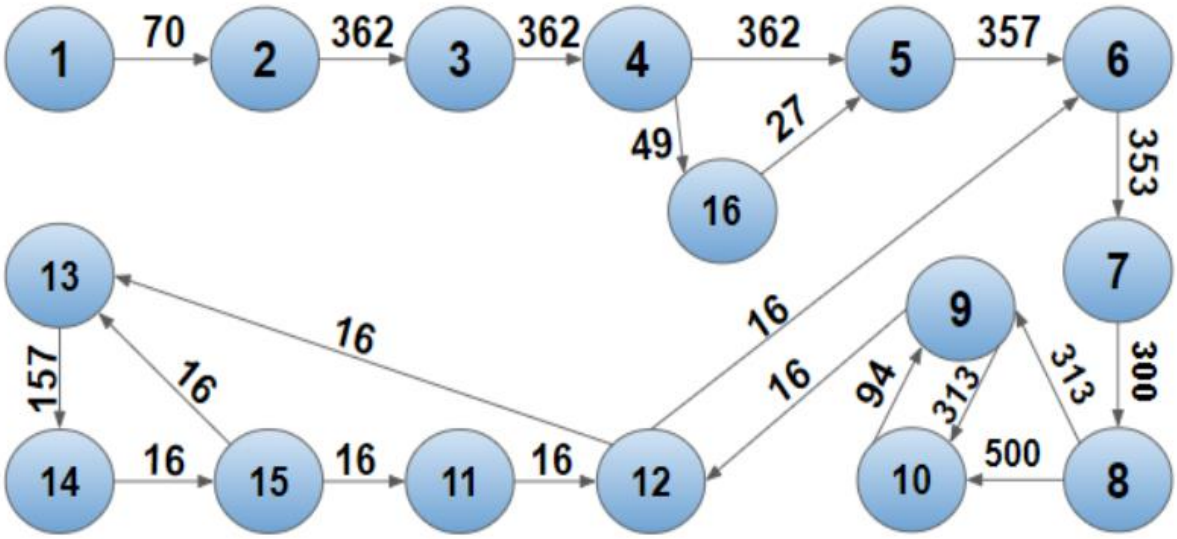


Figure 5-4 : Benchmark VOPD.

- **Benchmark MWD**

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

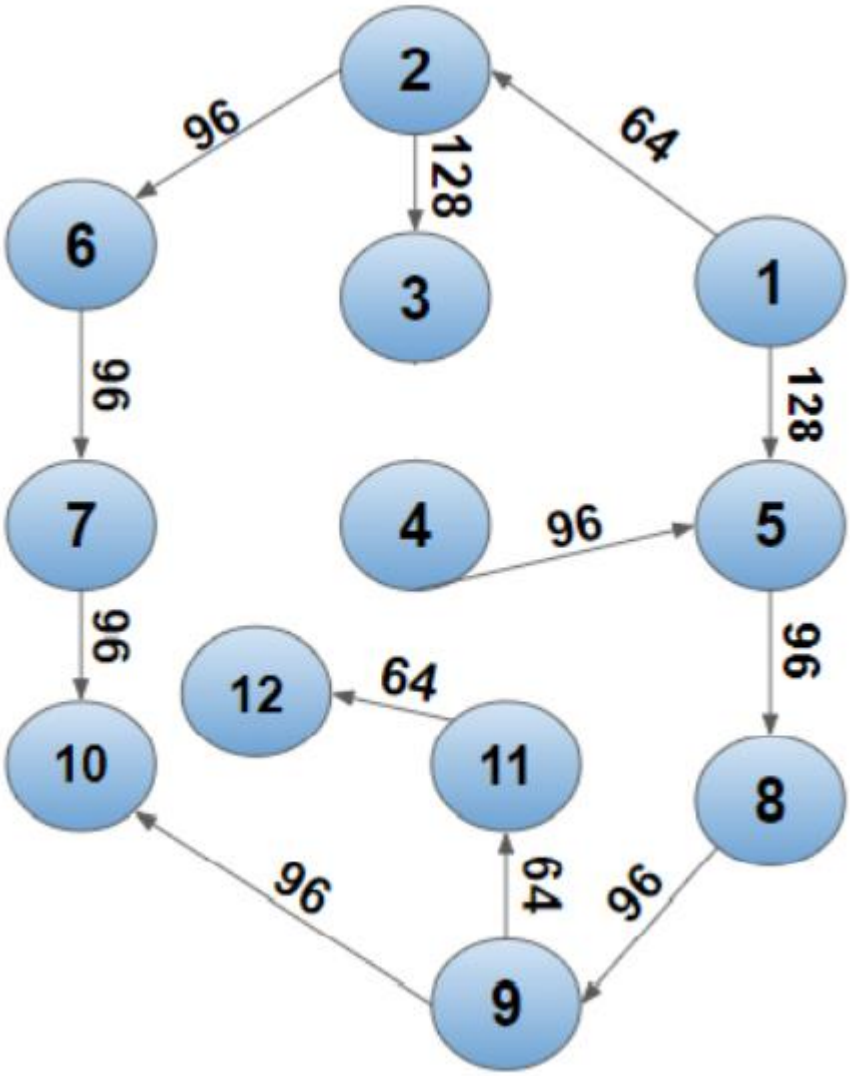


Figure 5-5 : Benchmark MWD.

5.5 Réalisation des tests

5.5.1 Paramétrage

Pour tester l'efficacité de notre algorithme proposé, nous effectuons différents tests sur les paramètres, les paramètres architecturaux et les paramètres de l'algorithme proposé.

Comme nous l'avons vu dans les deux chapitres précédents, l'un des facteurs qui doit être prioritaire pour que l'algorithme aigle royal donne des résultats efficaces sont : propension

à attaquer Pa = [0.5,2], propension à la croisière Pc= [1,0.5] et 2 valeurs fixées aléatoires entre 0 et 1 sont r1, r2 durant exécution.

Nous avons utilisé trios architectures sont Noc 2D, Noc 3D et WNoC 2D.

5.5.2 Réalisation des tests

Nous avons testé les trois architectures en faisant varier un ensemble de caractéristiques, et nous avons comparé les résultats d'autres algorithmes que nous avons testés par rapport au même benchmark, avec les résultats suivants :

5.5.2.1 Réalisation des tests dans Noc 2D

- ***Changement de la taille de population***

Nous avons varié la taille de la population initiale tout en gardant le même nombre d'itérations à 1000. Les résultats obtenus sont les suivants :

Tableau 5-1 : Résultats obtenus en changeant la taille de population initiale dans Noc 2D.

Graphe	*Topologie*	*Itération*	*Population*	*Noc 2D*
TG1	**10*10**	**1000**	10	782500
			50	775300
			100	762200
			200	757900
			500	732600
			1000	726100
			1500	725100

TG2	**10*10**	**1000**	10	762500
			50	740200
			100	703600
			200	693600
			500	689100
			1000	668600
			1500	658900
TG3	**11*11**	**1000**	10	955200
			50	967200
			100	941100
			200	939500
			500	886600
			1000	902800
			1500	895300
			10	620000
			50	574900
			100	579200

TG4	**10*10**	**1000**	200	540600
			500	566800
			1000	554200
			1500	548800
TG5	**8*8**	**1000**	10	432500
			50	413100
			100	415800
			200	401500
			500	391900
			1000	391800
			1500	389300
TG6	**8*8**	**1000**	10	325700
			50	328500
			100	294900
			200	306600
			500	308400
			1000	301700

			1500	289400
TG7	**8*8**	**1000**	10	188100
			50	172300
			100	174400
			200	163300
			500	168400
			1000	167500
			1500	152900
TG8	**6*6**	**1000**	10	116300
			50	109800
			100	107700
			200	112500
			500	102600
			1000	102100
			1500	101400
			10	63000
			50	60700

TG9	**6*6**	**1000**	100	60400
			200	56400
			500	52900
			1000	52900
			1500	48300
TG10	**6*6**	**1000**	10	47000
			50	43600
			100	42100
			200	39900
			500	39700
			1000	36300
			1500	37700

Nous remarquons dans le tableau ci-dessus que les résultats sont améliorés lorsque la taille de la population est grande, et que la plupart des meilleurs résultats obtenus ont été trouvés dans des populations de taille 1500.

- ***Changement de nombre d'itérations***

Nous comparons maintenant les résultats tout en variant sur le nombre d'itérations, en gardant la même taille de population 1000 individus.

Tableau 5-2 : Résultats obtenus en changeant la taille d'itération dans Noc 2D.

Graphe	*Topologie*	*Population*	*Itération*	*Noc 2D*
TG1	**10*10**	**1000**	2000	730900
			4000	726800
			5000	730600
TG2	**10*10**	**1000**	2000	655400
			4000	678900
			5000	693600
TG3	**11*11**	**1000**	2000	899700
			4000	904400
			5000	921100
TG4	**10*10**	**1000**	2000	548800
			4000	554300
			5000	548100
TG5	**8*8**	**1000**	2000	404100
			4000	406600
			5000	376400

TG6	**8*8**	**1000**	2000	304000
			4000	306700
			5000	294400
TG7	**8*8**	**1000**	2000	160400
			4000	160700
			5000	166600
TG8	**6*6**	**1000**	2000	101700
			4000	107300
			5000	109400
TG9	**6*6**	**1000**	2000	51600
			4000	51100
			5000	52100
TG10	**6*6**	**1000**	2000	37400
			4000	38600
			5000	36100

Il ressort des résultats que le nombre d'itération n'améliore pas toujours dans les résultats. Nous constatons que la majorité des résultats en augmentant le nombre d'itération

sont des fois mauvaises que les précédents comme le cas des graphes TG1, TG2, TG7 et TG8 les résultats ou le nombre d'itération égale 2000 sont meilleur que 5000 et 4000.

Comparaison de GEO avec GWO-TAB, BFO et ABC

Après avoir des résultats de l'algorithme d'optimisation Golden Eagle dans le cas où on changerait dans des caractéristiques (nombre de population et le nombre d'itérations), nous comparons maintenant ces résultats avec d'autres résultats des algorithmes existant dans la littérature comme : BFO (Bacterial Foraging Algorithm), ABC (Artificiel Bee Colony) et GWO-TAB.

L'algorithme GWO-TAB est un hybride entre l'algorithme GWO basé sur le comportement de nourriture chez les loups gris et l'algorithme de recherche tabou(Yeddou, 2020).

L'algorithme BFO est une méta-heuristique inspire le comportement de butinage chez les bactéries(MESSAOUDI, 2019).

L'algorithme ABC est une méta-heuristique inspire sur le comportement de butinage chez les abeilles.

Tableau 5-3 : Comparaison GEO avec GWO-TAB, BFO, ABC et sur Noc 2D.

Graphe	*Topologie*	*GWO-TAB*	*BFO*	*GEO*	*ABC*
TG1	**10*10**	681300	697500	725100	793700
TG2	**10*10**	599000	648000	655400	711500
TG3	**11*11**	828600	886800	886600	954800
TG4	**10*10**	462900	518700	540600	597500
TG5	**8*8**	380200	382200	376400	419600

TG6	**8*8**	271200	268200	289400	326400
TG7	**8*8**	109800	134900	152900	179900
TG8	**6*6**	95800	88500	101400	115100
TG9	**6*6**	37900	46700	48300	63400
TG10	**6*6**	25500	32400	36100	49300

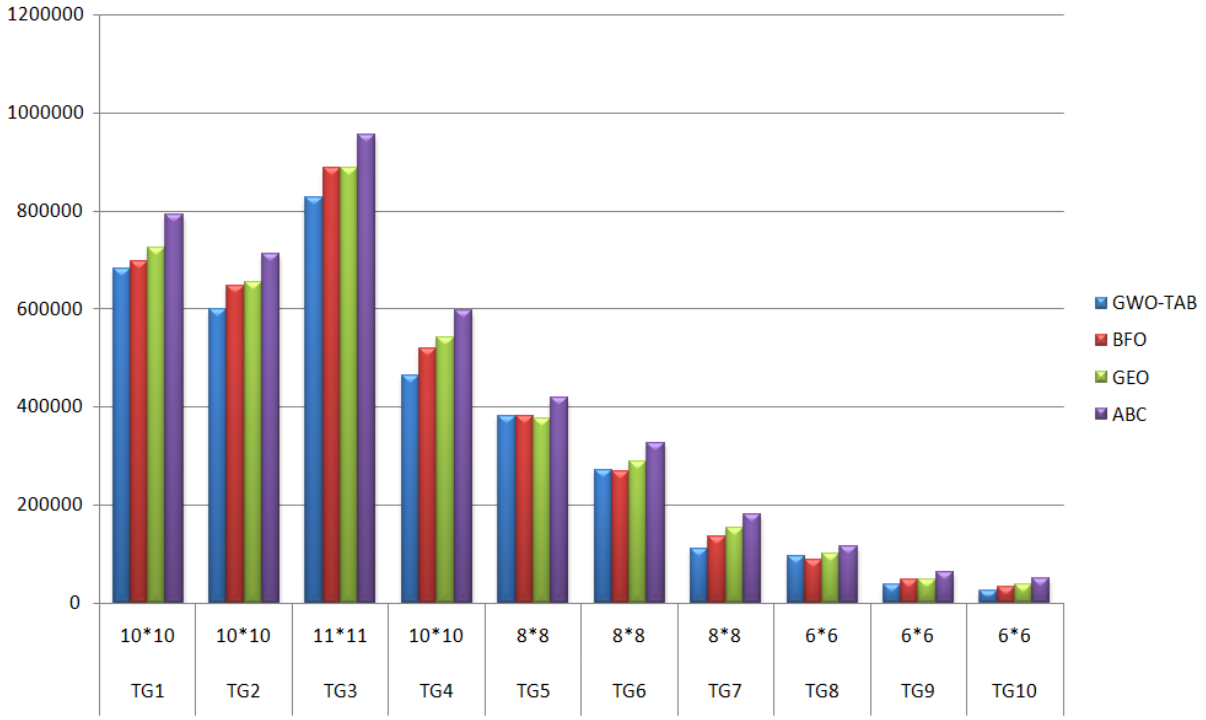


Figure 5-6: Comparaison des meilleurs résultats de GEO, GWO-TAB, ABC et FBO dans Noc 2D.

D'abord, nous remarquons que la technique GEO dans tous les Benchmarks, donne des bons résultats par rapport à la technique ABC. Aussi, nous constatons que GEO fournit de résultats mieux par rapport à FBO dans les benchmarks TG3 et TG5, et mieux par rapport à

GWO-TAB dans le benchmark TG5.La dernière remarque est que les techniques FBO et GWO-TAB sont mieux que GEO.

5.5.2.2 Réalisation des tests dans Noc 3D

- ***Changement de la taille de population***

On fait varier la taille de la population initiale tout en gardant le même nombre d'itérations à 1000. Les résultats obtenus sont les suivants :

Tableau 5-4 : Résultats obtenus en changeant la taille de population initiale dans Noc 3D.

Graphe	*Topologie*	*Itération*	*Population*	*Noc 3D*
TG1	**5*5*5**	**1000**	**500**	522 300
			1000	531 100
			1500	531 100
TG2	**5*5*5**	**1000**	**500**	500 900
			1000	501 600
			1500	491 000
TG3	**5*5*5**	**1000**	**500**	602 600
			1000	606 000
			1500	610 400
			500	414 900

TG4	**5*5*5**	**1000**	**1000**	417 400
			1500	420400
TG5	**4*4*4**	**1000**	**500**	297 100
			1000	295 800
			1500	296 600
TG6	**4*4*4**	**1000**	**500**	222 200
			1000	223 000
			1500	205 200
TG7	**4*4*4**	**1000**	**500**	119 600
			1000	123 300
			1500	118 700
TG8	**3*3*3**	**1000**	**500**	75 400
			1000	76 900
			1500	78 000
TG9	**3*3*3**	**1000**	**500**	38 700
			1000	35 200
			1500	37 700

TG10	**3*3*3**	**1000**	**500**	28 800
			1000	28 800
			1500	28 100

D'après les résultats obtenus, nous remarquons que l'algorithme GEO peut fournir des bons résultats lorsque la taille de la population est grande. En plus les meilleurs résultats trouvés dans des foules de 1000 et 1500 comme le cas des Benchmarks TG2,TG3,TG5,TG6,TG7,TG9,TG10 . Mais ces résultats sont encore relatifs, du fait du caractère aléatoire de l'algorithme GEO.

- ***Changement de nombre d'itérations***

Nous comparons maintenant les résultats tout en variant sur le nombre d'itérations, en gardant la même taille de population 200 individus.

Tableau 5-5 : Résultats obtenus en changeant la taille d'itération dans Noc 3D.

Graphe	*Topologie*	*Population*	*Itération*	*Noc 3D*
TG1	**5*5*5**	**100**	**2000**	533700
			5000	525300
TG2	**5*5*5**	**100**	**2000**	500100
			5000	502000
TG3	**5*5*5**	**100**	**2000**	613900
			5000	606800

TG4	**5*5*5**	**100**	**2000**	411 600
			5000	407 600
TG5	**4*4*4**	**100**	**2000**	291 900
			5000	284 400
TG6	**4*4*4**	**100**	**2000**	223 900
			5000	217 200
TG7	**4*4*4**	**100**	**2000**	121 100
			5000	119 900
TG8	**3*3*3**	**100**	**2000**	78 700
			5000	77 400
TG9	**3*3*3**	**100**	**2000**	38 300
			5000	36 100
TG10	**3*3*3**	**100**	**2000**	28 100
			5000	26 700

Notez que le nombre d'itérations améliore les résultats, mais l'amélioration se produit après un grand nombre d'itérations, ce qui signifie que l'amélioration est lente. De plus, le nombre de tâches affecte l'amélioration, nous avons donc remarqué qu'un grand nombre de tâches nécessitent plus d'itérations pour améliorer les résultats.

Comparaison de GEO avec GWO-TAB, BFO et ABC

D'après avoir des résultats de l'algorithme d'optimisation Golden Eagle dans le cas où on changerait dans des caractéristiques (nombre de population et le nombre d'itérations), nous comparons maintenant ces résultats avec d'autres résultats des algorithmes existant dans la littérature comme : BFO (Bacterial Foraging Algorithm), ABC (Artificiel Bee Colony) et GWO-TAB.

Tableau 5-6 : Comparaison GEO avec GWO-TAB, BFO, ABC et sur Noc 3D.

Graphe	*Topologie*	*GWO-TAB*	*BFO*	*GEO*	*ABC*
TG1	**5*5*5**	473100	385900	522300	521200
TG2	**5*5*5**	447800	344900	491000	431100
TG3	**5*5*5**	576200	505400	606800	570400
TG4	**5*5*5**	339200	220400	407600	329100
TG5	**4*4*4**	286500	240600	284400	273200
TG6	**4*4*4**	200100	162500	205200	199900
TG7	**4*4*4**	96700	96100	118700	121100
TG8	**3*3*3**	71200	62600	75400	73000
TG9	**3*3*3**	33600	33700	35200	40600
TG10	**3*3*3**	24300	27800	26700	31700

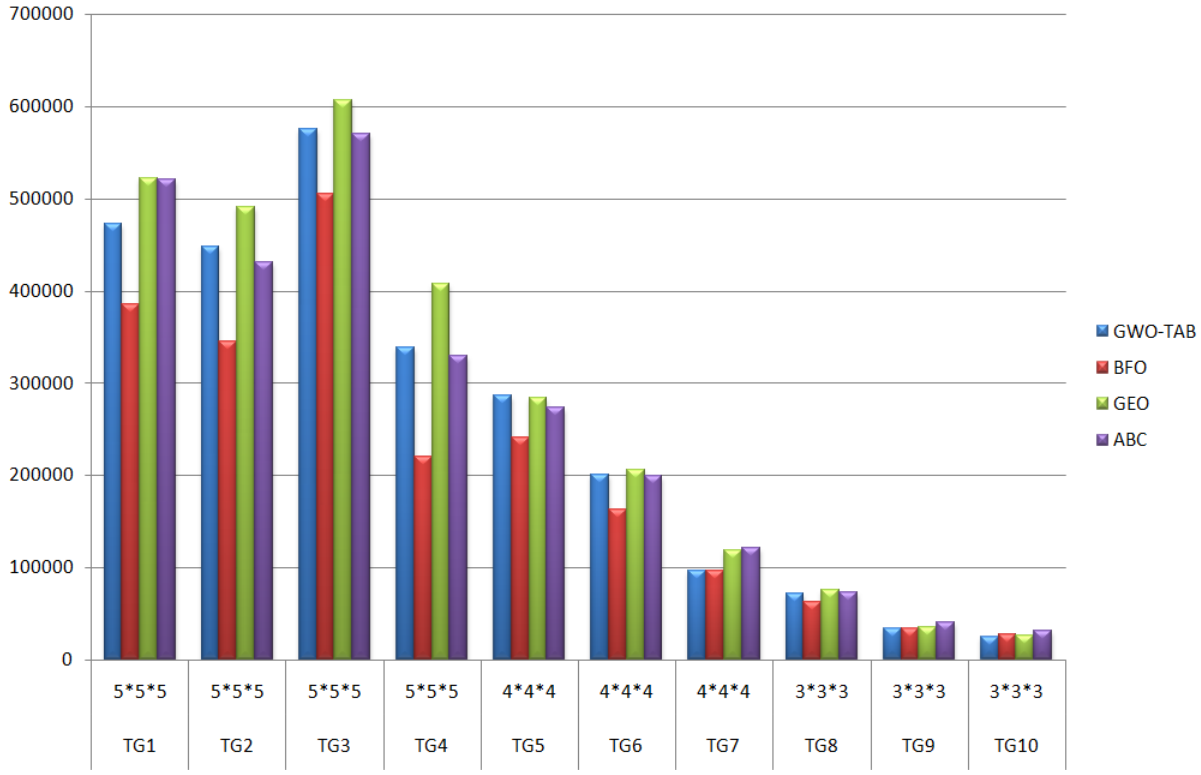


Figure 5-7: Comparaison des meilleurs résultats de GEO, GWO-TAB, ABC et FBO dans Noc 3D.

Tout d'abord, on note que dans les benchmarks TG7, TG9 et TG10, la technique GEO donne des meilleurs résultats par rapport à ABC, et donc dans le cas des graphes plus petits, GEO donne de meilleurs résultats que les techniques ABC. Aussi, nous constatons que GEO fournit de résultats mieux par rapport à BFO dans le benchmark TG10, et mieux par rapport à GWO-TAB dans le benchmark TG5.La dernière remarque est que les techniques FBO ABC et GWO-TAB sont mieux que GEO dans Noc 3D.

5.5.2.3 Réalisation des tests dans WNoC 2D

- ***Changement de la taille de population***

On fait varier la taille de la population initiale tout en gardant le même nombre d'itérations à 1000. Les résultats obtenus sont les suivants :

Tableau 5-7 : Résultats obtenus en changeant la taille de population initiale dans WNoC 2D.

Graphe	*Topologie*	*Nombre SR*	*Itération*	*Population*	*WNoC 2D*
TG1	**10*10**	**4**	**1000**	10	786700
				50	763600
				100	740400
				200	744600
				500	735600
				1000	725600
				1500	689500
TG2	**10*10**	**4**	**1000**	10	758300
				50	737500
				100	709800
				200	687400
				500	675200
				1000	671800
				1500	646400
				10	963200

TG3	**12*12**	**4**	**1000**	50	961300
				100	936000
				200	928800
				500	907200
				1000	874100
				1500	889600
TG4	**10*10**	**4**	**1000**	10	613100
				50	598100
				100	584600
				200	590400
				500	589500
				1000	596800
				1500	576200
TG5	**10*10**	**4**	**1000**	10	432000
				50	425200
				100	412600
				200	414300

				500	403600
				1000	392300
				1500	387700
TG6	**10*10**	**4**	**1000**	10	324600
				50	321300
				100	319200
				200	303800
				500	306700
				1000	287600
				1500	304700
TG7	**10*10**	**4**	**1000**	10	179700
				50	181200
				100	172600
				200	166100
				500	164700
				1000	163400
				1500	151300

TG8	**6*6**	**4**	**1000**	10	120200
				50	104100
				100	105300
				200	104300
				500	107600
				1000	106200
				1500	98600
TG9	**6*6**	**4**	**1000**	10	58700
				50	55300
				100	55000
				200	50100
				500	53300
				1000	52800
				1500	51500
				10	41900
				50	43100
				100	41600

TG10	**6*6**	**4**	**1000**	200	41600
				500	38900
				1000	39200
				1500	34100

À travers les résultats obtenus, nous remarquons que la majorité des meilleurs résultats obtenus se retrouvent dans une population de taille 1500, donc la population améliore dans les résultats de coût de communication. Mais ces résultats sont encore relatifs, du fait du caractère aléatoire de l'algorithme GEO.

- ***Changement de nombre d'itérations***

Nous comparons maintenant les résultats tout en variant sur le nombre d'itérations, en gardant la même taille de population 1000 individus.

Tableau 5-8 : Résultats obtenus en changeant la taille d'itération dans WNoC 2D.

Graphe	*Topologie*	*Nombre SR*	*Population*	*Itération*	*WNoc 2D*
TG1	**10*10**	**4**	**1000**	2000	776100
				4000	738800
				5000	776500
TG2	**10*10**	**4**	**1000**	2000	655200
				4000	663500
				5000	709400

TG3	**12*12**	**4**	**1000**	2000	906900
				4000	922700
				5000	926800
TG4	**10*10**	**4**	**1000**	2000	563600
				4000	567400
				5000	552100
TG5	**8*8**	**4**	**1000**	2000	418100
				4000	402500
				5000	384700
TG6	**8*8**	**4**	**1000**	2000	324000
				4000	306500
				5000	319400
TG7	**8*8**	**4**	**1000**	2000	160100
				4000	170700
				5000	171600
TG8	**6*6**	**4**	**1000**	2000	109600
				4000	105000

				5000	113500
TG9	**6*6**	**4**	**1000**	2000	51600
				4000	52000
				5000	54100
TG10	**6*6**	**4**	**1000**	2000	40900
				4000	38400
				5000	43100

Il ressort des résultats que le nombre d'itérations n'améliore pas toujours dans les résultats. Nous constatons que la majorité des résultats en augmentant le nombre d'itérations sont des fois mauvaises que les précédents. En plus les meilleurs résultats trouvés dans des foules de 2000 et 4000 itérations comme le cas des Benchmarks TG1, TG2, TG3, TG6, TG7, TG8, TG9,TG10.

- **Comparaison entre les topologies Noc 2D, WNoC 2D et Noc 3D**

Dans cette partie nous faisons une comparaison entre l'architecture Noc 2D, NoC 3D et WNoC 2D en choisissant les meilleurs résultats obtenus dans les trois architectures.

Tableau 5-9 : Meilleurs résultats pour les trois topologies.

	Noc 2D	***WNoC 2D***	***Noc 3D***
TG1	725100	689500	522300
TG2	655400	646400	491000
TG3	886600	889600	606800

TG4	540600	552100	407600
TG5	376400	387400	284400
TG6	289400	287600	205200
TG7	152900	151300	118700
TG8	101400	98600	75400
TG9	48300	50100	35200
TG10	36100	34100	26700

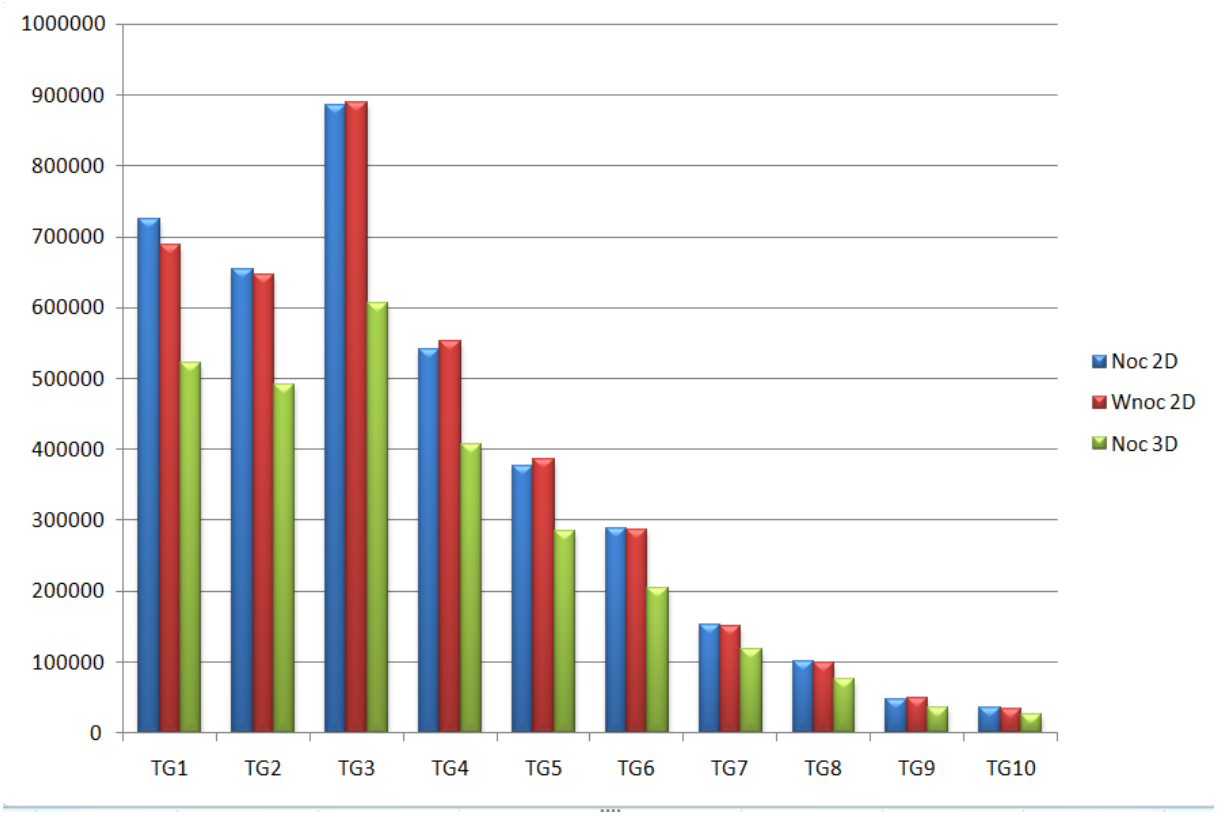


Figure 5-8: Comparaison des meilleurs résultats pour les trois topologies.

D’après le graphique et le tableau de compassion, nous remarquons que la technique GEO fournit de meilleurs résultats dans le cas de la topologie Noc 3D dans tous les Benchmarks par rapport ou les deux êtres topologies Noc 2D et WNoc 2D.Aussi, nous constatons que WNoc 2D fournit de résultats mieux que la topologie 2D dans les benchmark TG1, TG2, TG6, TG7, TG8 et TG10.La dernière remarque est que la topologie Noc 3D fournit une bonne minimisation de coût de communication, en appliquant l'algorithme d'optimisation Golden Eagle.

5.5.2.4 Comparaison entre les résultats obtenus et ceux de la littérature

Les dimensions sont présentées dans le tableau suivant :

Tableau 5-10 : Les dimensions utilisé dans les tests.

Benchmark	2D Mesh	3D Mesh
VOPD	4x4	2 x 4 x 2
MWD	4x4	2x 4 x 2

- **Comparaison entre GEO et les résultats obtenus sur NoC 2D**

- CSA (algorithme basé sur corbeaux) implémentées dans le projet de fin d’étude (TOUIMER, 2022) .

Tableau 5-11 : Comparaison entre GEO avec d’autres résultats s sur NoC 2D.

Algorithme de mapping	VOPD	MWD
2D ILP	4119	1120
SA	4231	1451

GA	4218	1321
PSO	4119	1120
BA	4119	1122
SCSO	4119	1122
CSA	4320	1248
GEO	5228	1696

.

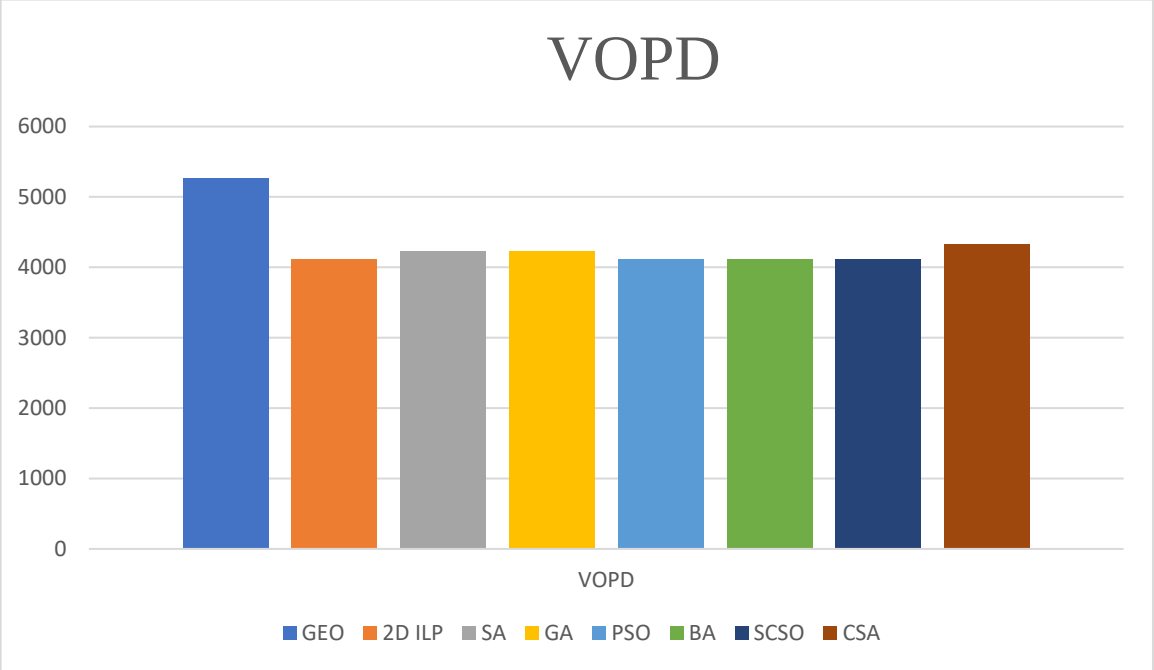


Figure 5-9 : comparaison des résultats utilisant le Benchmark VOPD dans 2D NOC.

Nous constatons que, comparée à tous les autres algorithmes dans la structure bidimensionnelle à l'aide du benchmark VOPD, l'algorithme GEO ne démontre aucune amélioration.

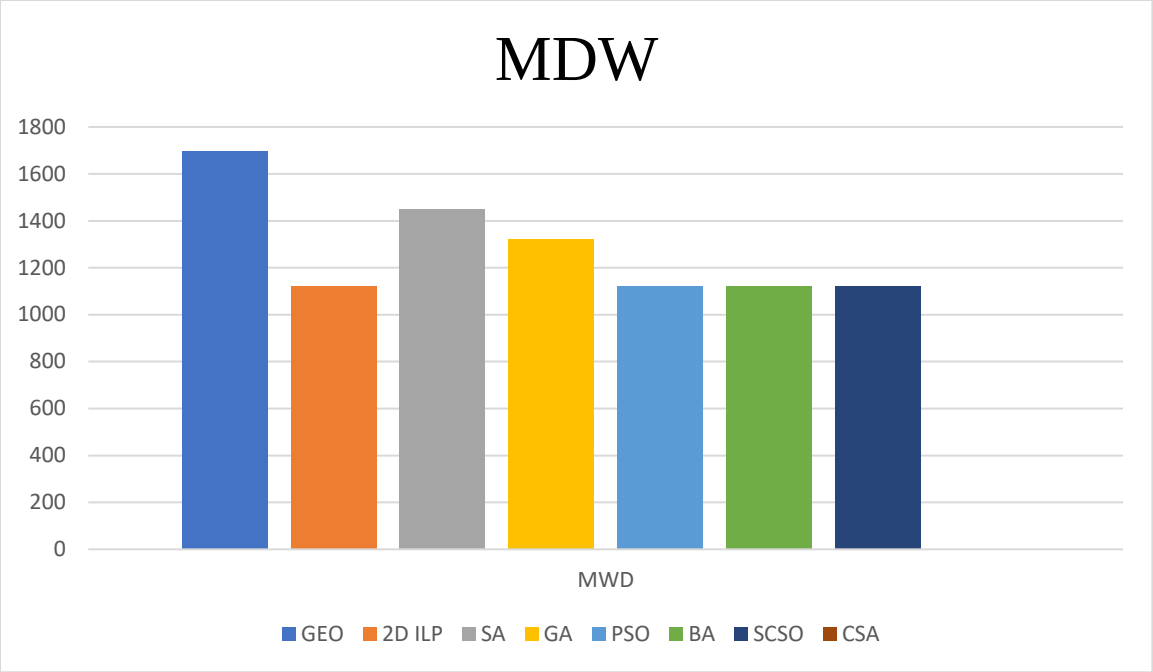


Figure 5-10: comparaison des résultats utilisant le Benchmark MWD dans 2D NOC.

Nous notons que l'algorithme n'a montré aucune amélioration sur tous les algorithmes comparés à lui dans la structure 2D en utilisant le Benchmark MWD.

- **Comparaison entre GEO et les résultats obtenus avec d'autres résultats dans la littérature sur NoC 3D**

Tableau 5-12 : Comparaison des meilleurs résultats utilisant le Benchmark VOPD, MWD dans 3D NOC.

Algorithme de mapping	VOPD	MWD
3D ILP	4110	1120
GA	4218	1321
PSO	4119	1120
BA	4119	1122
SCSO	4119	1122

CSA	4103	1216
GEO	3683	1088

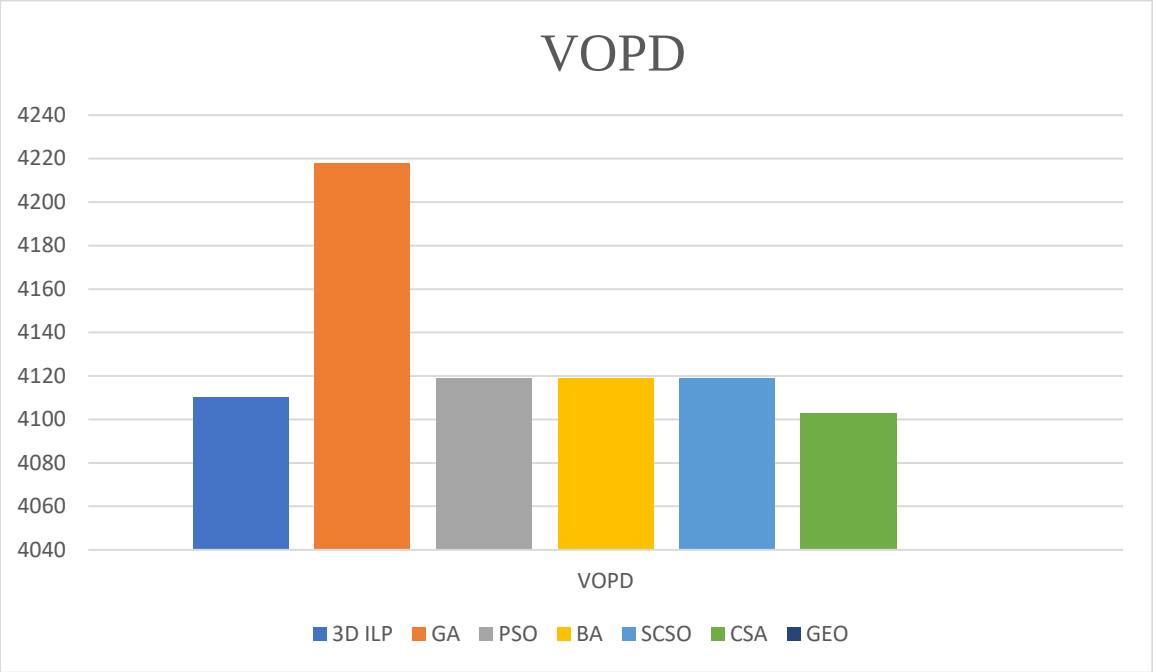


Figure 5-11: comparaison des meilleurs résultats utilisant le Benchmark VOPD dans NOC 3D.

Les résultats ci-dessus indiquent que l'algorithme GEO a donné un meilleur résultat que tous les algorithmes que nous avons comparés.

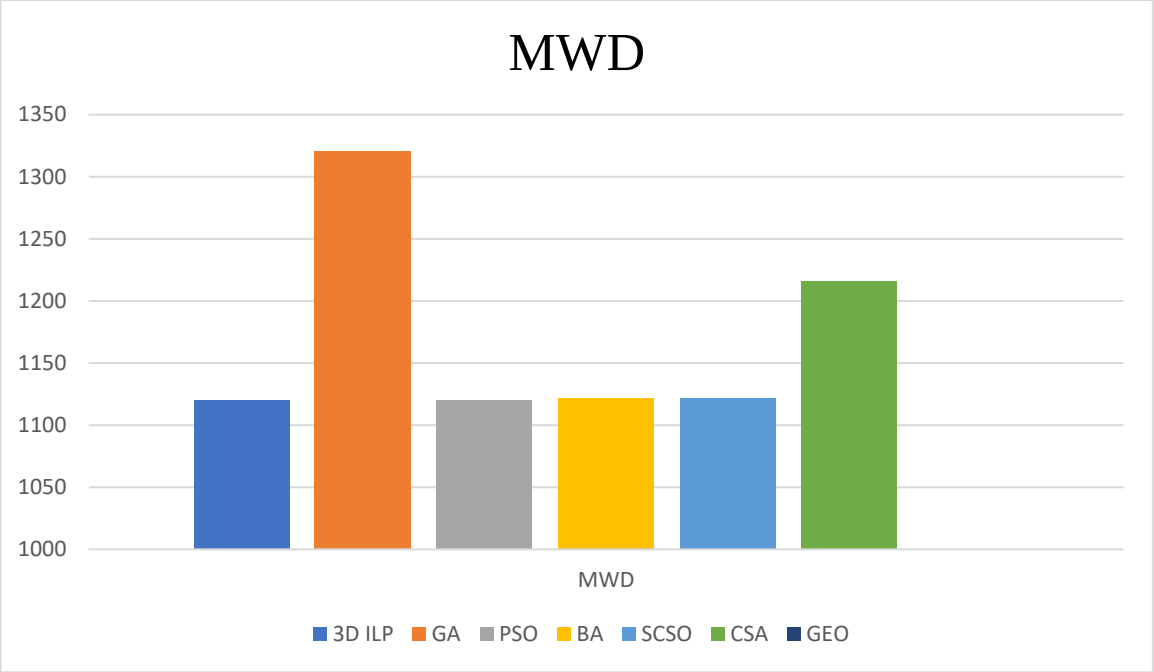


Figure 5-12 : comparaison des meilleurs résultats utilisant le Benchmark MWD dans NOC 3D.

Les résultats ci-dessus montrent que l'algorithme GEO a donné un meilleur résultat que tous les algorithmes auxquels nous l'avons comparé.

5.6 Conclusion

L'objectif de notre travail est d'optimiser le coût de communication dans les topologies Noc (Noc 2D, WNoc 2D et NoC 3D) pour les différents benchmarks qu'on a utilisés. Pour cela, nous utilisons l'algorithme d'optimisation Golden Eagle qui est une méthode basée sur le comportement de la nourriture chez les aigles.

Dans ce chapitre, nous avons testé la méthode de mapping implémentée en utilisant GEO dans trois architectures NoC et à comparer les résultats obtenus avec ceux obtenus par d'autres techniques publiées dans la littérature.

Après l'implémentation de cet algorithme et la réalisation des tests sur les résultats obtenus, nous avons constaté que la méthode GEO implémentée dans une architecture NoC 3D fournit une bonne minimisation de coût de communication par rapport à l'architecture NoC 2D et WNoc 2D.

Conclusion générale

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

Les recherches et travaux menés dans le cadre de ce projet ont permis de manifester un intérêt pour l'utilisation des structures de communication réseau sur puce, pour résoudre les problèmes de communication dans les systèmes sur puce. Parmi les défis de la conception des réseaux sur puce, nous trouvons le problème du placement (Mapping). Ce problème implique de trouver l'emplacement des tâches d'application sur les éléments de calcul pour une topologie Noc donnée.

L'objectif principal du travail présenté dans ce mémoire est de proposer une technique permettant de mapper l'IP d'une application sur une structure de réseau sur puce. Les étapes de mapping sont un problème NP difficile, dans ce cas l'utilisation d'algorithmes métaheuristiques est un bon moyen d'avoir un mapping, dans ce master nous proposons d'utiliser un nouvel algorithme basé sur l'aigle royal.

L'optimisation de l'aigle royal qui est inspiré principalement par l'intelligence des aigles royaux dans le réglage de la vitesse à différentes étapes de leur trajectoire en spirale pour la chasse, qui permet de réduire le coût des communications sur un réseau sur puce 2D, 3D et WNoc 2D.

Enfin, plusieurs expériences ont été réalisées sur le TGFF choisi comme échantillon teste pour démontrer l'efficacité de l'algorithme utilisé et implémenté.

Perspectives

Un nombre d'enrichissements pourra être apporté à notre travail :

- Appliquer d'autres topologies telles que WNoC 3D ou la topologie 3D avec des liens TSV minimiser.
- Hybridation de cet algorithme avec d'autres algorithmes (par exemple PSO ou tabou ou génétique).
- Appliquer la technique proposée au problème de mapping multi-objectifs.
- Inclure un réseau sur puce hétérogène.

Référence

[Guide ultime] Apprenez la programmation Java par la pratique. (n.d.).

6 Best Open Source IDEs for Java Programming Language. (2019). Ajoy Kumar.

Alagarsamy, A., & Gopalakrishnan, L. (2018). Mba: a new cluster based bandwidth and power aware mapping for 2d noc. *2018 International Conference on Circuits and Systems in Digital Enterprise Technology (ICCSDET)*, 1–4.

Alagarsamy, A., & Gopalakrishnan, L. (2016). SAT: a new application mapping method for power optimization in 2D—NoC. *2016 20th International Symposium on VLSI Design and Test (VDAT)*, 1–6.

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

Alimi, I. A., Patel, R. K., Aboderin, O., Abdalla, A. M., Gbadamosi, R. A., Muga, N. J., L, P. and A., & N, A. (2021). Network-On-Chip Topologies: Potentials, Technical Challenges, Recent Advances and Research Direction. In *Network-on-Chip-Architecture, Optimization, and Design Explorations.*

ALLARDBERNIER, J. (2011). *MÉTHODE DE RECONFIGURATION DYNAMIQUE POUR UN RÉSEAU-SUR-PUCE TOLÉRANT AUX FAUTES*. ÉCOLE POLYTECHNIQUE DE MONTRÉAL.

Amor, N., Noman, M. T., Petru, M., & Sebastian, N. (2022). Comfort evaluation of ZnO coated fabrics by artificial neural network assisted with golden eagle optimizer model. *Scientific Reports*, *12*(1), 1–16.

Aroussi, S. (2016). *Cours de Heuristique et méta heuristique*.

https://sites.google.com/a/esi.dz/s-aroussi/

ATAT, Y. (2007). *Conception de haut niveau des MPSoCs à partir d'une spécification Simulink : Passerelle entre la conception au niveau système et la génération d'architecture*. Institut National Polytechnique de Grenoble.

Belkacemi, D. (2020). *Mapping d'applications parallèles sur des architectures embarquées multiprocesseurs à base de réseaux sur puce*. Université Mouloud Mammeri.

Ben Abdallah, A. (2017). Advanced Multicore Systems-On-Chip. In *Springer Nature Singapore Pte Ltd*. https://doi.org/10.1007/978-981-10-6092-2

Bjerregaard, T., & Mahadevan, S. (2006). A survey of research and practices of network-on-chip. *ACM Computing Surveys*, *38*(1). https://doi.org/10.1145/1132952.1132953

BOUGUETTAYA, A. (2017). *Génération d'un réseau sur puce au format VHDL RTL à partir d'une modélisation de haut niveau UML par raffinement*. UNIVERSITE BADJI MOKHTAR - ANNABA.

Boumaaza, F., & Benyamina, A. E. H. (2012). *Mapping multi Objectifs d 'application intensive Sur architecture MPSOC*.

Bozorg-Haddad, O., Solgi, M., & Loáiciga, H. A. (2017). Meta-Heuristic and Evolutionary Algorithms for Engineering Optimization. In *Meta-Heuristic and Evolutionary Algorithms for Engineering Optimization*. https://doi.org/10.1002/9781119387053

Chakravarthi, V. S. (2020). A Practical Approach to VLSI System on Chip (SoC) Design. In *Springer International Publishing.* https://doi.org/10.1007/978-3-030-23049-4

Chariete, A. (2014). Approches d'optimisation et de personnalisation des r´eseaux sur puce (NoC : Networks on Chip) [Universite de Technologie de Belfort-Montb´eliard (UTBM)]. In *Universite de Technologie de Belfort-Montb´eliard (UTBM)*. https://doi.org/<tel-01174271

Chatmen, M. F. (2016). *Conception d'un réseau sur puce optimisé en latence*. Université de Bretagne Sud.

Chatmen, M. F. (2017). *Conception d ' un réseau sur puce optimisé en latence*. Université de Bretagne Sud.

Chatterjee, N., Mukherjee, P., & Chattopadhyay, S. (2018). Reliability-aware application

mapping onto mesh based Network-on-Chip. *Integration*, *62*, 92–113.

Dageleh, M. Z., & Jamali, M. A. J. (2018). V-CastNet3D: a novel clustering-based mapping in 3-D network on chip. *Nano Communication Networks*, *18*, 51–61.

Dally, Willliam james; Towlrd, B. (2004). *Principles and practices of interconnection networks*. Elsevier.

Delorme, J. (2007). *Méthodologie de modélisation et d ' exploration d ' architecture de réseaux sur puce appliquée aux télécommunications To cite this version : Thèse*. l'Institut national des sciences appliquées de Rennes.

Delorme, J., & Houzet, D. (2006). *Une technique de mapping pour les réseaux sur puce de structure 2D*.

DJABALLAH, N. (2011). *une architecture d'un réseau sur puce (NOC) dédiée au routage de paquets*. Université El Haj Lakhdar BATNA.

Érika, C., Alexandre, Amory, M., & Marcelo, S. L. (2011). *Reliability, Availability and Serviceability of Networks-on-Chip* (pringer Sc). Springer New York, NY. https://doi.org/https://doi.org/10.1007/978-1-4614-0791-1

Famenne, M. E. (2012). *NetBeans IDE - Students of IEPSCF Marche*.

Fang, J., Yu, T., & We, Z. (2019). Improved Ant Colony Algorithm Based on Task Scale in Network on Chip (NoC) Mapping. *Electronics*, *9*(1), 6. https://doi.org/10.3390/electronics9010006

Fennema, K. (2018). *Golden eagles in Scotland - Braemar Highland Experience*.

Gan, Y., Guo, H., & Zhou, Z. (2021). 3D NoC low-power mapping optimization based on improved genetic algorithm. *Micromachines*, *12*(10). https://doi.org/10.3390/mi12101217

Gherboudj, A. (2013). Méthodes de résolution de problèmes difficiles académiques. *Université de Constantine2*.

Ghoumari, A. (2018). *Métaheuristiques adaptatives d'optimisation continue basées sur des méthodes d'apprentissage*. Université Paris-Est.

Glover, F. (1986). Future paths for integer programming and links to artificial intelligence.

Computers and Operations Research, *13*(5), 533–549. https://doi.org/10.1016/0305-0548(86)90048-1

GUÉRARD, H. (2011). *INTÉGRATION D ' UN MODÈLE DE RÉSEAU SUR PUCE DANS UN FLÔT DE CONCEPTION DE NIVEAU SYSTÈME*. ÉCOLE POLYTECHNIQUE DE MONTRÉAL.

Hao, J.-K., Galinier, P., & Habib, M. (1999). Métaheuristiques pour l'optimisation combinatoire et l'affectation sous contraintes. *Revue d'intelligence Artificielle*, *13*(2), 283–324.

He, H., Fang, F., & Wang, W. (2018). *Improved Simulated Annealing Genetic Algorithm based Low power mapping for 3D NoC*. *02022*.

Hu, W.-H., Wang, C., & Bagherzadeh, N. (2012). Design and analysis of a mesh-based wireless network-on-chip. *2012 20th Euromicro International Conference on Parallel, Distributed and Network-Based Processing*, 483–490.

Isabirye, M., Raju, D. V. ., Kitutu, M., Yemeline, V., Deckers, J., & J. Poesen Additional. (2012). We are IntechOpen , the world ' s leading publisher of Open Access books Built by scientists , for scientists TOP 1 %. *Intech*, 13. http://dx.doi.org/10.1039/C7RA00172J%0Ahttps://www.intechopen.com/books/advanced-biometric-technologies/liveness-detection-in-biometrics%0Ahttp://dx.doi.org/10.1016/j.colsurfa.2011.12.014

Janga Reddy, M., & Nagesh Kumar, D. (2020). Evolutionary algorithms, swarm intelligence methods, and their applications in water resources engineering: A state-of-the-art review. *H2Open Journal*, *3*(1), 135–188. https://doi.org/10.2166/h2oj.2020.128

Janidarmian, M., & Fekr, A. R. (2012). A survey of meta-heuristic solution methods for mapping problem in Network-on-Chips. *Advanced Materials Research*, *403–408*, 3994–4008. https://doi.org/10.4028/www.scientific.net/AMR.403-408.3994

Jantsch, A., & Tenhunen, H. (2003). Networks on Chip. In *KLUWER ACADEMIC PUBLISHERS*. https://doi.org/10.1007/b105353

java. (n.d.). Retrieved July 2, 2022, from https://www.java.com/fr/

Jerger, N. E., Krishna, T., & Peh, L. S. (2017). *On-Chip Networks Synthesis Lectures on*

Computer Architecture. https://doi.org/https://doi.org/10.2200/S00772ED1V01Y201704CAC040

Joshi, S. S., & Biradar, S. R. (2021). A novel golden eagle optimizer based trusted ad hoc on-demand distance vector (geo-taodv) routing protocol. *International Journal of Computer Networks and Applications*, *8*(5), 538–548. https://doi.org/10.22247/ijcna/2021/209986

KAMECHE, A. H. (2013). *Approches basées sur la BBO pour le Data Clustering et le NoC Mapping*. École Doctorale Sciences et Technologies de l'Information et de la Communication(STIC).

Kirkpatrick, S., Gelatt, C. D., & M. P, V. (n.d.). Optimization by Simulated Annealing. *Paper Knowledge . Toward a Media History of Documents*, *220*(4598), 671–680.

Kundu, S., & Chattopadhyay, S. (2014). Network-on-chip: The next generation of system-on-chip integration. In *Network-on-Chip: The Next Generation of System-on-Chip Integration*. https://doi.org/10.1201/b17748

L'aigle royal, le plus majestueux des rapaces. (n.d.).

Lemaire, R., Jerraya, A. A., & Torres, L. (2006). *Conception et modélisation d ' un système de contrôle d ' applications de télécommunication avec une architecture de réseau sur puce (NoC)*. INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE N°.

Leroy, A. (2007). *Optimizing the on-chip communication architecture of low power Systems-on-Chip in Deep Sub-Micron technology*. Université Libre de Bruxelles.

Luca, B., & Giovanni, D. M. (2006). *Networks on Chips. Technology and Tools* (B. Luca (Ed.)). Morgan Kaufmann is an imprint of Elsevier.

Luo-Feng, G., Gao-ming, D., Duo-Li, Z., Ming-Lun, G., Ning, H., & Yu-Kun, S. (2008). Design and performance evaluation of a 2D-mesh network on chip prototype using FPGA. *IEEE Asia-Pacific Conference on Circuits and Systems, Proceedings, APCCAS*, 1264–1267. https://doi.org/10.1109/APCCAS.2008.4746257

MAHBOUBI, T. (2019). *Une approche intelligente pour un problème d'optimisation multicritère*.

Martí, R., & Reinelt, G. (2022). *Exact and Heuristic Methods in Combinatorial Optimization: A Study on the Linear Ordering and the Maximum Diversity Problem*. Springer.

MESSAOUDI, D. (2019). Mapping dans les réseaux sur puce 3D (3D NoCs) utilisant l ' algorithme des bactéries. In *Mapping dans les réseaux sur puce 3D (3D NoCs) utilisant l'algorithme des bactéries*. ECOLE NORMALE SUPERIEURE Kouba, ALGER Département.

Mohammadi-Balani, A., Nayeri, M. D., Azar, A., & Taghizadeh-Yazdi, M. (2021). Golden eagle optimizer: A nature-inspired metaheuristic algorithm. *Computers & Industrial Engineering*, *152*, 107050.

Moore, G. E. (2005). Understanding Moore's Law: Four Decades of Innovation. In D. Brock (Ed.), *The Electrochemical Society Interface* (Vol. 14, Issue 2, pp. 18–19). https://doi.org/10.1149/2.f01052if

N.Toubaline. (2018). *Aide a La Conception d ' un Reseau Sur*. Université de Saad dahleb Blida.

Nalci, Y., Kullu, P., Tosun, S., & Ozturk, O. (2021). ILP formulation and heuristic method for energy-aware application mapping on 3D-NoCs. *The Journal of Supercomputing*, *77*(3), 2667–2680.

NEHNOUH, C. (2019). *Conception & implémentation d'un algorithme de routage tolérant aux fautes.* Université Oran 1.

Ngan, N. (2011). *Etude et conception d'un réseau sur puce dynamiquement adaptable pour la vision embarquée*. Paris Est.

Okwu, M. O., & Tartibu, L. K. (2020). *Metaheuristic optimization: Nature-inspired algorithms swarm and computational intelligence, theory and applications* (Vol. 927). Springer Nature.

Ouardia, A. (2011). segmentation d'images par seuillage d'histogrammes bidimensionnels. *These de Magister, Université de Tizi-Ouzou*.

Ouled-khaoua, A., & Terranti, H. (2020). *Performance Evaluation of Networks On-chip Topologies*. Université de saad dahleb Blida.

Rezaei, A., Daneshtalab, M., Zhao, D., Safaei, F., Wang, X., & Ebrahimi, M. (2015). Dynamic application mapping algorithm for wireless network-on-chip. *2015 23rd Euromicro International Conference on Parallel, Distributed, and Network-Based*

Processing, 421–424.

Rezaei, A., Zhao, D., Daneshtalab, M., & Zhou, H. (2017). *Multi-Objective Task Mapping Approach for Wireless NoC in Dark Silicon Age **. 589–592. https://doi.org/10.1109/PDP.2017.12

Sacanamboy-Franco, M., Bolaños-Martinez, F., Bernal-Noreña, Á., & Nieto-Londoño, R. (2017). Genetic algorithm for task mapping in embedded systems on a hierarchical architecture based on wireless network on chip WiNoC. *Dyna*, *84*(201), 202–209.

Sahu, P. K., & Chattopadhyay, S. (2013). A survey on application mapping strategies for Network-on-Chip design. *Journal of Systems Architecture*, *59*(1), 60–76. https://doi.org/10.1016/j.sysarc.2012.10.004

Saïd, L. (2013). Méthodes bio-inpirées hybrides pour la résolution de problèmes complexes. *Université Constantine*, *2*.

SALEEM, A. (2015). *Implementation and Performance Analysis of Wishbone Shared Bus for Single Master-Multiple Slaves*. université technologique de Tampere.

Salim, A., & Selma, T. (2020). Optimisation multiobjectif des performances des réseaux sur puce. In *Université SAAD DAHLEB BLIDA-1*.

SAMUEL, K. M. (2021). Cerebras' New Monster AI Chip Adds 1.4 Trillion Transistors. *IEEE SPECTRUM*. https://spectrum.ieee.org/amd-3d-stacking-intel-graphcore

Séverine, R. (2005). Evaluation des paramètres des Réseaux sur puce. In *Laboratoire d'Informatique, de Robotique et de Micro-électronique de Montpellier*.

Séverine Riso. (2005). *évaluation des paramètres architecturaux des réseaux sur puce*.

Sheng Ma, Libo Huang, M. L. and W. S. (2015). Networks-On-Chip. In Z. Wang (Ed.), *Networks-On-Chip*. Morgan Kaufmann is an imprint of Elsevier. https://doi.org/10.1016/c2013-0-19178-7

Sikandar, S., Baloch, N. K., Hussain, F., Amin, W., Zikria, Y. Bin, & Yu, H. (2021). *An Optimized Nature-Inspired Metaheuristic Algorithm for Application Mapping in 2D-NoC*. 1–21.

Suovanen, J. (2020). *Vue éclatée du Samsung Galaxy S7*.

Tan, L. X., & Guo, L. (2009). Quantum and biogeography based optimization for a class of combinatorial optimization. *2009 World Summit on Genetic and Evolutionary Computation, 2009 GEC Summit - Proceedings of the 1st ACM/SIGEVO Summit on Genetic and Evolutionary Computation, GEC'09*, *2*(1), 969–972. https://doi.org/10.1145/1543834.1543986

Tanenbaum, A. . ., & Wetherall, D. J. (2010). *Computer Network* (5 edition). Prentice Hall.

Tatas, K., Siozios, K., Soudris, D., & Jantsch, A. (2014). Designing 2D and 3D network-on-chip architectures. In *Designing 2D and 3D Network-on-Chip Architectures* (Vol. 9781461442). https://doi.org/10.1007/978-1-4614-4274-5

TOUATI, L. (2012). Développement d'une plateforme hétérogène de mapping dans les réseaux sur puce (Nocs). In *Ecole supérieure d'informatique ESI*. Institut National de formation en Informatique.

TOUIMER, N. (2022). *Mapping dans les réseaux sur puce 3D utilisant l ' algorithme basé sur les corbeaux*. ECOLE NORMALE SUPERIEURE Vieux – kouba.

Vallerio, K. (2008). Task Graphs for Free (TGFF v3 . 0). *Processing*, 1–9.

Vasic, M. (2014). *Physical design of a 3D router: reducing the number of vertical connections and enabling asynchronous operation*. University of Technology.

Wang, C., Hu, W. H., & Bagherzadeh, N. (2011). A wireless network-on-chip design for multicore platforms. *Proceedings - 19th International Euromicro Conference on Parallel, Distributed, and Network-Based Processing, PDP 2011*, 409–416. https://doi.org/10.1109/PDP.2011.37

Wang, S., & Jin, T. (2014). Wireless network-on-chip: a survey. *The Journal of Engineering*, *2014*(3), 98–104. https://doi.org/10.1049/joe.2013.0209

Wei,song; Guangda, Z. (2022). *Asynchronous on-chip networks and faut-tolerant tchniques* (Taylor & F). Taylor & Francis. http://taylorandfrancis.com

Xiaodong, W., Yi, L., & Yintang, Y. (2020). *Network-on-chip heuristic mapping algorithm based on isomorphism elimination for NoC optimisation*. 272–280. https://doi.org/10.1049/iet-cdt.2019.0212

Yeddou, I. (2020). *using grey wolf algorithm for the mapping problem in NoC3D*. High

School of Teachers , KOUBA – ALGIERS.

Yin, S., Hu, Y., Zhang, Z., Liu, L., & Wei, S. (2012). *Hybrid Wired / Wireless On-Chip Network Design for Application-Specific SoC ∗*. *4*, 495–505.

Zerioul, L. (2016). *Modélisation comportementale d'un réseau sur puce basé sur des interconnexions RF Lounis*. Cergy Pontoise (UCP),.

Zimmermann, H. (1980). OSI Reference Model - The ISO Model of Architecture for Open Systems Interconnection. *IEEE Transactions on Communications*, *28*(4), 425–432. https://doi.org/10.1109/TCOM.1980.1094702