Approches d'optimisation et de personnalisation des réseaux sur puce (NoC : Networks on Chip)

Abderrahim Chariete

SPIM

Thèse de Doctorat

école doctorale sciences pour l'ingénieur et microtechniques
UNIVERSITÉ DE TECHNOLOGIE BELFORT-MONTBÉLIARD

Approches d'optimisation et de personnalisation des réseaux sur puce (NoC : Networks on Chip)

Abderrahim CHARIETE

SPIM

Thèse de Doctorat

école doctorale **sciences pour l'ingénieur et microtechniques**

UNIVERSITÉ DE TECHNOLOGIE BELFORT-MONTBÉLIARD

N° |_|_|_|

THÈSE présentée par

Abderrahim CHARIETE

pour obtenir le

Grade de Docteur de
l'Université de Technologie de Belfort-Montbéliard

Spécialité : **Informatique**

Approches d'optimisation et de personnalisation des réseaux sur puce (NoC : Networks on Chip)

Soutenue le 11 mars 2014 devant le Jury :

Mr. Maxime WACK	Directeur de thèse	HDR-MCF à l'UTBM
Mr. Pascal LORENZ	Rapporteur	Professeur à l'UHA de Colmar
Mr. Bruno ROUZEYRE	Rapporteur	Professeur à l'UM2
Mr. Michel HASSENFORDER	Examinateur	Professeur à l'UHA de Mulhouse
Mr. Jaafar GABER	Examinateur	MCF à l'UTBM
Mr. Mohamed BAKHOUYA	Examinateur	HDR-Professeur associé à l'UIR

Remerciements

Ce travail a été réalisé au sein du laboratoire Systèmes et Transports (SeT), fédéré par l'Institut de Recherche sur les Transports, l'Energie et la Société (IRTES) et Sous la tutelle de l'Université de Technologie de Belfort-Montbéliard (UTBM). Que l'institution trouve ici l'expression de ma forte reconnaissance.

De manière moins formelle, je tiens à témoigner de ma gratitude au directeur de l'équipe de recherche à laquelle j'appartiens et directeur de ma thèse, Mr. M. WACK. Il a su m'écouter et m'encourager ; il a montré patience et compréhension. La thèse terminée, j'espère avoir été à la hauteur de la confiance qu'il a mis en moi. C'était un honneur pour moi d'accomplir cette thèse de doctorat sous sa supervision.

Je tiens à remercier Mr. M. BAKHOUYA qui se trouve à l'origine de ce sujet de thèse. Il a suivi de près toutes les phases de son élaboration ; il a été présent durant les péripéties de sa rédaction. Tout en m'accordant la liberté d'envisager et d'explorer des pistes non prévues lors de la formulation initiale du sujet, il a su me ramener régulièrement sur le chemin de l'efficacité. Pour ses conseils et son soutien, le temps et les efforts qu'il a passé pour vérifier et interpréter les résultats de simulation. Qu'il soit remercié.

Je remercier également Mr. J. GABER pour être toujours serviable et attentionné, j'ai eu une excellente occasion d'apprendre beaucoup de choses de lui dans les réunions et les séances de travail. Sans son aide, les résultats de simulation n'ont pas étais possibles dans un si petit laps de temps. Pour tout ça, qu'il trouve ici l'expression de ma reconnaissance.

Mes remerciements vont encore à Mr. A. CAMINADA, directeur du département informatique, ainsi que tous les enseignants/chercheurs et les doctorants du SeT qui ont étaient à l'écoute pour répondre à mes différentes questions.

Je remercie également Messieurs : le Pr. B. ROUZEYRE, le Pr. P. LORENZ et le Pr. M. HASSENFORDER, pour avoir consacré une partie de leur temps à mon travail en acceptant de rapporter et/ou de faire partie du jury de cette thèse.

Enfin, je remercie très chaleureusement mes parents, ma femme et mon fils Yasser, ainsi que toute la famille pour leur soutien moral et sentimental. Plus particulièrement, mon père et mon frère Issam qui ont accepté de lire avec soin l'intégralité de la thèse. Pour leurs remarques et leurs encouragements, leurs approbations et leurs critiques, leur disponibilité enfin, qu'ils soient remerciés.

Bien que S. BIRI n'a absolument pas lit une seul page, il reste mon alter ego le plus fort et le plus sérieux dans plusieurs domaines. Pour sa camaraderie intellectuelle, qu'il soit remercié. Je remercie également tous mes amis pour leur soutien et leur encouragement sans faille qui m'a fait momentanément oublier la monté de stresse avant le jour de soutenance. Ils sont tellement nombreux que je me trouve dans l'impossibilité de les remercier autrement que de manière collective.

Dédicaces

Je dédie ce travail à ma tendre Maman et à mon cher Papa

Puis à mon ange fils Yasser et à ma précieuse Femme

A mes deux sœurs et cinq frères et à toute la famille

A tous mes amis et mes collègues

Table des matières

I. Introduction générale

IV.1. Domaine abordé

Contexte général : Optimisation de performances des réseaux sur puce (*NoC*: *Network-on-Chip*)

Contexte précis :

- Développement d'un outil d'évaluation de NoC qui combine l'évaluation analytique et la simulation.
- Mise en œuvre de plusieurs approches adaptatives pour la conception des réseaux NoCs :
  - une approche de personnalisation de la topologie du NoC sans tenir compte de l'application.
  - une approche d'optimisation des ressources NoC, qui permet d'allouer dynamiquement les espaces tampons selon une application spécifique.
  - une approche de contrôle de flux pour détecter et éviter les congestions dans les tampons.

IV.2. Problématique traitée

Les systèmes intégrés sur puce (*SoC: System on Chip*) ont émergé comme une technologie clé pour la pluparts des systèmes embarqués et les systèmes miniaturisés intelligents (*smart miniaturized systems*), notamment pour les télécommunications et les applications multimédias, pour offrir une grande flexibilité et une meilleure performance. Grâce à l'évolution de la technologie des circuits intégrés (la technologie silicium sur isolant [1] (*SOI*: *Silicon On Insulator*)) et des performances toujours croissantes des systèmes électroniques, les SoCs sont devenus de plus en plus complexes. C'est-à-dire, les concepteurs des SoCs embarquent de plus en plus de composants IPs (*IPs*: *Intellectual property*), pour répondre aux besoins de nouvelles applications. Ce qui se traduit par une augmentation des communications (échanges de données, contrôle de flux, etc.). Par conséquent, la qualité de l'architecture d'interconnexion devient primordiale pour la performance des SoCs. Donc, il est désormais nécessaire de proposer des solutions alternatives efficaces de systèmes intégrés, afin que la communication sur puce réponde à la qualité de service exigée et que la performance globale du système soit améliorée.

Pour s'adapter à cette hausse de complexité, le NoC a donc émergé comme une toile de communication pervasive pour connecter les ressources IPs. Il est la solution alternative aux systèmes non-évolutifs classiques généralement basés sur des bus partagés ou des crossbars [2] [3], qui ne supportent pas les débits élevés, manquent de flexibilité et ne seront pas adaptées pour les systèmes futurs implémentant plusieurs centaines de composants IPs. Ce type d'architecture, le NoC, présente de réels atouts au niveau de la conception, de la performance et de l'implémentation. Les cinq principales caractéristiques du réseau sur puce sont ; la technique de commutation, le contrôle de flux, la topologie, le mécanisme de routage et les paramètres d'évaluation. Plusieurs approches ont été proposées pour la conception de réseau sur puce (NoC). Elles peuvent être classées en deux catégories principales, des approches au moment de la conception qui sont généralement adaptées pour un domaine d'application spécifique fournissant ainsi un NoC personnalisé [4] [5] [6], et des approches au moment de l'exécution qui fournissent des techniques qui permettent à un NoC d'adapter en permanence sa structure et son comportement lors de l'exécution. Différentes architectures NoC basées sur la commutation de paquets ont été étudiées et adaptées pour les systèmes sur puce. Par exemple, Fat-Tree (FT), Butterfly-Fat Tree (BFT), 2D Mesh, 2D Torus, Ring, Spidergon, Octagon, WK, etc. [3] [7].

Dans les SoCs modernes, l'infrastructure de communication sur puce ou le réseau sur puce (NoC) est le facteur dominant pour la conception, la validation et l'analyse de performances. Il

est donc très important d'obtenir une bonne compréhension de la performance du système au préalable, c'est-à-dire avant que le système soit implémenté en détail et construit. Par conséquent, les modèles de performance abstraits ont été déployés dans la conception du système pendant de nombreuses décennies et ont, plus récemment, été adoptés pour l'étude des réseaux de communication sur puce. Pour faire face à l'incertitude dans les applications NoC, les architectures et les paramètres du système, tels que l'algorithme de routage et la technique de commutation devrait être adaptées lors de l'exécution. Par exemple, vu que les ressources sont partagées, la congestion ou les goulets d'étranglement peuvent être créés dans certains commutateurs, conduisant donc à une mauvaise performance. Lorsque le réseau est congestionné, des techniques sont nécessaires pour permettre au NoC de réacheminer/re-router le trafic, au moment de l'exécution, dans la zone congestionnée (ex., le cas du modèle de trafic *HotSpot*), ou bien de changer dynamiquement la bande passante des liens congestionnés. Les principaux objectifs sont donc d'utiliser efficacement les ressources par reconfiguration dynamique du NoC pour satisfaire les exigences de qualité de service, c'est-à-dire de minimiser le coût en silicium et de minimiser la consommation d'énergie en séparant les communications du calcul [8] [9] [10] [11] [12]. Bien que ces architectures NoC s'appuient sur des concepts hérités des systèmes parallèles et distribués pour interconnecter des cœurs IPs d'une manière structurée et évolutive, elles peuvent souffrir d'une latence élevée, d'un faible débit, d'une grande surface silicium et d'une forte consommation d'énergie lorsqu'elles sont adaptées pour les applications SoC. Par rapport aux systèmes parallèles et distribués, les systèmes sur puce actuels ont des ressources limitées et devraient être implémentées avec un très faible coût en surface silicium, les liens de communication dans le NoC sont relativement plus courts que ceux des réseaux informatiques. Ainsi, la conception d'architectures de communication sur puce flexibles, évolutives et fiables qui répondent aux exigences des applications SoC actuelles est nécessaire.

Les concepteurs des SoCs qui se sont intéressés à l'évaluation de la performance ont comme but ; soit de fournir la plus haute performance à un coût donné, ou bien de fournir un niveau minimal de performance au minimum coût possible. Dans les deux cas, une mesure fiable de la performance est indispensable. Ces outils d'estimation de performance peuvent être classés en modèles de simulation et modèles analytiques [3] [5] [14] [15] [16] [17]. Les concepteurs des SoCs ont abordé l'analyse de performance en explorant l'espace de conception par l'utilisation de la simulation détaillée. Les outils de simulation sont souples et précis, mais souvent ils sont complétés par des approches de modélisation analytique de la performance. En outre, et comme complément aux techniques à base de simulation, des modèles mathématiques ont été utilisés pour modéliser et évaluer la performance de communication. En particulier, les modèles analytiques peuvent analyser les pires-case. En général, les simulations ont besoin de modèles exécutables et sont adaptées pour étudier les effets dynamiques et sporadiques en révélant des résultats plus réalistes pour l'évaluation des moyens-cas [18]. Cependant, la simulation prend du temps et fournit peu d'indications sur la façon dont les différents paramètres de conception influent la performance réelle du NoC. Par ailleurs, les modèles analytiques peuvent permettre une évaluation rapide des mesures de performance de grands systèmes dans le processus initial de conception, mais ils ne considèrent que l'évaluation des pires-cas. Plusieurs travaux, telle que présentée dans [19], ont démontré qu'il existe un besoin crucial d'outils et de méthodologies de conception de systèmes pour évaluer analytiquement les architectures NoC.

Avec la nécessité croissante de la demande de services, il est difficile de concevoir des NoC qui sont capables de satisfaire toutes les exigences des applications et de prédire des paramètres dans les premiers stades de développement (c.à.d. le modèle de trafic n'est pas connu ou prévisible à l'avance). Par exemple, les futurs terminaux mobiles de communication devraient supporter de nombreuses applications, allant des applications de navigation Web aux applications multimédias en temps réel, comme la communication audio/vidéo. Ainsi, la conception de ces systèmes devrait être très souple et adaptable, tout en offrant une haute performance et une faible consommation d'énergie.

Dans cette thèse, nous avons étudié les questions architecturales liées aux réseaux d'interconnexion sur puce. Les principales questions qui devraient être abordées lors de la conception d'un réseau d'interconnexion sur puce peuvent être classés en deux catégories : qualitatifs (ex., l'extensibilité et la fractalité, la régularité, la symétrie, etc.) et les aspects quantitatifs (ex., l'énergie, la superficie, la charge, le débit, la latence, etc.). Pour répondre à ces questions, en premier lieu, des recherches sont nécessaires pour évaluer les architectures existantes. Deuxièmement, ces architectures doivent être évaluées en termes de performance de communication, d'énergie et de superficie en silicium. Des approches d'optimisation de performance des architectures NoCs et de contrôle de communication et de congestion doivent être étudiées. Pour permettre une évaluation réaliste des approches et des architectures proposées, un simulateur et un ensemble de méthodes et outils d'analyse et d'évaluation doivent être élaboré. Enfin, nous étudierons la possibilité d'adapter dynamiquement l'architecture du NoC à l'évolution des conditions du fonctionnement.

IV.3. Contribution de la thèse

L'objectif principal de cette thèse est d'étudier de nouvelles approches pour l'inter-connectivité des couches physique et liaison des réseaux sur puces, en tenant compte des performances du NoC, notamment la latence, le débit, la consommation d'énergie et la simplicité de la mise en œuvre.

Dans un premier temps, un travail bibliographique a été mené sur les approches d'optimisation des réseaux d'interconnexions NoC et les différents problèmes liés à la conception de systèmes intégrés sur puce. De nombreuses études ont montré que pour améliorer les performances et réduire la consommation d'énergie pour un domaine d'applications spécifique, l'architecture du réseau pouvait être personnalisée par l'insertion de nombreux liens entre les routeurs. Dans ce contexte, nous avons porté un intérêt particulier à l'ajout de longs liens de communication (long links insertion), cette technique est assez bénéfique aux communications entre les diverses parties des SoCs. L'idée est commencer par un NoC régulier (ex., 2D Mesh) et de l'optimiser/personnaliser par insertion de longs liens entre des commutateurs stratégiques, sous la contrainte d'un budget limité en termes de nombre de liens. Nous avons évalué notre approche analytiquement et par simulation en considérant plusieurs types de NoC, tels que 2D Mesh, 2D Torus, 2D X-Mesh, Ring, Spidergon et WK, et plusieurs types d'applications, telles que *HotSpot*, *Uniform*, *Transpose*, *Bit-Reversal* et *Shuffle*. L'évaluation analytique porte sur l'optimisation des paramètres physiques de l'architecture du NoC, tels que minimiser la distance moyenne, minimiser le diamètre et maximiser le dégrée de clusterisation, sans tenir compte de l'application qui devrait s'exécuter sur le NoC. L'optimisation des paramètres physiques de l'architecture du NoC devrait impacter ces performances de communication, tels que minimiser la latence, la consommation d'énergie et la charge des liens, et de maximiser le débit. L'évaluation de cette approche par simulation est effectuée en utilisant le simulateur Nirgam, qui est dédié aux réseaux NoC. Les résultats obtenus ont montré l'efficacité de notre approche et ont fait l'objet de plusieurs publications :

- *«An Approach for Customizing On-chip Interconnect Architectures in SoC Design». International Conference on High Performance Computing & Simulation (HPCS'2012), Madrid, 2012.*
- *«FracNoC: a Fractal On-Chip Interconnect Architecture For System-on-Chip». International Workshop on High Performance Interconnection Networks, In International Conference on High Performance Computing & Simulation (HPCS-HPIN'2013), Helsinki, 2013.*
- *«A methodology for customizing on-chip interconnect architectures». Concurrency and Computation: Practice & Experience Journal, 2013* (*accepted with revisions*).

Dans un second temps, une étude bibliographique sur le thème de la gestion des ressources dans les réseaux d'interconnexions sur puce nous a conduit à nous intéresser au cas particulier de l'allocation optimisée de l'espace des tampons. Utiliser des tampons avec une taille fixe génère de la congestion au niveau des routeurs quand le trafic est dense, ce qui augmente la consommation énergétique et influence la performance du NoC. Cependant, nous avons introduit une technique de personnalisation dynamique de la taille des tampons selon la variation des flux d'une application spécifique, permettant ainsi d'obtenir une augmentation de la bande passante des liens congestionnés. Ce travail a donné lieu à une publication :

- *«A Buffer Allocation Approach for Application-specific Network-On-Chip». 9th IEEE/ACS International Conference on computer systems and applications (AICCSA'2011), Sharm-el-cheik, 2011.*

Enfin, nous avons proposé une approche conjointe basée sur la modélisation de flux du NoC par réseau à compartiments pour la gestion dynamique de la congestion. Cette approche permet aux éléments du NoC d'ajuster dynamiquement les flux en utilisant un mécanisme de contrôle des flux basé sur la rétroaction de flux. Les résultats d'analyse et de simulation montrent la viabilité de ce mécanisme pour éviter la congestion dans les réseaux NoC. Ces résultats ont donné lieu aux publications suivantes :

- *«An Adaptive Approach for Congestion Avoidance in Network-on-Chip». 20th Euromicro International Conference on Parallel, Distributed and Network-Based Computing (PDP'2011), Rosenheim, 2011.*
- *«Performance Evaluation of a Flow Control Algorithm for Network-on-Chip». International Conference on High Performance Computing & Simulation (HPCS'2012), Madrid, 2012.*
- *« A System Dynamics Approach for Congestion Avoidance in Network-on-Chip ». Soumis au Journal of Systems Architecture (JSA), 2013.*

IV.4. Organisation du document

IV.4.1. Présentation générale

La thèse est structurée en quatre chapitres principaux, plus une introduction générale et des conclusions & perspectives. L'introduction générale met en avant la problématique traitée dans cette thèse ainsi que les principaux objectifs et contributions de la thèse. Le premier chapitre présente un panorama exhaustif sur les concepts et les protocoles des réseaux sur puces. Le deuxième chapitre présente un large état de l'art sur les travaux de recherche réalisés sur les approches de personnalisation des NoCs, sur les méthodes et les outils d'évaluation des NoC et sur les architectures NoC existantes. Dans le troisième chapitre, deux approches de personnalisation de l'architecture du NoC sont présentées, l'une basée sur l'insertion de longs liens entre des commutateurs stratégiques, et l'autre basée sur l'allocation dynamique et optimisée de l'espace tampon des commutateurs congestionnés. Dans le quatrième chapitre, nous présentons une approche dynamique de contrôle des flux dans les NoCs. Enfin, les conclusions et les perspectives de cette thèse sont présentées.

IV.4.2. Plan: vue globale

- Introduction générale
- Chapitre 1: Les réseaux sur puce : concepts et protocoles
- Chapitre 2: Etat de l'art : Approches d'optimisation et outils d'évaluation des NoCs
- Chapitre 3: Approches de personnalisation des NoCs
- Chapitre 4: Une approche dynamique de contrôle du flux dans les NoCs

- Conclusions & perspectives

IV.4.3. Contenu des chapitres

A. Introduction générale

Nous avons commencé cette introduction par décrire l'évolution des architectures NoCs, puis nous avons présenté les types d'approches de conception de ces architectures. Ensuite, nous avons présenté les différentes architectures proposées dans la littérature et les paramètres de performance. Enfin, nous avons présenté les tendances actuelles dans ce domaine et les problématiques rencontrées ainsi que nos objectifs pour y remédier à ces problématiques.

B. Chapitre 1 : Les réseaux sur puce : concepts et protocoles

Ce chapitre donne une large compréhension du fonctionnement des réseaux sur puce. Nous présentons donc les principaux mécanismes et aspects relatifs aux NoCs. Nous commençons par une présentation rapide de l'évolution des réseaux sur puce à commutation de paquets. Les paramètres architecturaux ainsi que les principaux composants du NoC sont ensuite décrits. Puis, nous présentons les concepts/protocoles de communication et les différents types d'algorithmes de routage du NoC, empruntés aux réseaux informatique et des systèmes parallèles. Enfin, nous présentons les détails des paramètres de performance utilisés pour l'évaluation des réseaux NoCs.

C. Chapitre 2 : Etat de l'art : Approches d'optimisation et outils d'évaluation des NoCs

Dans ce chapitre, nous commençons par présenter un état de l'art des travaux de recherche concernant les approches de personnalisation et d'optimisation des réseaux NoCs et nous proposons une classification de ces approches selon trois niveaux, physique, communication et application. Dans la suite, nous présentons un état de l'art sur les méthodes et les outils d'évaluation/expérimentation des réseaux NoCs, et on distingue deux grandes catégories : les méthodes d'expérimentation par modèles de systèmes, incluant les modèles analytiques et les modèles de simulation, et les méthodes d'expérimentation par systèmes existants. Dans la partie des méthodes d'évaluation par modèles de simulation, nous présentons une étude exhaustive et comparative des simulateurs NoC proposés dans la littérature. Enfin, nous présentant une synthèse de nos choix de méthodes d'évaluation analytique et par simulation, ainsi que le positionnement de nos approches d'optimisation des NoCs par rapport aux approches existantes dans la littérature, selon la classification présentée dans ce chapitre.

D. Chapitre 3 : Approches de personnalisation des NoCs

Dans ce chapitre, nous présentons deux approches de personnalisation de l'architecture NoC : la première approche concerne la personnalisation des architectures NoCs, elle est basée sur l'insertion de longs liens entre des commutateurs stratégiques, dont le but est d'améliorer la performance, minimiser la consommation d'énergie et de minimiser la surface en silicium du NoC candidat. Cette approche permet aux concepteurs d'adapter un réseau NoC donné avec différentes applications SoCs. Elle est modélisée par un algorithme qui permet tout d'abord la sélection de liens qui optimisent les paramètres physiques de l'architecture du NoC, puis l'insertion de ces liens sur ce NoC, sous la contrainte d'un budget limité. Ensuite, différents modèles de trafic peuvent être exécutés sur ce NoC pour évaluer l'impact de l'optimisation de son architecture sur ces performances de communication (ex., la latence, le débit, l'énergie, etc.). Cette approche concerne donc la personnalisation du réseau du NoC en fonction des propriétés physiques de sa topologie, qui pourraient supporter un maximum d'applications SoCs. La deuxième approche concerne la modélisation de trafic NoC par réseau à compartiments, pour l'évaluation

analytique des architectures d'interconnexion sur puce. L'objectif principal de cette approche de modélisation est de permettre aux concepteurs, d'allouer dynamiquement les espaces tampons qui conviennent le mieux aux besoins d'une application spécifique.

E. Chapitre 4 : Une approche dynamique de contrôle du flux dans les NoCs

Dans ce chapitre, nous introduisons une approche de contrôle de flux pas à pas dans le NoC. Ce mécanisme de contrôle de flux est basé sur la théorie des systèmes dynamique ainsi que la modélisation de trafic par réseau à compartiments présenté dans le chapitre précédent. L'analyse des caractéristiques du trafic de communication sur puce permet de connaître à l'avance le modèle de trafic. C'est ainsi que le concepteur peut allouer plus de ressources, en mémoire tampon, à des commutateurs congestionnés. Cette approche pourrait être utilisée au moment de l'exécution pour éviter au réseau NoC d'être congestionné et ainsi d'éviter les débordements de tampons.

F. Conclusions et perspectives

Dans ce paragraphe, nous présentons les conclusions de cette thèse ainsi que nos perspectives.

II. Chapitre 1 : Les réseaux sur puce (concepts et protocoles)

Ce chapitre introduit au fonctionnement des réseaux sur puce (NoCs) et présente les principaux mécanismes et aspects relatifs aux NoCs. Nous commençons par une présentation rapide de l'évolution des réseaux sur puce à commutation de paquets. Puis, nous présentons les caractéristiques de l'architecture du NoC empruntées des systèmes parallèles et des réseaux informatiques, ainsi nous présentons les principaux composants du NoC. Ensuite, nous présentons les concepts et les protocoles de communication dans les NoCs. Enfin, nous présentons les détails des paramètres de performance utilisés pour l'évaluation des NoCs.

V.1. Evolution des réseaux sur puce à commutation de paquets

V.1.1. Interconnexion point à point

La toute première façon de concevoir l'infrastructure de communication pour les systèmes sur puce était l'utilisation directe de l'interconnexion point à point entre les composants matériels du système appelés IPs (*Intellectual Property*). Ceci permet aux IPs de communiquer directement à travers des liens dévolus sans la nécessité d'arbitrage centralisé. Pour un système avec un grand nombre de composants IPs, cette architecture de communication nécessite beaucoup de broches (*PINs*) pour chaque IP, une longue durée de routage et un câblage complexe.

De même, les délais et la qualité des signaux deviennent imprévisibles dans le système lorsque des interconnexions directes sont utilisées pour la communication. Cela rend très difficile de tester le système. En raison de ces inconvénients, les infrastructures d'interconnexion point à point ne permettent qu'une faible évolutivité, une faible utilisation de l'acheminement des ressources et une très faible possibilité de réutilisation.

D'autre part, un SoC conçu avec un petit nombre d'IPs basé sur ce type d'interconnexions est susceptible de donner une meilleure performance [20]. Un exemple de SoC conçu avec des connexions point à point est illustré par la figure 1.1.

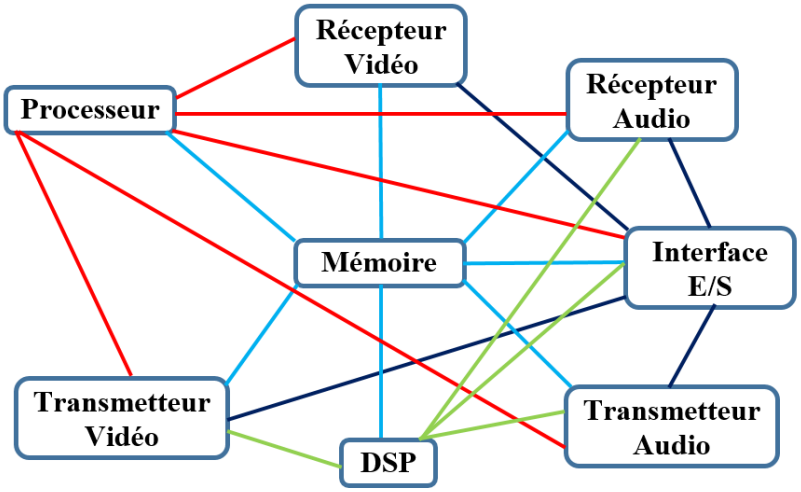


Figure 1.1 : SoC basé sur l'infrastructure de communication avec des connexions point à point

V.1.2. Intégration basée sur le bus partagé

La communication inter-IPs de la majorité des systèmes sur puce est conçue avec des bus en fonction de l'infrastructure de communication. Dans ce type de communication, tous les IPs partage un ou plusieurs bus. Les IPs sont connectés au bus par le biais d'une interface. Un arbitre du bus gère la communication et le conflit entre les IPs (cf. figure 1.2). En architecture à base de bus, les IPs nécessitent moins de broches d'E/S par rapport à une architecture avec des interconnexions point à point. De même, le coût et la surface de câblage nécessaire pour la communication sont également réduits.

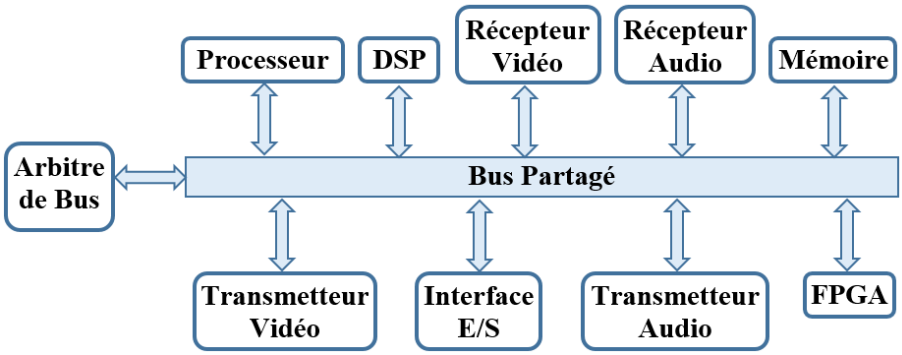


Figure 1.2 : SoC basé sur l'infrastructure de communication en bus partagé

Dans la littérature, il y a plusieurs propositions pour une utilisation efficace des bus comme par exemple le bus hiérarchique, le bus segmenté et le bus pipeline, etc. Malgré ces propositions efficaces et les avantages mentionnés ci-dessus, le bus partagé ne s'adapte pas au-delà d'une certaine limite en fonction du nombre d'IPs. En outre, le conflit et l'arbitrage dans le bus ralentit également le transfert de données. L'arbitre donne la main aux maîtres en fonction d'une stratégie d'arbitrage qui peut être plus ou moins complexe, à tour de rôle (Round Robin), avec gestion des priorités ou avec des tranche de temps (*Time-Out*). La complexité et les performances de l'arbitre dépendent du nombre de maîtres à gérer et de la stratégie choisie. Un exemple de SoC intégré sur un bus partagé est illustré par la figure 1.2.

V.1.3. Intégration basée sur le crossbar

Dans un bus partagé, si une ressource IP de type maitre communique avec une autre ressource IP de type esclave aucune autre ressource IP de type maitre ne peut utiliser le bus même si elle veut communiquer avec une autre ressource IP de type esclave. Une architecture crossbar permet d'avoir un chemin reliant n'importe quel couple de ressources IPs maitre-esclave, par exemple CPU-CPU, CPU-Mémoire, ou Mémoire-Mémoire, comme le montre la figure 1.3 (*CPU*: *Central Processing Unit*). Comme dans l'exemple de la figure 1.3, tous les couples sont interconnectés par une grille de commutateurs crosspoint, ces commutateurs sont activés ou désactivés selon le trafic généré par les ressources IPs. Comparé à un bus partagé, le crossbar offre une meilleure bande passante et donc une meilleure latence du fait que les chemins maitres-esclaves sont séparés les uns des autres.

Par contre, le nombre de connexions physiques nécessaires à l'implémentation d'un crossbar complet limite grandement les domaines où il peut être appliqué. Le crossbar n'est plus adapté au-delà d'un certain nombre d'IPs [21], puisque il devient très couteux en termes de ressources (liens et commutateurs), il est donc réservé à des cas très particuliers d'application.

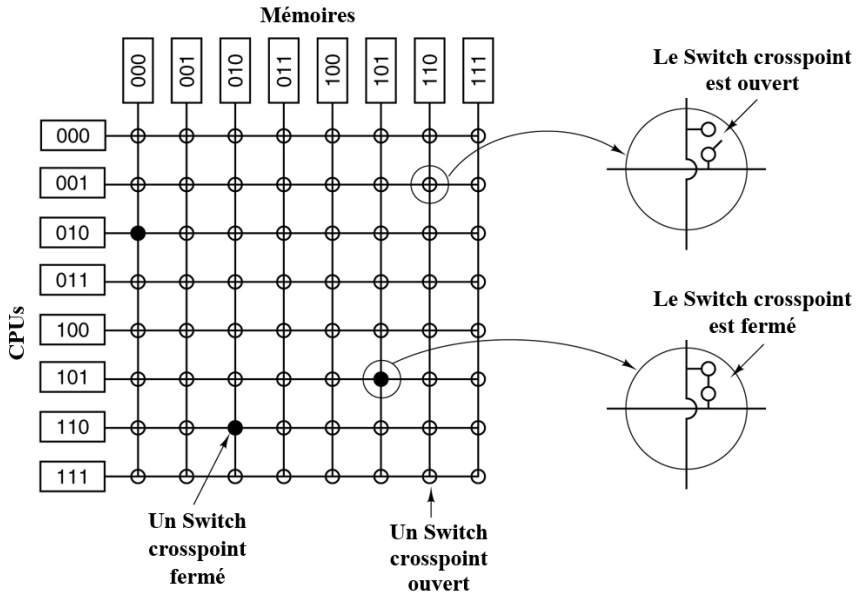


Figure 1.3 : SoC basé sur l'infrastructure de communication en crossbar

V.1.4. 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 la très peu ou pas de réutilisation, une nouvelle méthode est nécessaire pour la communication entre les IPs des systèmes sur puce SoCs. Les réseaux à commutation de paquets sont proposés comme une solution alternative efficace pour la conception de la communication entre les IPs dans les SoCs [2] [20] [22] [23] [24] [25].

Un réseau à commutation de paquets est constitué d'un réseau de commutateurs/routeurs. Les ressources IPs dans le réseau sont connectées aux routeurs à travers l'interface ressource-réseau (IRR), en anglais *Ressource-Network Interface* (RNI). Les données circulent à partir de la source vers la destination sous forme de paquets formatés traversant un ou plusieurs routeurs dans le réseau. Ce type d'infrastructure de communication sur puce est hautement évolutif. Il offre également une grande possibilité de réutilisation, réduisant ainsi les délais de commercialisation et le coût du système.

L'infrastructure de communication à base de commutation de paquets pour les systèmes sur puce est représentée en figure 1.4. De même, un exemple de SoC basé sur une topologie NoC 4x4 2D Mesh est illustré dans la figure 1.7.

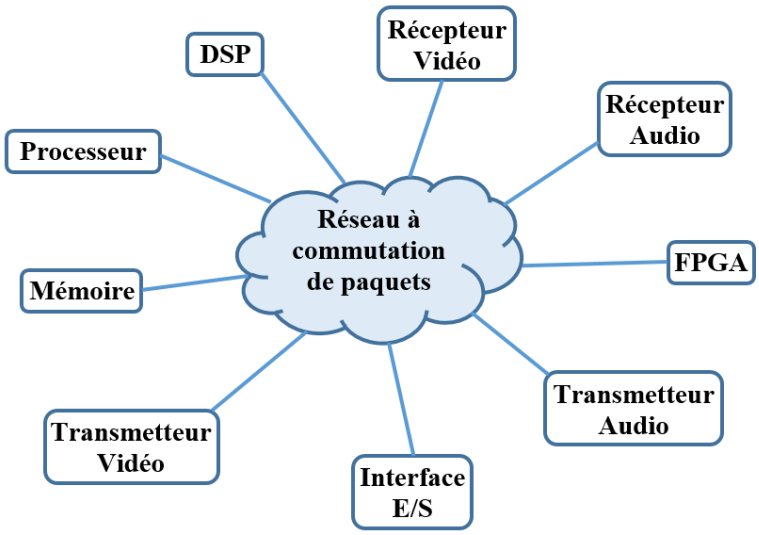


Figure 1.4 : SoC basé sur l'infrastructure de communication en réseau à commutation de paquets

V.2. Architecture du NoC

Les protocoles utilisés dans le NoC sont généralement des versions simplifiées des protocoles de communication utilisés dans les réseaux informatiques. De même, la plupart des définitions qui sont utilisées dans les réseaux de communication informatiques sont également applicables dans le NoC. Certains de ces concepts et définitions sont abordées dans les paragraphes suivants.

V.2.1. 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. Pour simplifier l'implémentation du protocole, celui-ci est divisé en différentes couches avec des fonctionnalités spécifiques. Le modèle OSI classique (*Open Systems Interconnection*) comporte sept couches appelées couches OSI, telles que, de bas en haut, la couche physique, la couche liaison de données, la couche réseau, la couche transport, la couche session, la couche présentation et la couche application (cf. figure 1.5).

Une couche peut être considérée comme une combinaison de composants logiciels ou matériels effectuant quelques fonctionnalités. Chaque couche exécute sa tâche indépendamment des autres couches et fournit des services à la couche supérieure et obtient les services des couches inférieures. La manière dont une couche exerce sa fonction est masquée par les autres couches. Les couches communiquent entre elles via des interfaces standards. Chaque couche met en œuvre des règles différentes appelées protocoles.

La représentation des couches du modèle OSI classique n'est pas adaptée à la description d'un réseau NoC, car contrairement aux réseaux informatiques où les protocoles sont complexes vu qu'ils sont utilisés pour une topologie inconnue, celui-ci n'intègre pas toutes ces divisions en couches. En NoC, principalement quatre couches sont considérés telles que : la couche physique, la couche liaison de données, la couche réseau et la couche transport. Les principes des couches session, présentation et application sont également pris en charge par les ressources IPs du SoC.

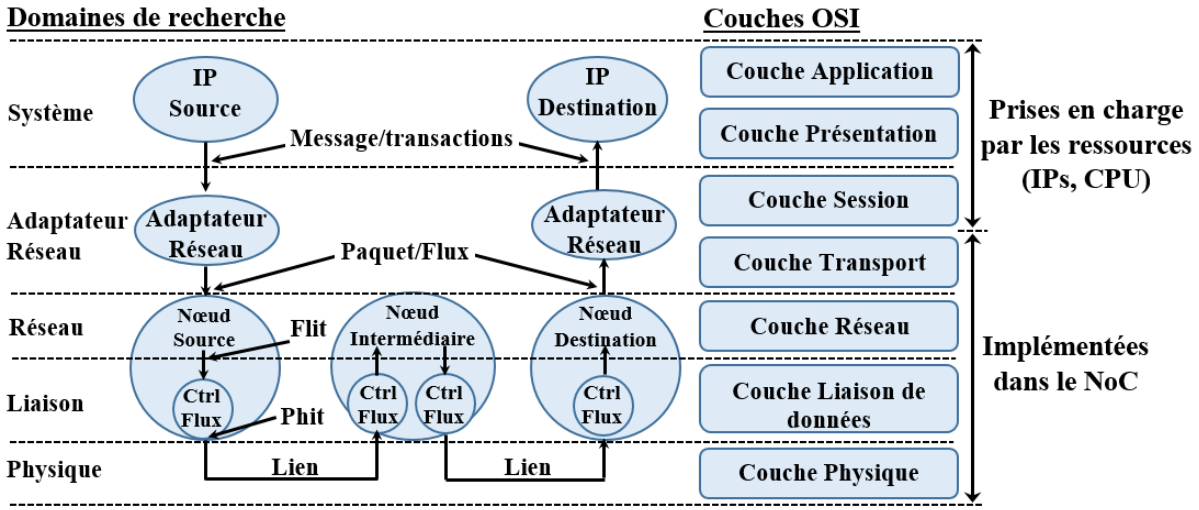


Figure 1.5 : Les éléments du NoC et les couches réseau

Comme il n'existe pas de modèle standardisé pour représenter les couches réseau d'un NoC. Nous pouvons utiliser le modèle représenté par la figure 1.5 qui est adapté de [26] [27]. Cette figure nous montre le flot d'une donnée depuis une IP source vers une IP destinataire au travers des éléments d'un NoC. Elle indique de plus la correspondance avec les domaines de recherches et les couches réseau du modèle OSI [28].

Dans cette thèse, nous nous sommes concentrés principalement sur des approches de personnalisation de l'architecture d'interconnexion sur puce basées sur la couche physique et la couche réseau. Quelques aspects de la couche liaison de données sont également considérés.

V.2.2. Rôles des couches OSI dans le NoC

Nous présentons ci-dessous le rôle que les différentes couches OSI jouent dans le NoC.

V.2.2.1. La couche application

C'est dans cette couche que se trouvent les ressources (IPs). Il est important de noter qu'une ressource IP peut effectuer de nombreux processus différents. Par exemple, un microprocesseur peut exécuter plusieurs processus simultanément.

V.2.2.2. La couche présentation

Comme les ressources IPs peuvent avoir différentes représentations de données, il doit y avoir certaines opérations de conversions entre elles. Par exemple, les entiers peuvent être représentés dans un format *big-endian* ou dans un format *little-endian* (cf. figure 1.6). Un autre exemple pourrait être le cas d'un microprocesseur utilisant un processeur de signal pour accélérer les calculs. Supposons que le microprocesseur utilise une représentation en virgule flottante et le processeur de signal utilise un format de point fixe. Donc, la conversion entre ces formats est nécessaire. Cette fonctionnalité est localisée dans la couche de présentation.

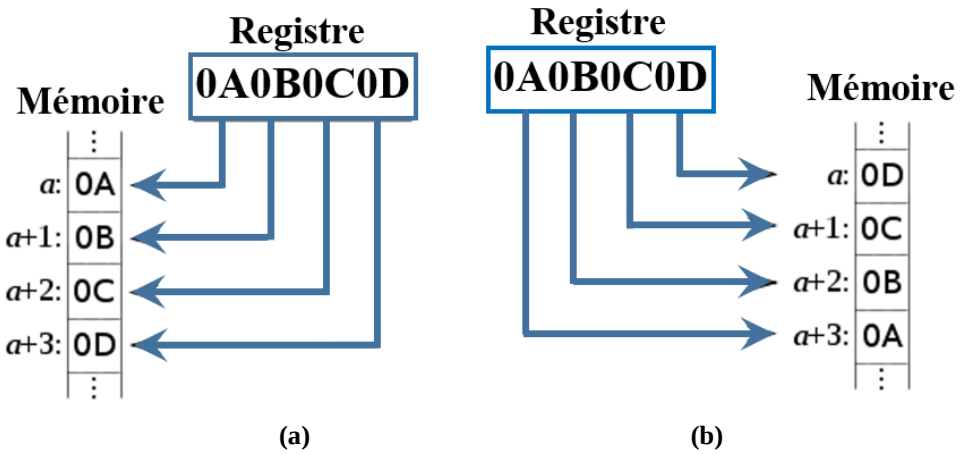


Figure 1.6 : Stockage de la valeur "0A0B0C0D" en mémoire, a) en *Big-endian*, b) en *Little-endian*

V.2.2.3. La couche de session

En utilisant la couche transport, la couche session sera responsable de l'établissement de connexions entre les ressources. Elle adapte les protocoles de l'IP et du NoC (rôle du wrapper).

V.2.2.4. La couche transport

Certains mécanismes qui vérifient qu'aucun paquet n'est perdu dans le niveau inférieur doivent être implémentés dans cette couche. En outre, elle découpe les messages en paquets et leur ajoute l'information indiquant leur destination (paquetisation/depaquetisation). Elle contrôle le flux des communications de bout en bout (rôle du *IRR : Interface Ressource-Réseau*, ou en anglais *RNI*: *Ressource-Network Interface*). Le contrôle de flux de bout en bout a pour but de synchroniser la transmission et d'éviter que des données ne soient envoyées vers une destination alors qu'elle n'a plus de place disponible pour les recevoir (les techniques de contrôle de flux sont présentées en détail dans la section V.3.5).

V.2.2.5. La couche réseau

Les commutateurs sont implémentés dans cette couche. Cette couche pend en charge également la topologie du réseau. Comme l'adressage est fortement lié à la topologie, il est également traité dans cette couche. Elle définit la technique de routage employée pour l'acheminement des paquets à travers le réseau et la topologie. Elle effectue donc le routage des paquets de la source vers la destination, l'adressage de ressources et la mise en mémoire tampon de paquets. Elle est également chargée de fournir la qualité de service en abordant les problèmes de latence, de débit, et de gigue, etc. Le *Protocol Data Unit* (PDU) ou l'Unité de données du protocole, qui est l'ensemble des informations échangées entre niveaux dans le système OSI, utilisé pour la communication entre les entités de cette couche est appelé paquet.

V.2.2.6. La couche liaison de données

Elle définit le protocole d'échange des bits sur les liens entre les routeurs (ex., poignée de mains, crédit d'émission, etc.). Elle est concernée donc par la flitisation/paquetisation, la communication de nœud en nœud, la détection et la correction d'erreur de transmission (bit de parité), le contrôle de flux, l'encodage des schémas, etc. Cette couche transmet idéalement des données d'un point à un autre, elle pourrait être omise dans une simulation. L'omission se fait sous l'hypothèse que toutes les données sont transmises sans erreurs. La détection et la correction de la couche physique sont implémentées dans la couche de liaison de données.

V.2.2.7. La couche physique

C'est la couche de plus bas niveau d'un NoC, elle traite le transfert réel de données. Elle est responsable des signaux de l'horloge pour chaque connexion, le nombre et la nature de l'interconnexion, les signaux de contrôle, les niveaux électriques, le moyen physique par lequel seront transportés les paquets, ainsi que la façon dont les données sont transportées (largeur des mots transportés en nombre de bits), etc. Comme cette couche traite des propriétés électriques, la simulation est mieux faite dans un simulateur analogique comme Spice [29]. Dans la plupart des six couches décrites ci-dessus, la plus petite unité de temps est un cycle d'horloge. Cependant, lors de la simulation de la couche physique, le temps devrait être exprimé en temps physique. Une co-simulation avec le temps physique et de cycles d'horloge serait difficile à mettre en œuvre et coûteuse en calcul.

V.2.3. Principaux composants du NoC

Il existe trois principaux types de composants à savoir les routeurs/commutateurs, les ressources également appelés les cœurs (appelés aussi en anglais *Intellectuel Property* (*IP*)) et les interfaces ressource-réseau abrégées IRR (ou en anglais *RNI*: *Resource Network Interfaces*). Un exemple d'une architecture NoC en topologie 4x4 2D Mesh avec ces composants est illustré par la figure 1.7. Dans la suite de la thèse on utilisera l'appellation "nœud", où un nœud est composé d'une ressource connectée à un routeur via une interface ressource-réseau IRR, comme le montre le zoom illustré sur la figure 1.7.

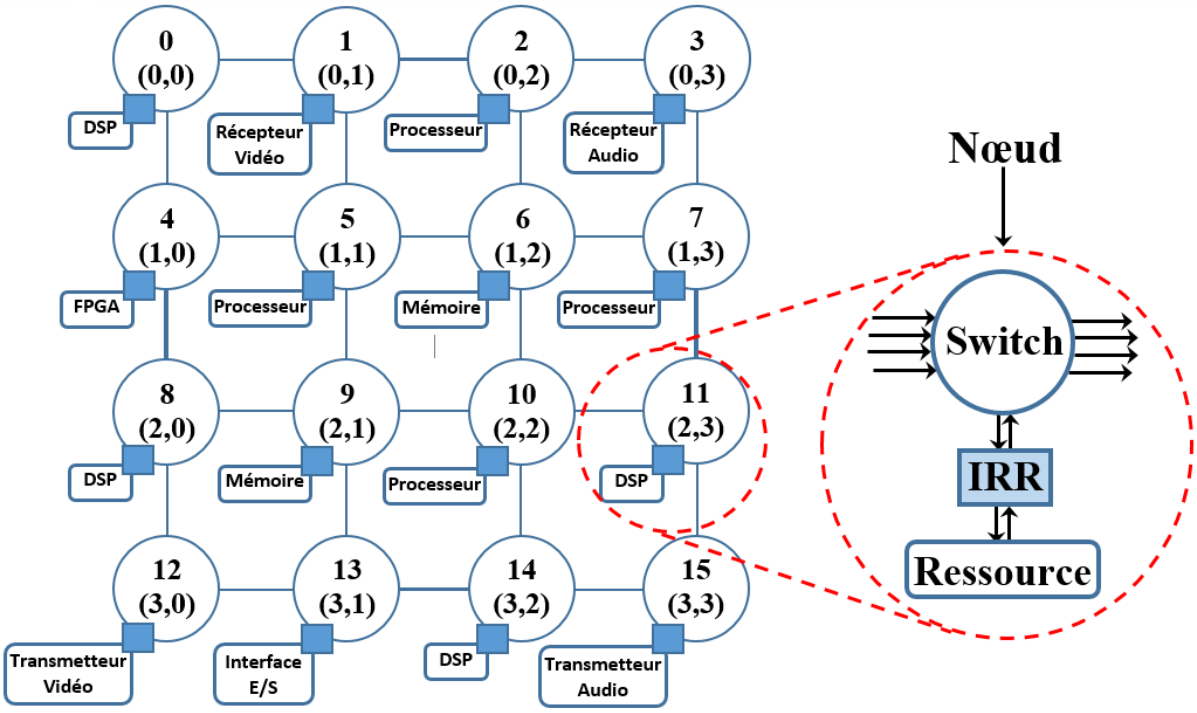


Figure 1.7 : La topologie NoC 4x4 Mesh avec les principaux composants identifiés

V.2.3.1. Les ressources

Une ressource peut être un processeur d'usage général, un DSP (de l'anglais "Digital Signal Processor", qu'on pourrait traduire par "processeur de signal numérique"), qui est un microprocesseur optimisé pour exécuter des applications de traitement numérique du signal (filtrage, extraction de signaux, etc.) le plus rapidement possible, une mémoire, un composant matériel d'application spécifique, contrôleur d'E/S, contrôleur graphique, le module de signal mixte, la fréquence radio unitaire (*RF*: *radio frequency*), etc. Les ressources devraient être mises en œuvre en utilisant la même technologie que celle du commutateur de réseau [20]. Le concepteur peut construire des ressources propres ou réutiliser les ressources commerciales disponibles chez différents fournisseurs de composants.

V.2.3.2. L'Interface Resource-Réseau (IRR) (ou en anglais RNI: Ressource Network Interface)

L'IRR relie une ressource à un routeur réseau. Par conséquent, elle permet à la ressource d'envoyer des données au routeur. Le rôle de l'IRR dans les NoCs est le même rôle que celui de la carte réseau pour l'Internet [20]. L'IRR se compose de deux parties, la partie dépendante des ressources et la partie indépendante des ressources comme le montre la figure 1.8. La partie indépendante des ressources est conçue de telle sorte que l'IRR apparaît comme un autre routeur connecté au routeur. La conception de la partie indépendante des ressources est commune à toutes les ressources. Si des ressources homogènes sont utilisées alors la partie dépendante des ressources peut être réutilisée et elle sera de même pour toutes les ressources, sinon elle est différente pour chaque ressource.

La partie dépendante des ressources est responsable de la flitisation/deflitisation et la mise en œuvre du schéma d'encodage. Dans le cas de routage source, l'IRR contient également des tables de routage et elle est le responsable de l'ajout du chemin complet dans le flit d'entête.

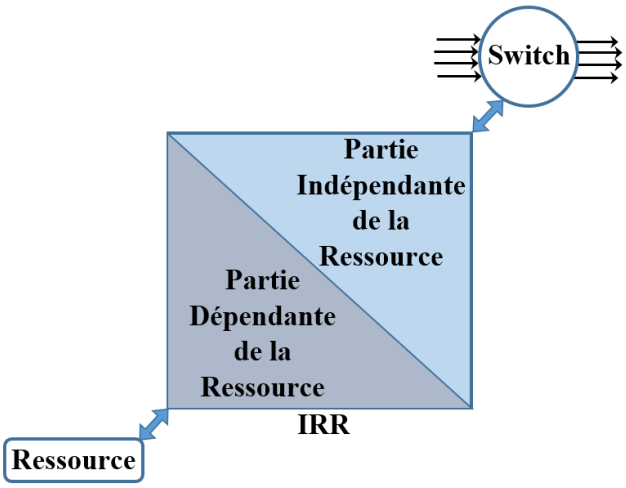


Figure 1.8 : L'Interface Resource-Réseau (IRR)

V.2.3.3. Le routeur/commutateur (Router/Switch)

Comme dans n'importe quel autre réseau, le routeur est l'élément le plus important pour la conception de la communication, épine dorsale d'un système sur puce. Dans un réseau de commutation de paquets, la fonctionnalité du routeur est de transmettre le paquet entrant de la ressource source vers la ressource destinataire si celle-ci est directement reliée à lui, ou de le transmettre à un autre routeur voisin. Il est primordial que la conception d'un routeur NoC soit aussi simple que possible, parce que le coût de la mise en œuvre augmente considérablement avec l'augmentation de la complexité de la conception du routeur.

La tâche principale d'un routeur conçu pour le routage distribué est de mettre en œuvre la fonction de routage. Pour un paquet entrant, la route peut être sélectionnée soit par consultation d'une table de routage déjà stockée dans la mémoire du routeur soit calculée dynamiquement par l'exécution d'un algorithme de routage.

La conception d'un routeur utilisé pour le routage distribué peut devenir complexe, car elle nécessite une logique supplémentaire et une mémoire pour mettre en œuvre la fonction de routage.

La conception du routeur pour le routage source sera très différente du routeur conçu pour gérer l'algorithme de routage distribué. Il n'a pas besoin de sélectionner le port de sortie pour le paquet entrant. L'information pré-décidée de la route est disponible dans le paquet lui-même. Mais il reste encore à mettre en œuvre les autres fonctions comme la mise en mémoire tampon du paquet et l'arbitrage, en utilisant par exemple la priorité pour résoudre les conflits de ports lorsque deux paquets ou plus nécessitent d'utiliser le même port de sortie. La simplicité d'un routeur NoC à routage source lui permet d'être plus rapide. Différents types d'architectures de routeur NoC pour le routage distribué ont été proposées par les concepteurs [22] [30] [31] [32].

Les routeurs [33] sont constituées de ports d'entrée/sortie contenant (ou pas) des files d'attente pour stocker les données qui ne pourront pas être transmises tout de suite, d'une matrice de commutation pour relier physiquement les ports d'entrée aux ports de sorties, des blocs de contrôle et d'arbitrage des paquets et enfin, d'une table de routage ou bien d'un bloc de calcul de la fonction de routage si ce-dernier est distribué [34].

La logique de contrôle peut servir à des tâches comme la détection/correction d'erreurs ou bien pour le contrôle de flux.

A. La temporisation des paquets

Afin de ne pas être obligé de détruire des paquets en conflits, ces derniers peuvent être temporisés dans des files d'attentes. Or, dans la majorité des NoCs, la surface consommée par

un routeur provient de la place prise par ces files. Dès lors, un compromis doit être trouvé lors de la conception du routeur pour limiter au maximum la surface utilisée par les files d'attente sans pour autant dégrader les performances requises.

Les files d'attentes sont principalement caractérisées par leur taille et leur emplacement au sein du routeur [34].

a. La taille des files d'attente

La taille comprend la largeur des files en nombre de bits et leur profondeur définissant le nombre de mots qui peuvent être stockés.

Le premier paramètre qui influe directement sur la taille des files d'attente est le mode de commutation choisi [35]. En effet, les files pourront contenir plusieurs paquets, un seul paquet ou encore une partie de paquet. Concernant la largeur de la file, elle ne pourra pas être plus petite que celle d'une unité de contrôle de flux (flit). De plus, la taille doit être rigoureusement déterminée car elle peut affecter la fréquence d'horloge maximale et le coût en termes de surface et de consommation d'énergie des routeurs. Enfin, une taille inappropriée peut impliquer des congestions au sein du réseau avec une réduction de la bande passante utile.

Les files d'attente peuvent être placées à différentes positions dans le routeur : en entrée, en sortie ou en sortie virtuelle [36].

b. La position des files d'attente

i. Files d'attente en entrée

Dans ce premier cas, chaque port d'entrée du routeur possède une file d'attente. Bien que cette technique soit la moins couteuse en surface, elle peut induire une saturation à cause du blocage de tête de ligne (*head-of-line*). Cela arrive lorsqu'une donnée en tête de file ne peut accéder au port de sortie associé, bloquant ainsi les autres paquets de la file, même si leur sortie est libre.

ii. Files d'attente en sortie

Lorsque les files d'attentes sont positionnées en sortie du routeur, chaque port de sortie a autant de files d'attente que de ports d'entrée. Cette technique offre de meilleures performances que la précédente mais la surface est forcément plus importante.

iii. Files d'attente en sortie virtuelle ou canaux virtuels

L'idée des files d'attente en sortie virtuelle est de combiner les avantages des deux techniques précédentes. Ainsi, chaque port d'entrée possède plusieurs files d'attente servant à répartir les paquets entrant en fonction de leur destination ou bien de leur priorité. On parle alors de canaux virtuels car pour un unique canal physique, il y aura autant de canaux virtuels que de files d'attente [34] [37].

Bien qu'ils impliquent une augmentation de la surface et de la latence, du fait de la mémorisation et du contrôle nécessaires, les canaux virtuels possèdent plusieurs avantages :

- ils évitent les interblocages (les paquets bloqués sont stockés dans différents canaux virtuels pour laisser passer les autres paquets);
- ils optimisent l'utilisation des liens;
- ils augmentent les performances du réseau (amélioration de la latence et de la bande passante);
- ils fournissent des services séparés (à travers des trafics séparés et des niveaux de priorités différents).

B. La matrice de commutation

Les ports d'entrée et les ports de sortie du routeur sont tous connectés les uns aux autres grâce à la matrice de commutation [33]. Elle peut être implantée par un crossbar ou alors en

cascadant plusieurs étages de multiplexeurs car la matrice a pour rôle de multiplexer les données en entrées du routeur vers ses ports de sortie [34].

C. L'arbitrage

Lorsqu'au sein d'un routeur, au moins deux paquets souhaitent sortir par le même port et ce, en même temps, un arbitrage doit être fait afin de choisir celui qui sortira le premier. Bien qu'il existe différentes techniques pour arbitrer les paquets en conflit, la caractéristique la plus commune est l'impartialité. En effet, l'arbitre doit être capable de fournir un accès équitable aux ports de sorties si les paquets ont la même priorité [34].

Les principales politiques d'arbitrage que l'on peut trouver dans les NoCs sont l'Accès Multiple à Répartition dans le Temps (*TDMA*: *Time-Division Multiple Access*), la réservation de Time-slot, le tourniquet ou Round-robin, et la priorité fixe [35].

- Par TDMA : c'est un mode de multiplexage consistant à diviser le temps en périodes courtes appelées plages de temps, ou time-slots, et à attribuer à chaque élément communiquant une ou plusieurs plages. Ainsi, durant toute la durée d'une plage, la ressource partagée sera entièrement réservée à l'élément auquel le time-slot est associé. Cependant, la plage de temps doit être minutieusement alignée au risque d'introduire des latences supplémentaires [35].
- Par réservation de time-slot : la politique par réservation de time-slot est en fait une politique de type TDMA dans laquelle la réservation de la plage de temps est répétée périodiquement.
- Par tourniquet ou Round-Robin : l'arbitrage de type tourniquet ou round-robin représente la politique d'arbitrage la plus impartiale. En effet, à chaque cycle, elle donne l'accès à un paquet provenant d'une entrée différente.
- Par priorité fixe : dans la technique par priorité fixe, chaque paquet possède une priorité fixe qui lui a été attribuée par l'émetteur avant son entrée dans le réseau. Cette priorité est utilisée pour l'acheminement du paquet de la source vers la destination.

V.2.3.4. Les liens

Les liens sont des connexions logiques entre deux (ou plusieurs) éléments communicants [33] [35]. Ils permettent les connexions de type point à point entre les ressources IPs et les interfaces ressource-réseau IRRs, entre l'IRR et les routeurs et entre les routeurs. Ce sont eux qui offrent la bande passante et qui transportent l'information entre les éléments communicants. Un lien peut consister en plusieurs canaux virtuels, il existe trois types de liens qui sont définis par le sens des transmissions. Ils peuvent être unidirectionnels en entrée, unidirectionnels en sortie ou bidirectionnels (entrée/sortie).

Dans les technologies de fabrication de composants électroniques (*CMOS: Complementary Metal Oxide Semiconductor*) actuelles, les canaux sont implantés en tant que signaux globaux du circuit. C'est pourquoi, bien qu'étant réduits, les problèmes de délais et de dispersion des signaux peuvent exister aussi dans les NoCs. Afin de résoudre ces problèmes, les signaux sont communément segmentés par des répéteurs, le plus souvent des tampons, qui permettent de restaurer le niveau de la tension des files. Dès lors que les files ont une longueur significative, les répéteurs peuvent incorporer des registres pour pipeliner la transmission des données. Ainsi, celles-ci arrivent à une cadence qui coïncide avec leur utilisation par les éléments de calculs connectés au réseau [34].

V.2.4. La topologie

La topologie d'un réseau définit comment les routeurs sont interconnectés entre eux en utilisant les liens du réseau. Elle spécifie l'organisation, souvent modélisée par un graphe, physique du réseau (nombre de routeurs, nombre de ports à chaque routeur, schémas de liens entre les routeurs, placement des ressources IPs, etc.).

V.2.4.1. Topologies existantes

Comme pour les réseaux informatiques, de nombreuses topologies sont envisageables pour construire un NoC. La figure 1.9 illustre les topologies les plus couramment utilisées dans la conception des NoC.

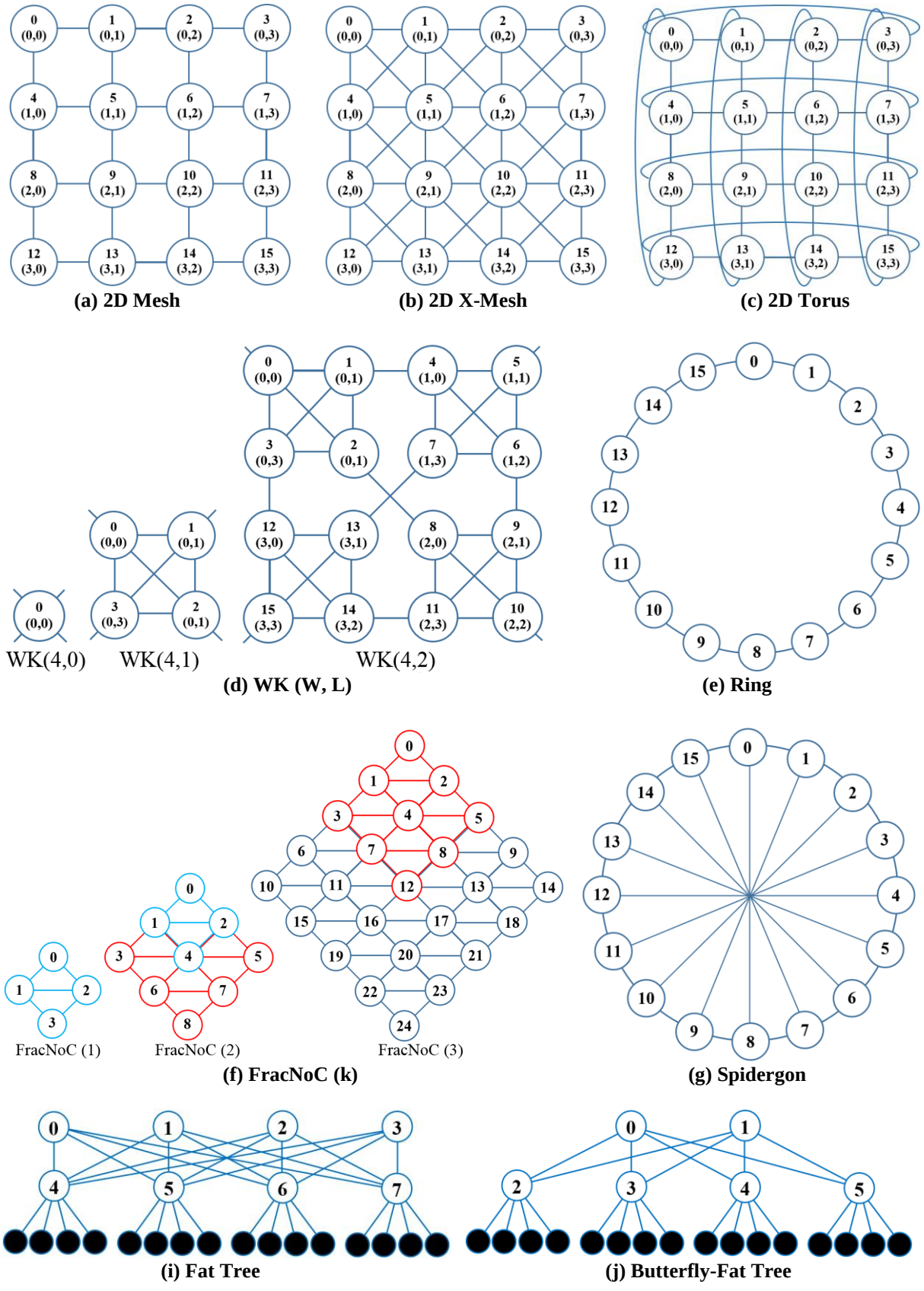


Figure 1.9 : Topologies NoC

Chaque topologie possède des avantages et des inconvénients. Afin de comparer les topologies entre elles, différents paramètres physiques sont utilisés [3] [4] [7] [38]. Parmi les paramètres les plus couramment considérés dans les publications on note le degré du routeur (nombre de ports), le nombre total de liens, le diamètre du réseau, la distance moyenne du réseau, le degré de clusterisation, la régularité, la symétrie, l'extensibilité, la fractalité, la diversité de chemins de routage, et la largeur de bissection.

V.2.4.2. Paramètres physiques de la topologie

A. Le diamètre

C'est le plus grand nombre de sauts dans tous les chemins les plus courts. Ce critère peut être utilisé comme indicateur pour l'évaluation de la latence maximale. Un petit diamètre permet une communication rapide entre les nœuds les plus éloignés. En d'autres termes, la latence maximale est proportionnelle au nombre maximal de sauts, c'est pour cette raison que le diamètre est considéré comme une mesure importante.

B. La distance moyenne

C'est le nombre moyen de sauts entre tous les nœuds dans le réseau. Ce critère peut être une indication pour évaluer la latence moyenne.

C. Le degré

C'est le nombre de voisins directs d'un nœud dans le NoC. Il indique si l'architecture NoC est régulière. Un degré élevé permet à de nombreux voisins proches d'effectuer des communications rapides. Toutefois, afin de réduire les coûts matériels et d'augmenter la capacité d'extension, un degré plus petit et fixe, est exigé.

D. La bissection

Représente le nombre minimal de liaisons à couper pour séparer le réseau en deux portions égales. Ce critère permet d'évaluer le coût de transfert des données de la moitié du réseau vers l'autre moitié. La bande passante collective de ces liens est nommée bande passante de bissection, et elle permet de mesurer la performance d'une topologie. Il s'agit d'un indicateur d'évaluation du débit. Par exemple, une grande bissection est nécessaire afin de fournir plus de chemins entre deux sous-réseaux, et améliore donc la performance globale du réseau.

E. Le nombre de liens

C'est le nombre de liens bidirectionnels du NoC. Ce critère peut être une indication de la charge du réseau. En outre, l'ajout de liens dans un NoC pourrait réduire le diamètre, d'améliorer la communication, et réduire la latence de communication. Cependant, le NoC avec un nombre de liens plus élevé (ex., un NoC entièrement connecté) est coûteux en termes de surface de silicium.

F. Le degré de clusterisation

Permet de spécifier la façon dont les nœuds sont interconnectés les uns aux autres. Le degré de clusterisation Cn d'un nœud i est donné par la formule :

$$Cn_i = {}^{2\,l_i}/_{n_i(n_i-1)} \quad (1)$$

Où n_i est le nombre de voisins du nœud i et l_i est le nombre de liens entre ses nœuds voisins.

Un degré de clusterisation plus élevé indique que les nœuds les plus proches les uns des autres sont fortement connectés. En d'autres termes, il indique la disponibilité des chemins de routage alternatifs afin de re-router des paquets congestionnés à un nœud donné, ce qui permet de réduire la congestion dans les nœuds intermédiaires. Un degré de clusterisation plus élevé favorise aussi la communication locale.

G. La régularité

Un réseau est dit régulier lorsque tous les sommets ont le même nombre de voisins, c'est-à-dire le même degré ou valence. Un réseau régulier dont les sommets sont de degré K est appelé un réseau k-régulier ou réseau régulier de degré k.

H. La symétrie

Un réseau est dit symétrique si quel que soit la paire de nœuds (i, j), il existe autant de liens l de la forme l_{ij} que l_{ji}.

I. La diversité de chemins de routage

Un réseau a une diversité de chemins de routage si la plupart des couples de routeurs ont de multiples chemins de routage entre eux. La diversité de chemins de routage ajoute de la robustesse au réseau.

V.2.4.3. Comparaison entre topologies

Le tableau 1.1 montre une comparaison entre les topologies les plus couramment utilisées par rapport à certains paramètres physiques.

Ces critères fournissent une première indication sur certaines performances/coûts métriques. Plus précisément, ils caractérisent chaque architecture NoC et les distinguent l'une de l'autre. Une petite distance moyenne permet d'avoir une latence de communication réduite, en particulier pour la technique de commutation à distance sensibles, comme la commutation de paquets en mode *store and forward*. Mais il est également crucial pour la technique de commutation à distance insensible, comme la commutation de paquets en mode trou de ver (*wormhole switching*), donc une petite distance moyenne a un impact sur les performance du réseau, à savoir l'utilisation de moins de liens et d'espace tampon, et donc moins de conflits de communication et de latence [39].

OCI / Propriété	Links	Diamètre	Dégrée	Bissection	Dégrée de Clusterisation	Distance Moyenne
2D Mesh	$2N - 2\sqrt{N}$	$2(\sqrt{N} - 1)$	2, 3, 4	$\sqrt{N}$	0	$2\sqrt{N}/3$
Torus	$2N$	$2(\sqrt{N} - 1)$	4	$2\sqrt{N}$	0	$\sqrt{N}/2$
WK(W, K)	$W * (W^L - 1)/2$	$2^L - 1$	3, 4	$(W/2)^2$	Min = 0,5 Max = 1	Non Défini
Spidergon	$(3/2)\ N$	$N/4$	3	$(N/2) + 2$	0	$\frac{N^2 + 4N - 8}{8N - 8}$
X-Mesh	$4N - 6\sqrt{N} + 2$	$(\sqrt{N} - 1)$	3, 5, 8	$3\sqrt{N} - 1$	Min = 0,42 Max = 1	Non Défini
FracNOC	$3N - 4\sqrt{N} + 1$	$2(\sqrt{N} - 1)$	2, 3, 4, 5, 6	$2\sqrt{N} - 1$	Min = 0,4 Max = 1	Non Défini

Tableau 1.1 : paramètres physiques des topologies NoC

Plus le diamètre d'un réseau est petit, moins le temps qu'il faut pour envoyer un message d'un nœud au nœud le plus éloigné dans le réseau est grand.

Par exemple, si une architecture NoC a un petit diamètre, un haut degré, et un degré de clusterisation élevé, ce réseau peut prendre en charge différents modèles de trafic de données. Tel que présenté dans le tableau 1.1, aucun NoC ne pourrait fournir le meilleur rendement pour une large gamme d'applications. Cependant, plusieurs études SoC ont choisi, par exemple, le 2D Mesh comme une topologie de base en raison de sa facilité d'implémentation, la simplicité des stratégies de routage, l'évolutivité du réseau, sa régularité et de la faible complexité matérielle qu'elle offre [40]. En plus de la topologie 2D Mesh, la topologie 2D Torus est aussi utilisée afin de réduire le diamètre du réseau et d'augmenter la bande passante. Toutefois, cette topologie possède certaines limitations. Sa structure est plus complexe à implémenter que pour

un réseau à maillage simple et les liens utilisés pour boucler le réseau sont plus longs (routeurs à la frontière). Ces limitations pénalisent donc les performances des liens. Pour réduire la longueur de ces liens, la topologie de maillage en tore replié à deux dimensions (2D folded Torus) est utilisée. Cependant, cette topologie rend les algorithmes de routage plus complexes.

Toutes ces topologies maillées (2D Mesh, 2D Torus et 2D Folded Torus) ont un problème avec la latence réseau associée et, par conséquent, avec la consommation d'énergie.

Dans la topologie NoC en anneau (Ring), chaque nœud est relié à deux nœuds voisins, et une ressource locale (IP). Il peut être représenté comme un tableau unidimensionnel de commutateurs, où le dernier commutateur est connecté au premier. L'anneau NoC a reçu l'attention de plusieurs projets industriels récents, tels que *Element Interconnect Bus of Cell Processor* (Sony, Toshiba, IBM) [41], qui utilise 4 anneaux parallèles qui relient 12 *IPs*. Cette architecture peut être très efficace pour les systèmes sur puce, mais lorsque tous les composants ont besoin de communiquer les uns avec les autres et que le trafic n'est pas local, elle offre une performance inférieure aux topologies maillées [14] [42]. Ceci est dû aux caractéristiques de cette architecture, c'est à dire à son faible degré, sa petite bissection, son degré nul de clusterisation et de son diamètre élevé. L'anneau NoC pourrait être utilisé pour connecter un petit nombre d'IPs, exécutant des applications non dominées par les communications de données en local. Cependant, la topologie anneau avec cordes (chordal Ring) [27] est utilisée pour obtenir plus de performance de communication que la topologie en anneau (Ring), mais elle rend l'interconnexion plus complexe et le routage plus difficile.

D'autres topologies intéressantes (ex., *fat-tree*, *butterfly fat-tree*, *Spidergon*, *WK*) ont été adaptées pour la conception de SoC [7] [38] [43] [44]. La topologie d'arbre élargi (*FT: fat-tree*) et la topologie d'arbre élargi en papillon (*BFT: butterfly fat-tree*) conduisent à un diamètre plus petit et la latence réseau associé est donc réduite. Les principaux inconvénients de ces topologies sont des problèmes de degré du routeur et la complexité d'interconnexion. La topologie BFT est utilisée pour obtenir un degré inférieur à la topologie FT. Cependant, cette topologie accroît la complexité d'interconnexion et le coût en surface, en raison de sa hauteur d'arbre plus élevé [27].

V.2.4.4. Synthèse

Le choix de la topologie pour une architecture NoC est complexe. Il est fortement lié aux nombreuses contraintes de l'application et au trafic du réseau. Nous devons faire un compromis entre la performance intrinsèque de la topologie et le coût que représente son implémentation.

Cependant, il n'y a pas d'architecture NoC universelle, qui pourrait soutenir tous les modèles de trafic des applications SoC. Afin de permettre aux concepteurs d'adapter l'architecture du réseau d'interconnexion sur puce candidat pour qu'il puisse soutenir un large éventail d'applications (ex., les différents modèles de trafic), une approche appropriée qui consiste à ajouter de longs liens stratégiques afin de correspondre à une grande charge de travail de l'application. Cette approche fait l'objet d'une de nos contributions dans cette thèse, elle sera détaillée dans le chapitre 2.

V.3. La communication dans les NoCs

V.3.1. Eléments de communication

Certains termes du réseau de communication couramment utilisés comme message, paquet, flit et phit sont abordées dans le présent paragraphe. La figure 1.10 montre la structure d'un paquet.

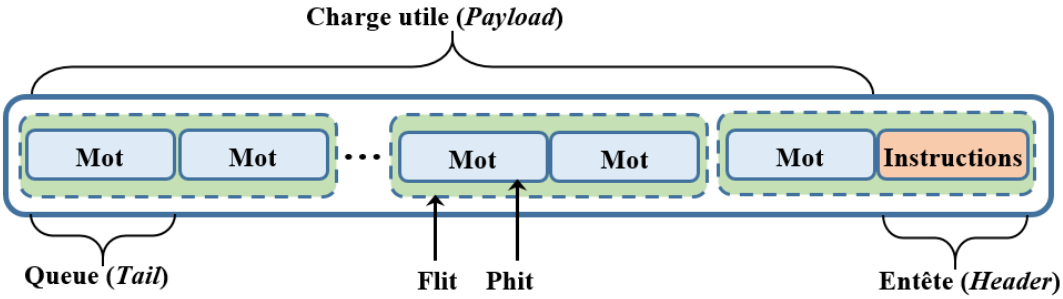


Figure 1.10 : La structure d'un paquet NoC

a. **Message :** il représente les données qui doivent être transférées à partir de la source vers la destination. Le message est défini dans la couche application. La taille du message peut être fixe ou variable. Un message peut être divisé en plusieurs paquets.

b. **Paquet :** un paquet représente une unité d'un ensemble de données. Il peut être transféré dans le réseau indépendant des autres paquets du même message. Il contient suffisamment d'informations, formatées pour une structure précise, qui traversent le réseau pour atteindre leur destination. Un paquet comporte principalement deux parties, la première partie sert à contenir des informations de contrôle et de routage et est appelée entête (*header*) et l'autre partie, appelée charge-utile (*payload*), contient des données. Parfois, les paquets contiennent également une queue (*tail*) indiquant la fin du paquet. Un paquet peut être de taille fixe ou variable et il peut être décomposé en unités de données plus petits appelés flits.

c. **Unités de contrôle de flux, ou *Flow Control Unites* (Flit) :** un flit est la plus petite quantité d'information (le plus petit morceau du message) pour lequel on peut définir un contrôle de flux. Il se compose d'un nombre constant de bits. Un flit peut être formés d'un ou plusieurs phits (sur l'exemple de la figure 1.10 un flit contient deux phits).

d. **Unités de transfert physique, ou *Physical Transfer Unites* (Phit) :** en fonction de la taille du flit et de la largeur des liens, plusieurs cycles peuvent être nécessaires pour transmettre un flit. Un phit représente donc la quantité d'information que l'on peut transmettre en un cycle (typiquement un flit a le même nombre de bits que de fils dans un lien).

Exemple **:** *Considérant qu'un paquet d'une taille fixe de* 512 *bits est utilisé dans le NoC. La taille du flit est fixée à* 32 *bits. Un paquet peut donc être décomposé en* 16 *flits. Si* 32 *fils sont disponibles entre tous les routeurs voisins du réseau, alors la taille du phit sera également de* 32 *bits. Mais si seulement* 16 *fils sont disponibles, la taille du phit sera alors de* 16 *bits et chaque flit sera ensuite transféré entre les routeurs en deux étapes de communication (chaque flit sera composé de deux phits).*

V.3.2. Génération de trafic sur puce

Les performances du réseau dépendent fortement du trafic réseau et, par conséquent, l'accès aux différents modèles de trafics est d'une importance capitale pour une analyse approfondie des différentes caractéristiques du réseau tels que les architectures, les protocoles et les implémentations. Il existe un travail de recherche préalable sur les modèles de trafic pour les différents réseaux allant de l'Internet [45] [46], l'Ethernet [47] [48], le Wireless LAN [49] jusqu'aux réseaux de multiprocesseur à mémoire partagée (*SMP*: *Shared Memory multiprocessor*) [50] [51] [52]. Ces modèles fournissent des renseignements cruciaux sur le comportement du trafic des réseaux correspondants.

Le réseau sur puce, cependant, expose sensiblement des comportements de trafic différents par rapport à ceux des réseaux traditionnels. Les différences sont évidentes par rapport à l'Internet, l'Ethernet et le Wireless LAN, où les applications, les protocoles et les implémentations diffèrent simplement en objectif, en échelle et en granularité [19] [33].

Même par rapport aux traditionnels réseaux d'interconnexion multi-châssis de multiprocesseurs symétriques à mémoire partagée (*SMP*: *symmetric shared memory multiprocessor*), de clusters et de supercalculateurs, les systèmes sur puce multi-cœurs ont des caractéristiques de trafic réseau différentes pour un certain nombre de raisons [52].

- Les systèmes intra-puce (on-chip) ont une hiérarchie de mémoire similaire à celle des systèmes inter-puce (inter-chip), les délais très courts de communication dans les systèmes intra-puce et la surface de silicium limitée, suggère une organisation de mémoire cache plus compacte avec une petite cache L1 et une cache L2 partagée [53] [54]. Il y a plus de transactions de communication entre les composants IPs sur puces menant le trafic NoC que sur les systèmes inter-puce.
- Les nouvelles architectures multiprocesseurs sur puce (*CMP*: *Chip Multi-Processors*) sont explorées activement dans l'industrie et les laboratoires universitaires [55] [56] [57], où chaque architecture CMP et son organisation affectent uniquement le volume et le profil du trafic NoC.
- Dans les applications spécifiques aux systèmes sur puce multiprocesseur (*MPSoC: multiprocessor System-on-Chip*), chaque MPSoC et son associé le réseau sur puce (*NoC : network-on-chip*) sont conçus pour un ensemble distinct d'applications cibles [58] [59] et, par conséquent, la conception de réseau et le trafic réseau associé sont très différents de ceux des réseaux d'interconnexion traditionnels.

Les profils de trafic pour la conception et l'analyse des NoC peuvent être classés en deux grandes approches. L'approche déterministe, qui consiste à chercher à reproduire exactement le trafic d'un composant, et l'approche synthétique (stochastique) qui consiste à modéliser le trafic à l'aide de processus stochastique et à synthétiser des réalisations de ce processus pour générer du trafic.

Chaque ressource NoC se comporte soit comme source (*Sender*), soit comme réceptrice (*Sink*), ce qui implique qu'elle est capable soit de générer ou de recevoir des paquets. Dans ce qui suit nous allons vous présenter quelques exemples d'applications dans chacune des deux catégories (sections V.3.2.1 et V.3.2.2). Ensuite, nous discuterons brièvement dans la section V.3.3 des modèles de trafic largement utilisés dans la littérature.

V.3.2.1. Génération de trafic déterministe sur puce

Les générateurs de trafic déterministe ont été utilisés pour analyser la consommation d'énergie et la latence des NoC. Les exemples incluent les modèles de trafic CODEC audio GSM [60], SPLASH-2 [61], Mediabench [62], et SPEC [63]. Cependant, il est à noter que les modèles de trafic générés par les différents modules dans un NoC dépendent fortement de l'application pour laquelle le NoC est conçu. Comme les performances du NoC sont déterminées en fonction du modèle de trafic, la façon la plus précise d'évaluer les caractéristiques du NoC serait d'invoquer les modèles de trafic correspondant à l'application.

Dans de nombreux cas, le système est conçu pour de multiples applications. Dans ces cas, les modèles de trafic correspondant à toutes les demandes doivent être utilisés lors de la conception et l'analyse du NoC. Cela peut prendre beaucoup de temps, même si toutes les applications sont connues à l'avance. Comme autre option on peut également choisir, les modèles synthétiques de trafic qui peuvent représenter une classe d'applications intéressante. Ceci suggère l'utilisation de modèles de trafic à la fois déterministe et synthétique, ce qui forme un ensemble complet pour l'évaluation des techniques proposées pour les systèmes NoC.

V.3.2.2. Génération de trafic synthétique sur puce

Les applications de ce type reposent sur la génération complète d'un trafic en se basant sur des heuristiques et des statistiques généralement admises pour modéliser l'évolution du

comportement d'une ressource NoC dans le temps. Cependant, le trafic généré est souvent trop simple et souffre d'un manque de réalisme. Quelques applications sont décrites ci-dessous.

A. Constant Bit Rate (CBR)

Elle peut être traduite en français par Bits à Taux Constant ou encore Débit en Bit Constant, c'est une application de génération de trafic qui peut être attachée à une ressource NoC, celle-ci génère des paquets à un débit binaire constant. L'application CBR est un processus d'alternance de deux fonctions, "Début CBR" qui provoque la ressource source pour commencer à générer des paquets, et "Arrêt CBR" qui provoque la source d'arrêter de produire des paquets. Les paramètres configurables pour l'application CBR sont les suivants:

- La taille du paquet (en octets)
- Le pourcentage de la charge des liens (le pourcentage de la bande passante du lien qui peut être utilisée)
- La destination : l'utilisateur peut spécifier une destination fixe ou "aléatoire", soit laisser l'algorithme de routage choisir aléatoirement une destination au moment du routage, cette destination est alors différente à chaque cycle d'horloge. Dans le cas du routage "Source", la destination est représentée par un code décimal appelé code de route (*route code*)
- L'intervalle inter-flit (en cycles d'horloge) : il est défini par l'intervalle en cycles d'horloge entre deux flits successifs

B. Bursty

C'est un trafic en rafales, il est représenté par une alternance de périodes *On* et *Off*. Au cours d'une période *On*, les paquets sont générés à intervalles fixes. Pendant la période *Off*, aucun paquet n'est généré. Le trafic est caractérisé par les variables distribuées exponentielles suivantes:

- La longueur du Burst (rafale) : nombre de paquets dans un Burst (sur la période)
- Le temps d'arrêt : intervalle entre deux rafales

Les paramètres configurables pour le trafic en rafales sont les suivants:

- La taille du paquet (en octets)
- Le pourcentage de la charge des liens (le pourcentage de la bande passante du lien qui peut être utilisée)
- La destination : l'utilisateur peut spécifier une destination fixe (le *route-code* dans le routage source) ou bien de mettre "aléatoire", L'intervalle inter-flit (en cycles d'horloge) : il est défini par l'intervalle en cycles d'horloge entre deux flits successifs
- La durée moyenne de rafale (en nombre de paquets)
- Le temps moyen d'arrêt (en nombre de cycles d'horloge)

C. Input trace based

Cette application génère le même trafic que celui de la simulation qui la précède. Cela permet de comparer la performance du NoC pour un ensemble différent de choix de paramètres pour le même trafic.

V.3.3. Modèles de Trafic (Traffic Patterns)

Le NoC se comporte différemment selon les algorithmes de routage utilisant différent types de modèles de trafic [64] [65]. Les chercheurs ont utilisé différents types de générateurs de trafic pour l'évaluation des NoC [64] [65] [66] [67]. Les modèles de trafic les plus largement utilisées pour l'analyse de la consommation d'énergie et la latence dans les réseaux d'interconnexion sont : Aléatoire (*Uniforme*), Transposé de type 1 et 2 (*Transpose*), Bit-Complément (*Bit-Complement*), Brouillé (*Shuffle*), Bit-Inverse (*Bit-Reversal*), Point-chaud (*Hotspot*), auto-similaire (*Self-Similar*) [68], et le point-chaud pour des applications spécifiques de trafic, etc.

V.3.3.1. Type de modèles de trafic

Pour décrire ces modèles de trafic synthétique, on suppose que chaque nœud (*x*, *y*) dans la conception NoC soit étiqueté avec une adresse résultant de la concaténation des indices du nœud *x* et *y* (ex., *xy*). La représentation binaire à *m* bits de *xy* est $n_1, n_2, \ldots n_{m-1}, n_m$. Avec le complément de zéro égal à un, et inversement, c.à.d. $\overline{0} = 1$ et $\overline{1} = 0$.

A. Uniforme

Chaque nœud envoie des messages à d'autres nœuds avec une probabilité égale. Les nœuds de destination sont choisis de façon aléatoire à l'aide d'une fonction uniforme de distribution de probabilité.

B. Hotspot

Chaque nœud envoie des messages à d'autres nœuds avec une probabilité égale, sauf pour un nœud spécifique (appelé Hotspot) qui reçoit des messages avec une plus grande probabilité. Le pourcentage de messages supplémentaires qu'un nœud Hotspot reçoit par rapport aux autres nœuds est indiqué après le nom Hotspot (par exemple, Hotspot 10%).

C. Transpose

Chaque nœud envoie des messages uniquement vers une destination avec la moitié haute/basse de sa propre adresse transposée. Soit la source avec l'adresse $(n_1, n_2, \ldots n_{m-1}, n_m)$ communique avec la destination dont l'adresse est donnée par $(n_{m/2}, n_{(\frac{m}{2})+1}, \ldots n_m, n_1, n_2, \ldots n_{(\frac{m}{2})-1})$.

D. Bit-Complement

Chaque nœud envoie des messages uniquement à celui qui est le complément de sa propre adresse, c'est à dire, la source avec l'adresse $(n_1, n_2, \ldots n_{m-1}, n_m)$ communique avec la destination dont l'adresse est donnée par $(\overline{n_1}, \overline{n_2}, \ldots \overline{n_{m-1}}, \overline{n_m})$.

E. Bit-Reversal

Chaque nœud envoie seulement au nœud dont l'adresse de l'expéditeur est inversé bit par bit, c'est à dire, l'adresse de destination est donnée par $(n_m, n_{m-1}, \ldots, n_2, n_1)$ pour la source $(n_1, n_2, \ldots n_{m-1}, n_m)$.

F. Shuffle

Chaque nœud envoie seulement au nœud dont l'adresse de l'expéditeur est décalé avec rotation à gauche par un bit, c'est à dire, pour l'adresse source $(n_1, n_2, n_3, \ldots, n_{m-2}, n_{m-1}, n_m)$, l'adresse de destination est donnée par $(n_m, n_1, n_2, \ldots, n_{m-3}, n_{m-2}, n_{m-1})$.

V.3.3.2. Synthèse

Le modèle de trafic *Uniforme* est une référence standard utilisée dans les études de routage réseau. Ce modèle peut être considéré comme un modèle de trafic pour des calculs de mémoire partagée bien équilibrés. Dans le modèle de trafic *HotSpot*, un ou plusieurs nœuds sont désignés comme étant des points chauds, qui reçoivent le trafic *HotSpot* en plus du trafic régulier. Par conséquent, le nœud *HotSpot* représente un nœud très occupé. Par exemple, dans les multiprocesseurs, les nœuds *HotSpot* pourrait être le représentant de trafic des calculs dans lequel il y a des sections critiques ou des données répliquées/partagées avec d'autres échanges de paquets. Pour le trafic *Transpose*, deux types de modèles sont proposés. Avec le premier modèle de trafic *Transpose*-1, un nœud (i, j) envoi uniquement des messages au nœud $(n - j, n - i)$ où n est la dimension du réseau (par exemple, $n \times n$ dans la topologie Mesh). Ce modèle de trafic est très semblable à la transposée de matrice. Dans le second modèle de trafic

Transpose-2, un nœud (i, j) envoi seulement des messages au nœud (j, i). Les modèles de trafics *Bit-Complement*, *Bit-Reversal*, et *Transpose* peuvent modéliser des traces d'applications liées aux calculs numériques [69].

De nouveaux modèles de trafic synthétiques peuvent être inspirés par l'analyse des modèles de trafic dans une classe d'applications. Par exemple, dans [68], un concept d'autosimilarité est utilisé pour proposer un nouveau modèle de trafic.

La discussion ci-dessus montre que chaque modèle de trafic synthétique est utile pour certaines catégories d'applications. Dans cette thèse, nous ne considérons que cinq modèles de trafic pour l'évaluation de nos approches pour le NoC, soit donc les modèles : *Uniform*, *HotSpot*, *Bit-Reversal*, *Shuffle* et *Transpose*.

V.3.4. Techniques de commutation

Les techniques de commutation sont utilisées pour acheminer des données depuis la source vers la destination dans un réseau de communication [70]. Généralement, la transmission de données entre deux ressources se fait en passant par des routeurs intermédiaires. La commutation consiste à mettre en relation temporairement deux ressources IPs, ce qui consiste à créer une liaison logique. L'équipement de commutation est un commutateur/routeur. La latence du réseau dépend principalement de la technique de commutation utilisée [36] [71]. Il existe plusieurs techniques de commutation, telles que :

V.3.4.1. La commutation de circuits (circuit switching)

La communication par commutation de circuits est basée sur la construction d'un chemin unique exclusif entre la source et la destination, lors de l'établissement d'une séquence de dialogue entre ces deux ressources communicantes. Le chemin, appelé un circuit, ainsi créé reste jusqu'à la clôture de la séquence de dialogue, comme le montre la figure 1.11. Actuellement, les concepteurs s'orientent vers l'utilisation de la commutation de circuits virtuels. La commutation de circuits se décompose en trois phases :

a) la connexion
   - Un chemin est construit entre la source et la destination
   - Liaison virtuelle directe entre la source et la destination

b) le transfert
   - Transmission de bout en bout des données sur le circuit
   - Ressources (liens, routeurs, etc.) attribuées pour toute la transmission

c) la libération
   - Restitution des ressources utilisées

Avantages :
- Gain de temps s'il y a plusieurs envois
- Pas de partage de la bande passante

Inconvénients :
- Occupation durable (blocage) des ressources
- Toute erreur de transmission nécessite de refaire intégralement la transmission

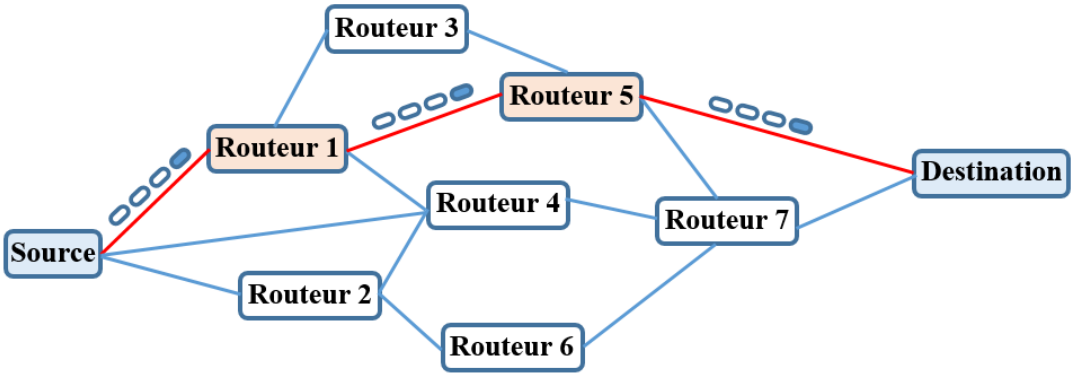


Figure 1.11 : La commutation de circuit

V.3.4.2. La commutation de messages (messages switching)

Dans la commutation de messages, aucun chemin physique n'est établi entre les deux ressources IPs. Chaque message est envoyé individuellement par le réseau d'une source vers une destination en passant par un ou plusieurs routeurs de commutation, comme illustré sur la figure 1.12. Chacun de ces routeurs attend la réception complète du message avant de le remettre sur le réseau. Cela demande de grands tampons sur chaque équipement, d'au moins la taille du message, ainsi qu'un contrôle de flux pour éviter les engorgements.

L'acheminement du message vers la destination se fait donc au fur et à mesure des commutations :

- le message arrivant au commutateur (routeur) $n-1$ est stocké, puis vérifié
- la liaison entre les commutateurs $n-2$ et $n-1$ est libérée
- le message est transmis au commutateur n après établissement d'une liaison entre les deux commutateurs

Ce processus est réitéré jusqu'à la livraison du message à la ressource destinataire.

Avantages (par rapport à la commutation de circuits):

- Une meilleure utilisation des lignes
- Le transfert est assuré même si le correspondant distant est occupé ou non connecté
- La diffusion d'un même message à plusieurs correspondants
- Le changement de format de messages
- L'adaptation des débits

Inconvénients :

- Risque de débordement des tampons de stockage, tout dépend de la taille des messages

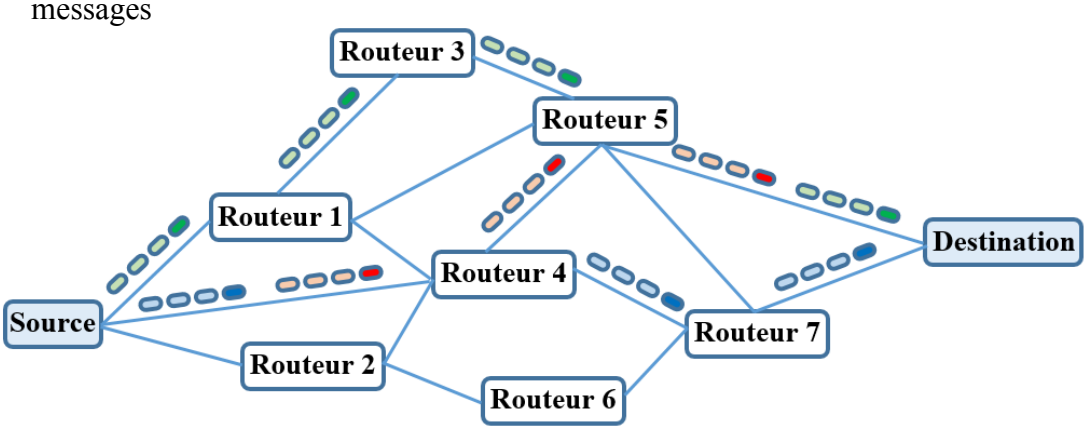


Figure 1.12 : La commutation de message

V.3.4.3. La commutation de paquets (packets switching)

La commutation de paquets, quant à elle, consiste à découper les données à communiquer en plusieurs paquets et à envoyer chacun d'entre eux sur un lien de façon différente afin d'en accélérer le transfert. La taille des paquets est variable mais limité à un maximum. Les délais de communications sont non-déterministes, du fait des contentions des ressources de communications. Ce type de commutation nécessite un contrôle de flux et de congestion. Il existe trois principaux modes de commutation [36], la commutation en mode différé (*store and forward*), la commutation de circuit virtuel (*virtual cut-through*) et la commutation de trou de ver (*Wormhole*).

A. La commutation en mode différé (store and forward)

Le commutateur met en tampon, et le plus souvent, réalise un contrôle de flux sur chaque paquet avant de l'envoyer. Cela implique que chaque routeur doit être capable de stocker la totalité d'un paquet de donnée, cf. figure 1.13. Lorsque la taille des paquets est grande, il leur faut un grand espace de stockage. Comme les ressources de stockage sont très coûteuses en termes de surface et de consommation, la taille des paquets devient donc limitée. De plus, le délai de transmission des paquets augmente à chaque routeur vu que tous les flits du paquet doivent être reçus par le routeur actuel avant qu'ils soient envoyés au routeur suivant. La latence totale est donc le produit du temps de transfert d'un paquet entre deux routeurs par le nombre de routeurs traversés par le paquet de sa source à sa destination. Cette technique pallie l'inconvénient de la taille des messages en fragmentant les données en paquets (la taille des paquets est fixée dans les protocoles). Donc, les commutateurs seront conçus avec une faible capacité de stockage.

Avantages :

- A débit constant le délai d'acheminement est plus court
- En cas d'erreur, seul le paquet erroné doit être retransmis

Inconvénients :

- Différences de vitesses des commutateurs des lignes
- Réassemblage des données à l'arrivée (paquets numérotés)

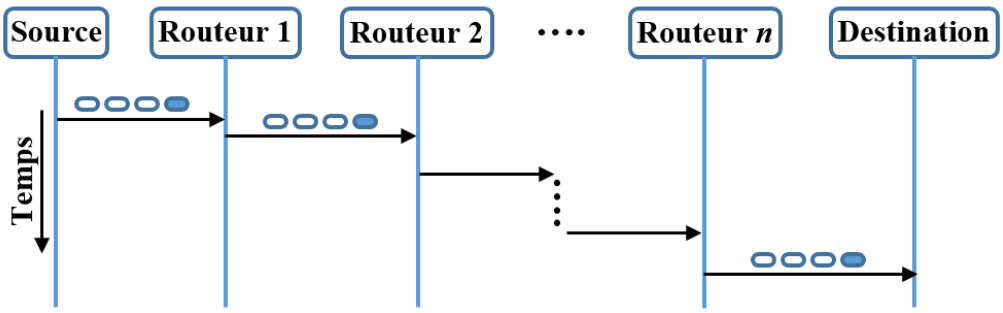


Figure 1.13 : La commutation en mode différé (*store and forward*)

B. La commutation en mode circuit virtuel (Virtual-Cut-Through)

Le circuit virtuel est établi et libéré à l'initiative d'une des ressources IPs communicantes. Ce mode de commutation a été proposé afin de réduire la latence des paquets à chaque étape de routage. Dans ce mode de commutation, un paquet peut commencer à être transféré au routeur suivant avant la réception complète de ce paquet par le routeur actuel, comme illustré dans la figure 1.14. Un circuit est créé entre le routeur "1" connecté à la source et le routeur "*i*" pour la commutation de circuit, ce circuit est un circuit virtuel (en principe optimal). Un paquet d'appel est construit par routage adaptatif, il sera transmis en premier vers la destination pour la réservation des ressources au fur et à mesure du routage, tous les paquets suivants, suivent le même chemin selon l'ordre de leur émission. Le paquet d'appel fixe le routage des paquets de données. Après la fin de transmission, le circuit est fermé par un paquet de libération qui parcourt le circuit, libérant les ressources.

Avantages :

- la latence est diminuée

Inconvénients :

- Le routeur actuel doit pouvoir stocker le paquet dans sa totalité si le routeur suivant n'est pas disponible

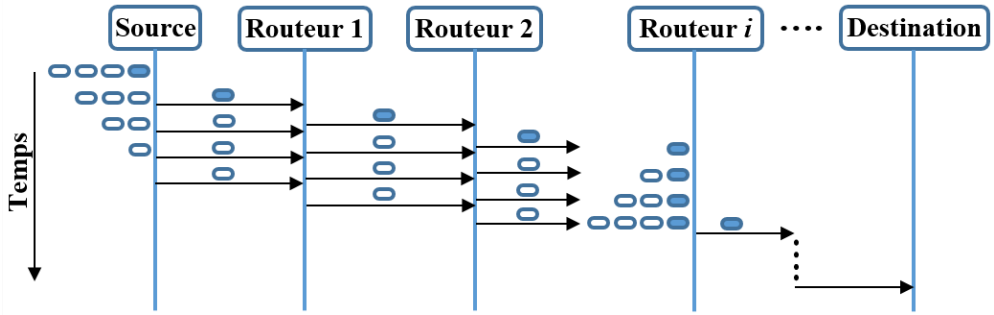


Figure 1.14 : La commutation en mode circuit virtuel (*Virtual Cut-Through*)

C. La commutation en mode trou de ver (wormhole)

Dans ce mode de commutation, un paquet est divisé en flits, Il y a trois types de flits à savoir le flit d'entête (header), qui peut être suivi par un ou plusieurs flit(s) de données (payload), et le flit de fin (tail). Le flit d'entête porte les informations de contrôle et de routage du paquet, les flits du corps portent la charge utile (payload) et le flit final contient la charge-utile ainsi que des informations de fin de paquet. Les flits passent de routeur en routeur dès qu'il y a de la place pour un flit et pas nécessairement pour un paquet complet (cf. figure 1.15). Le flit d'entête réserve le chemin de routage, tous les flits qui suivent doivent emprunter le même chemin que le flit d'entête.

Avantages :

- Un même paquet peut être réparti sur plusieurs routeurs du réseau ;
- Le routeur ne doit pas stocker un paquet complet ;
- La taille de stockage des tampons est donc diminuée ;
- La latence est réduite.

Inconvénients :

- Les risques de blocage sont plus importants car un paquet s'étend sur plusieurs routeurs.

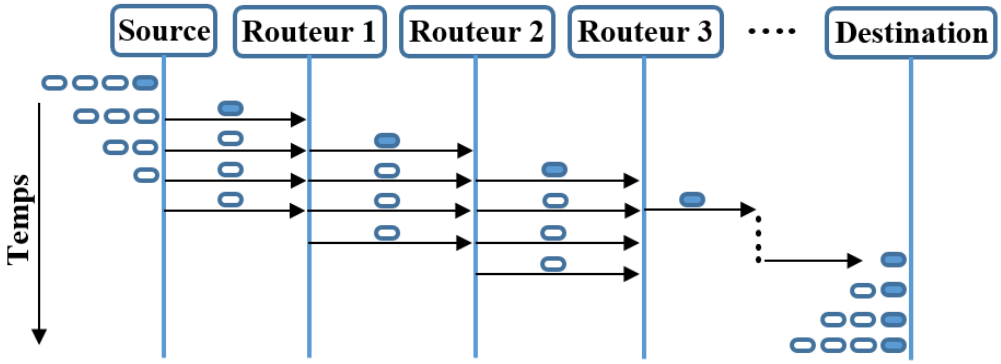


Figure 1.15 : La commutation en mode trou de ver (*wormhole*)

Dans notre travail, nous utilisons la technique de commutation *wormhole*. Les flits dans la commutation *wormhole* traversent le réseau dans un pipeline. Ainsi, les tampons dans les routeurs nécessitent une très petite capacité de stockage, aussi petite que la taille d'un flit. Les flits semblent se déplacer à travers le réseau comme un ver d'où le nom de commutation de trou de ver (*wormhole*) pour cette technique.

Exemple : *Dans cet exemple, on considère qu'un nœud est composé d'une ressource connectée à un routeur via une interface ressource-réseau (IRR), comme le montre le zoom illustré sur la figure* 1.7. *L'exemple représente une communication spécifique entre une ressource source connecté au routeur* (0,0) (*nœud (*0,0)) *avec une ressource destinataire connectée au routeur* (3,3) (*nœud* (3,3)). *Le nœud (0,0) envoie un paquet de cinq flits* (*un flit d'entête* "***H***" (*Header*), *trois flits de charge utile* "***P***" (*Payload*), *et un flit de queue* "***T***" (*Tail*)) *au nœud* (3,3). *Le routeur du nœud (*0,0*) décide d'envoyer le flit d'entête* "*H*" *au routeur du nœud* (0,1) *selon les informations de routage et l'algorithme de routage utilisés. La route des flits restantes est également verrouillée par ce routeur et les flits suivants ne seront pas vérifiés pour décider à nouveau l'itinéraire à suivre. De même, le flit d'entête continue à verrouiller d'autres routeurs* ((0,2), (0,3), (1,3), (2,3), *jusqu'à atteindre le routeur* (3,3) *relié à la destination) et le reste des flits le suivent en permanence sous forme de pipeline. À un certain temps t, ces flits sont sur le chemin de leur destination comme le montre la figure* 1.16.

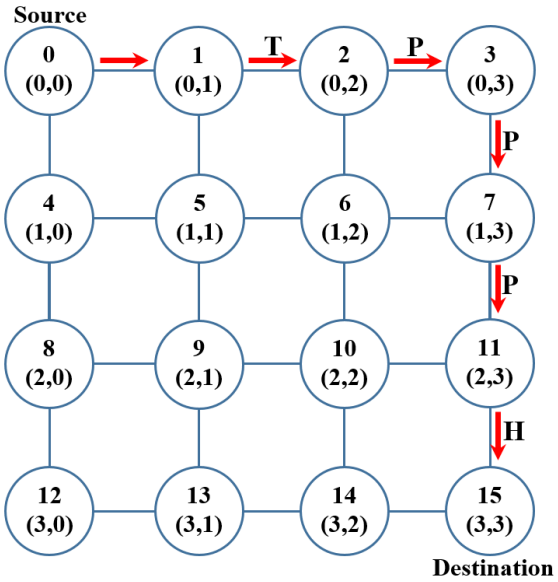


Figure 1.16 : Démonstration de commutation wormhole sur la topologie NoC 4x4 mesh

V.3.5. Le contrôle de flux

Le contrôle de flux est utilisé pour synchroniser la transmission de données à l'intérieur d'un routeur, entre les routeurs voisins et entre les IPs communicantes. Il assure qu'une IP source n'engloutie pas l'IP de destination (ou le port d'entrée n'engloutit pas le port de sortie dans un routeur) avec des données. Ainsi, le but du contrôle de flux est de garantir un transport de données sûr, sans perte de paquets du fait d'un dépassement de capacité de l'un des tampons parcourus, depuis l'IP source jusqu'à l'IP de destination. Il existe plusieurs protocoles de contrôle de flux [72], nous détaillerons dans ce qui suit les trois principales techniques.

Les techniques de contrôle de flux peuvent être classées en fonction de leur granularité de l'allocation de la bande passante du canal et de l'allocation de la mémoire tampon [23]. L'unité de base de l'allocation de la bande passante et de mémoire de stockage est le flit (*flow control digit*). Les paquets sont divisés en une séquence de flits. Différemment des paquets, les flits ne portent pas d'informations de routage et de séquencement. Le contrôle de flux flit-tampon alloue la bande passante et la mémoire tampon en unités de flits. Ceci a trois avantages: il (*i*) réduit la mémoire de stockage nécessaire pour le bon fonctionnement du routeur, (*ii*) fournit une contre-pression plus raide du point de vue congestion à la source du flux, et (*iii*) permet une utilisation plus efficace du stockage. Étant donné que ces avantages correspondent bien aux caractéristiques de la communication sur puce, les techniques de contrôle de flux flit-tampon sont considérées comme une solution prometteuse pour le NoC, où généralement la taille du flit correspond au parallélisme du canal. Les deux principales techniques de contrôle de flux de

haut niveau sont le contrôle de flux en mode trou de ver (*wormhole flow control*) et le contrôle de flux en mode canal virtuel (*virtual-channel flow control*). Ces techniques doivent être supportées par l'un des trois principaux mécanismes de contrôle de flux de bas niveau qui assure la gestion de la mémoire tampon et du contrôle, à savoir: l'*On/Off*, le *Credit-Based* et l'*Ack/Nack* [23]. En plus de l'allocation de la bande passante et des ressources de stockage NoC, les techniques de contrôle de flux doivent fournir de bonnes performances, en garantissant une grande largeur de bande passante pour la transmission d'un flux de flits, malgré la présence possible d'intervalles de décrochage de cycles (*stalling-cycles*) causés par la congestion dans les nœuds récepteurs en-avals. Le choix de la stratégie de contrôle de flux a des conséquences sur la conception des composants de réseau et, en particulier, sur la taille des files d'attente flit-tampons dans les routeurs. Si nous voulons éviter la perte de flits, le bon fonctionnement d'un contrôle de flux particulier définit une contrainte sur la taille minimale Q_{min} de la file d'attente Q. Une fois cette contrainte est respectée, l'augmentation de la valeur de Q conduit généralement à une meilleure performance.

Sous conditions que le trafic soit équilibré (injection/éjection), et que les capacités des tampons ne soient jamais dépassées, tous les protocoles sont fonctionnellement équivalents ; c'est-à-dire qu'ils permettent d'avoir exactement les mêmes débits de données. Cependant, sous un régime de trafic plus soutenu, des disfonctionnements apparaissent au niveau des performances ; en effet, sous de telles conditions de trafic les mécanismes de synchronisation sont activés plus souvent et induisent naturellement plus de pénalités sur le débit de données utiles. Les figures 1.17, 1.18 et 1.19 illustrent les modes de fonctionnement de chacun des protocoles *On/Off*, *Credit-Based* et *Ack/Nack* respectivement [73].

Le taux d'activité des protocoles de synchronisation joue aussi un rôle très important lorsque les délais des lignes de transport deviennent grands (ex., le cas d'une ligne série ou de liaisons off-chip). Dans ces cas-ci, le taux d'activité et d'autant plus grand que les délais sont élevés avec tout ce que cela implique en terme de diminution de débit et d'accroissement de la consommation dynamique.

V.3.5.1. Contrôle de flux On/Off

On/Off est un mécanisme de contrôle de flux simple qui minimise la quantité de signalisation de contrôle en échange d'une plus grande taille de la file d'attente Q. Le nœud émetteur en-amont (*upstream*) a un registre d'état d'un seul bit qui bascule entre l'état "*on*" et "*off*" sur la base du dernier signal de contrôle reçu du nœud récepteur en-aval (*downstream*). Ce dernier envoie un signal "*off*" de retour à chaque fois que le nombre d'emplacement libres dans sa file d'attente flit-tampon descend au-dessous d'un seuil F_{off} et envoie un signal "*on*" à chaque fois qu'il passe au-dessus d'un seuil F_{on}. Par conséquent, la taille minimale Q_{min} dépend du nombre de flits qui peuvent être reçus pendant le temps T_{rt} à partir de l'instant où un signal "*off*" quitte le nœud récepteur en-aval jusqu'à l'instant où le nœud récepteur en-aval reçoit le dernier flit transmis par le nœud émetteur en-amont avant de caler en raison du traitement et de la réception d'un signal "*off*" (cf. figure 1.17). Les valeurs des seuils F_{off}, F_{on} dépendent principalement des caractéristiques temporelles (délais de transport) du lien reliant les deux nœuds. Dans un NoC synchrone, si la latence du canal est K cycles d'horloge, alors :

$$T_{rt} = (2 + 2 * K) * T_{clk} \quad (2)$$

Ainsi, pour la justesse du protocole :

$$Q_{min} = 2 + 4 * K \quad (3)$$

Cependant, pour optimiser la bande passante, nous devons considérer également le double cas où le nœud récepteur en-aval envoie un signal "*on*" au nœud émetteur en-amont pour reprendre la transmission. Dans ce cas, au moins T_{rt} cycles doivent passer avant qu'un nouveau flit arrive au nœud récepteur en-aval. Pendant ce temps, afin d'avoir suffisamment de flits à

transmettre au prochain saut, la file d'attente du nœud récepteur en-aval doit être capable de contenir autant de flits supplémentaires pour une taille totale Q_{min}.

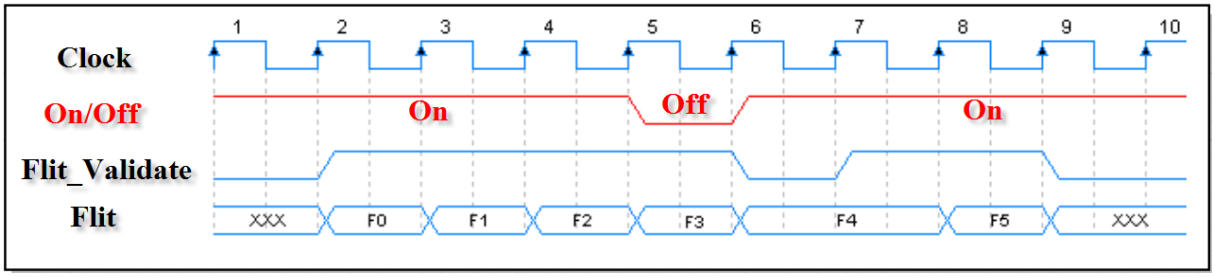


Figure 1.17 : Mécanisme de contrôle de flux On/Off

V.3.5.2. Contrôle de flux Credit-Based

Credit-Based est un mécanisme de contrôle de flux, où le nœud émetteur a un compteur pour suivre le nombre de d'emplacement libres disponibles dans la file d'attente du nœud récepteur (le crédit), ce nombre d'emplacement est équivalent au nombre de flits pouvant être émis en séquence. Le compteur est décrémenté à chaque fois qu'un flit est transmis et incrémenté à chaque fois qu'un signal de crédit arrive du nœud récepteur, qui à son tour envoie le crédit à chaque fois qu'il a réussi à transmettre un flit de sa file d'attente pour le prochain saut. Par rapport au contrôle de flux *On/Off*, le contrôle de flux *Credit-Based* nécessite plus de signalisation de contrôle, mais de plus petites files d'attente. En particulier, pour la justesse du protocole, une Q_{min} = 1 est suffisante. Toutefois, cette petite taille conduit à l'insertion de flits vides (des bulles) à chaque saut, vu qu'un seul crédit est disponible sur chaque canal à un moment donné. Par conséquent, un seul flit peut être transmis à chaque aller-retour de ce crédit, et plus la valeur de la latence du canal $K > 1$ est élevée, plus la performance est mauvaise. Pour éviter l'insertion de bulles, la file d'attente doit être dimensionnée en fonction de la latence aller-retour.

$$T_{crt} = (2 + 2 * K) * T_{clk} \quad (4)$$

Ce qui conduit à :

$$Q_{min} = 2 * (1 + K) \quad (5)$$

La figure 1.18 montre un exemple du mode de fonctionnement de ce mécanisme de contrôle de flux.

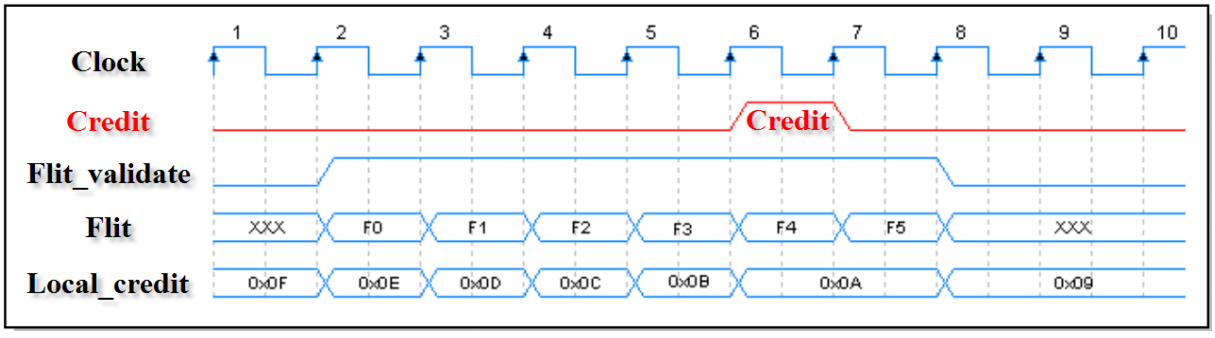


Figure 1.18 : Mécanisme de contrôle de flux *Credit-Based*

V.3.5.3. Contrôle de flux Ack/Nack

Le contrôle de flux *Ack/Nac*k ne nécessite aucun compteur dans le nœud émetteur en-amont (*upstream*) pour indiquer la disponibilité du tampon dans le nœud récepteur en-aval (*downstream*). C'est un protocole très simple où la logique de contrôle se charge d'acquitter chaque flit reçu du côté du nœud récepteur en-aval. Le nœud émetteur en-amont quant à lui doit donc conserver chaque flit émis jusqu'à ce qu'il ait été acquitté. Les flits sont envoyés avec

optimisme dès qu'elles sont disponibles, si le nœud récepteur en-aval dispose d'un emplacement libre dans la file d'attente, il accepte le flit en renvoyant un signal *Ack* au nœud émetteur en-amont, sinon il ne l'accepte pas et il renvoie un signal *Nack* au nœud émetteur en-amont. Le mécanisme de contrôle de flux *Ack*/*Nack* est traditionnellement considéré comme inefficace en termes de stockage (chaque flit transmis doit attendre un accusé de réception) et la bande passante (en raison des retransmissions potentiels) [23]. En outre, puisqu'il est basé sur l'acquittement de la réception de chaque flit spécifique, il fonctionne bien pour un canal avec une unité de latence. Mais, si le canal contient un nombre K de *FF*-répéteurs (ex., *FF*-800-Répéteur-Contrôleur qui a la possibilité de tout contrôler à partir des plus simples aux plus complexes systèmes de répéteur), il devient non-optimal (plusieurs unités de latence) par rapport au mécanisme *Credit-Based* où un accusé de réception désigne une transmission réussie d'un flit générique. Pourtant, le mécanisme *Ack*/*Nack* est efficacement utilisé pour implémenter les protocoles de tolérance aux pannes "Aller-Retour-N" (Go-Back-N) [72], avec les routeurs ayant des files d'attente de sortie de taille $Q_{min} = 1 + (2 * K)$.

Ce protocole est typiquement implémenté par le mécanisme du bit alterné "*Alternanting Bit Protocol*". Un exemple du mode de fonctionnement de ce mécanisme de contrôle de flux est illustré par la figure 1.19.

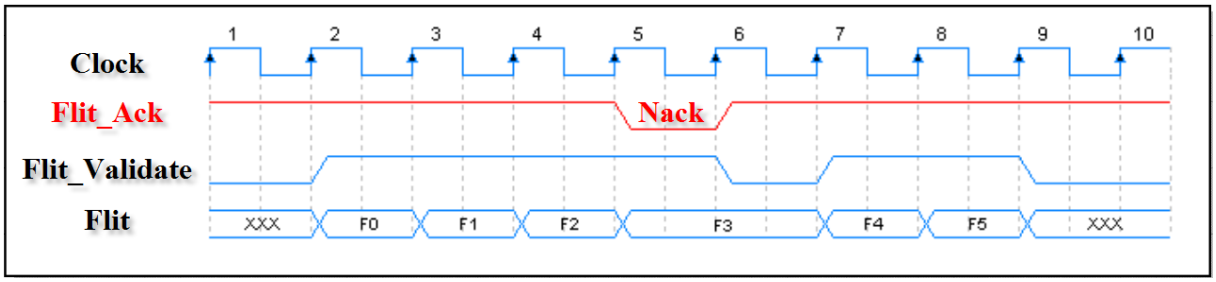


Figure 1.19 : Mécanisme de contrôle de flux *Ack/Nack*

V.3.6. Le routage dans les NoCs

V.3.6.1. Classification des techniques de routage

Comme d'autres réseaux, les performances de communication d'un NoC dépendent fortement du mode de routage utilisé. Les techniques de routage ont été classées dans la littérature de plusieurs façons. Dans cette section (V.3.6.1), nous discutons en détail ces différentes classes des modes de routage, nous illustrons leur utilisation dans le NoC et nous présentons leurs avantages et leurs inconvénients.

Nous nous sommes particulièrement intéressés dans nos travaux de thèse au routage source (déterministe). Par conséquent, nous allons vous montrer, dans la section suivante (V.3.6.2), les avantages de ce mode de routage, comparé à celui du routage distribué (adaptatif).

D. Routage source vs distribué (déterministe vs adaptatif)

Le routage dans le NoC peut être source ou distribué (déterministe ou adaptatif). Dans le routage source, l'itinéraire de l'ensemble du trajet depuis la source vers la destination est pré-calculé et fourni dans l'entête du paquet contrairement au routage distribué, où l'entête du paquet ne contient que l'adresse de destination et que la trajectoire est calculée dynamiquement par la participation des routeurs intermédiaires du chemin [20] [23] [74]. Dans le routage distribué, de multiples chemins de la source vers la destination sont possibles. Quand un paquet entre dans un routeur, l'adresse de destination est lue à partir de l'entête et en conséquence, la fonction de routage calcule tous les ports de sortie possibles où ce paquet peut être transmis. Ensuite, une fonction de calcul d'itinéraire sélectionne un des ports de sortie admissible pour transmettre le paquet. La sélection du port de sortie dépend des conditions dynamiques du réseau tel que la congestion et les défauts dans les liens. Il existe aussi des algorithmes de routage partiellement distribué qui restreignent certaines voies de

communication. Ils sont simples et faciles à mettre en œuvre par rapport aux algorithmes de routage distribué.

Avec le routage source, toutes les décisions de routage sont prises à l'intérieur du nœud source avant l'injection de tout paquet dans le réseau. A cet effet, chaque source contient des listes ou des tableaux qui contiennent des informations sur l'itinéraire complet pour atteindre toutes les autres ressources dans le réseau. Au lieu de stocker des tables dans la source, il est également possible d'ajouter une logique ou un logiciel supplémentaire dans les ressources sources qui implémente n'importe quel algorithme de routage adaptatif et dynamique qui calcule les chemins pour le routage source.

Afin d'acheminer un paquet à travers le réseau en utilisant le routage source, la ressource expéditrice consulte sa table de routage pour obtenir le chemin complet pour l'accès à la destination souhaitée. Ce chemin est alors écrit dans le champ dédié dans l'entête du paquet. Le paquet est transféré par la suite au réseau par l'interface ressource-réseau (IRR). Le paquet doit suivre le chemin en traversant le réseau vers sa destination. Chaque routeur qui reçoit ce paquet lit le champ de chemin dans l'entête du paquet et le transmet au port de sortie destiné. Contrairement à un routeur utilisé dans le routage distribué, ce routeur ne nécessite aucun calcul supplémentaire pour prendre les décisions de routage du fait que les paquets contiennent déjà des décisions pré-calculées.

Un très grand nombre d'algorithmes de routage ont été proposées dans la littérature [64] [65] [66] [75]. Toutes les propositions correspondent aux types de routage distribué. Le routage source n'est pas beaucoup utilisé jusqu'à présent pour les NoC, en raison de son apparente surcharge (*overhead*) pour stocker les informations du chemin dans l'entête. Cependant, les chemins dans le routage source sont pré-calculés hors ligne (*offline*), donc le routage source ne peut fournir qu'une adaptabilité, très limitée, au chemin, en cas de panne ou de congestion de trafic. Mais, le routage source dispose de nombreux avantages par rapport au routage distribué.

a. Avantages

Le routage source n'est peut-être pas approprié pour les réseaux dynamiques où la taille et la topologie du réseau évoluent. Mais dans un NoC avec une taille et une topologie fixes comme le 2D Mesh, les informations de trajet peuvent être codées de manière efficace avec un petit nombre de bits, deux bits sont suffisants pour coder chaque saut dans le chemin. Les avantages suivants du routage source font plus que compenser ses inconvénients [23] [71] [76].

i. La vitesse

Le principal avantage est la vitesse. Une fois qu'un chemin est choisi dans le tableau et inclus dans l'entête des paquets par la source, aucun temps supplémentaire n'est consacré au routage.

Quand chaque paquet arrive à un routeur intermédiaire, il peut sélectionner immédiatement son port de sortie à partir des informations de chemin pré-calculé dans l'entête du paquet sans aucun calcul ou accès à la mémoire. Ainsi, un routeur utilisé pour le routage source est plus rapide que celui utilisé pour le routage distribué.

ii. La conception du routeur est plus simple

Tandis que le paquet entrant au routeur contient la décision pré-calculée sur le port de sortie, il n'y a aucune nécessité d'une logique de routage ou des tables de routage dans le routeur et, par conséquent, la conception du routeur est considérablement simplifiée et sa mise en œuvre est aussi moins coûteuse.

iii. L'indépendance de la topologie

Le routage source est indépendant de la topologie. Il peut router des paquets sur tout type de topologie, régulière ou pas régulière, à condition qu'elle ne change pas de façon dynamique au cours du routage. Il convient de noter que cet avantage du routage source est limité par le nombre de ports du routeur, la taille de la table source et la longueur maximale d'un chemin.

iv. La mixité, routage minimal et non-minimal

Dans le cas du routage source, les avantages du routage non-minimal peuvent être appliqués. Le routage non-minimal, propose un certain nombre d'avantages tels que la répartition de la charge de liaison, le contrôle de congestion et la tolérance aux pannes. Le routage source offre aussi la possibilité de mélanger des chemins minimaux et non-minimaux.

v. L'évolutivité

Étant donné que seulement un nombre constant de bits de l'entête sont utilisés dans chaque routeur, sa conception est non seulement simple mais aussi indépendante de la taille du réseau. Les routeurs qui utilisent le routage source peuvent être utilisés dans des réseaux de taille arbitraire, parce que tous les limitations sur l'évolutivité du réseau, y compris la taille du réseau, la taille de la table source et la longueur du chemin sont déterminés par la source. Cela est un avantage majeur par rapport au routage distribué où le champ d'adresse de destination dépendra de la taille et la topologie du réseau.

vi. La facilité de changer de chemins

Dans le routage source, il est possible de changer de chemins très facilement, car la table de routage est stockée dans la mémoire source. Les chemins peuvent même être modifiés dynamiquement.

vii. La répartition de charge

Etant donné que l'itinéraire de l'ensemble du trajet depuis la source vers la destination est pré-calculé et fourni dans l'entête du paquet, les chemins correspondants au trafic de l'application cible seront calculés de façon optimale (le NoC est utilisé dans un système embarqué spécifique à une application donnée). Donc, nous pouvons avoir un bon profil du trafic de communication dans le NoC. Cela nous permet d'analyser le trafic pour calculer hors ligne des chemins efficaces et optimaux donnant les caractéristiques de performances souhaitées au NoC, telles que la distribution uniforme de la charge de liaison. D'autre part, dans le routage distribué, une certaine logique supplémentaire est nécessaire dans les routeurs pour maintenir l'état des ports des routeurs voisins dans le but d'équilibrer la charge et il est très difficile de répartir uniformément la charge de la liaison quand le routage réparti est utilisé.

viii. Pas de problème de live-lock

Le *live-lock*, qui peut être traduit par l'interblocage actif, est la situation dans laquelle un paquet ne cesse de voyager dans le réseau et ne jamais atteindre sa destination, comme dans une boucle infinie. Dans le routage source, il ne peut pas y avoir de problème de *live-lock*. Puisque, un paquet prend un nombre fini de sauts avant d'atteindre sa destination, du fait que le trajet de la source à la destination est de longueur fixe. Le problème *live-lock* peut survenir avec certains algorithmes de routage distribué, mais il peut être évité avec la construction minutieuse des tables de routage.

ix. Débit garanti

Le routage source est meilleur lorsque le débit garanti est exigé, surtout dans le cas de trafic en temps réel. Ceci peut être réalisé en attribuant des "chemins spéciaux" à une telle communication.

x. Livraison des paquets en ordre

Le chemin unique pour chaque paire dans le réseau évite le problème de la livraison alternée de paquets (en désordre), ce qui est fréquent dans le cas des algorithmes de routage adaptatif.

xi. Bon pour le dépannage du réseau

Le routage source peut également être utilisé pour dépanner un réseau. Si un lien du réseau est cassé ou n'importe quel routeur est en panne, alors les paquets de test sont envoyés à toutes les destinations en utilisant le routage source. Lors de la réception des paquets de test, les

destinations répondent en envoyant des paquets de réponse à l'expéditeur. Ainsi, les liens défectueux ou les routeurs en panne peuvent être identifiés facilement en analysant les paquets de réponse reçus (c.-à-d. celui qui ne renvoie pas de réponse est défectueux).

b. Inconvénients

Tous les avantages du routage source mentionnés ci-dessus sont au prix d'un certain nombre d'inconvénients qui sont discutés ci-dessous en détail.

i. Routage Overhead (surcharge)

Dans le routage source, un paquet doit porter les informations de routage dans l'entête du paquet, augmentant ainsi la taille du paquet. L'entête de paquet dans le routage source est plus grande comparée à celle du routage distribué.

ii. Nature statique et non-adaptive du routage source

Le routage source est statique par nature. Cela signifie que le chemin ne peut pas être modifié une fois que le paquet a quitté la source, à moins que certaines techniques sophistiquées soient utilisées. Le routage source ne prend pas en compte la tendance actuelle du trafic dans le réseau. Même si une certaine sorte d'adaptabilité peut être introduite en gardant plus d'un chemin pour chaque destination dans les tables sources, le routage source ne peut pas atteindre l'adaptabilité complète. En outre, le routage source est incapable de travailler dans la présence de panne dans le réseau, sauf si des techniques de tolérance aux pannes seront introduites.

iii. Limitation de la taille des tables sources

Dans le routage source, il existe une limitation de la taille de la table de routage stockée dans la source. Stocker de grandes tables dans les sources peut devenir très coûteux, volumineux et diminue la performance des ressources, en particulier pour les ressources qui ne sont pas de type processeur.

iv. Évolutivité

Dans le routage source, il existe une limitation de la longueur maximale de l'itinéraire, c'est-à-dire le chemin ne peut pas tenir dans le flit, à moins qu'une certaine technique spéciale soit utilisée. Cette technique devrait permettre une taille variable à l'information de trajet. Cela permettra d'accroître la surcharge du chemin et la complexité de la logique de décodage.

E. Routage Minimal vs non-minimal

Un routage qui utilise les plus courts chemins possibles pour la communication est appelé routage minimal. Il est également possible d'utiliser des chemins plus longs pour le transfert de données à partir de la source vers la destination. Cette possibilité résulte de l'adaptabilité offerte par l'algorithme de routage. Le type de routage qui utilise des chemins plus longs pour la communication bien que les plus courts chemins existent est connu comme routage non-minimal. Le routage non-minimal a certains avantages par rapport au routage minimal, comme la possibilité d'équilibrer la charge du réseau et la tolérance aux pannes.

F. Routage statique vs dynamique

Dans le routage statique, le chemin ne peut pas être changé après qu'un paquet quitte la source. Dans le routage dynamique, un chemin peut être modifié à tout moment au cours du routage en fonction des conditions de fonctionnement du réseau. Le routage source est statique alors que le routage distribué peut être statique ou dynamique en fonction de l'algorithme de routage utilisé. Il convient de noter que même si les algorithmes de routage adaptatifs sont utilisés pour calculer les chemins pour le routage source, il reste statique sauf si une technique de sélection sophistiquée est introduite dans le réseau.

G. Routage pour des applications spécifiques

Ce type de routage est utilisé pour des applications spécialisées ou un ensemble d'applications concurrentes. Pour une application spécifique du SoC basé sur un réseau NoC dans les systèmes embarqués nous pouvons avoir un bon profil des communications entre les différents IPs. Cela signifie qu'il est possible de savoir préalablement quel composant communique avec quels autres composants, et qui ne communiquent pas du tout avec les autres. Afin d'obtenir les meilleures performances du NoC pour une application spécifique, nous pouvons avoir un algorithme de routage spécifique à une application spécialisée. APSRA (*A Parallel Stereo Reconstruction Algorithm with applications in entomology*) est un exemple de ce type d'algorithme [66] [75].

V.3.6.2. Algorithmes de routage à base du modèle de tours (turn-model)

Un grand nombre d'algorithmes de routage distribués pour le NoC ont été proposées dans la littérature [64] [65] [66]. Dans cette section, nous considérons uniquement les algorithmes de routage "*turn-model*". Dans ce modèle, certains tours sont restreints pour la communication en fonction des règles utilisées. La caractéristique la plus importante à prendre en considération dans un algorithme de routage est l'absence d'interblocage. Tous les modèles d'algorithmes de routage à base de tours sont sans interblocage. L'absence d'interblocage ainsi que divers algorithmes de routage à base du modèle de tours sont présentés dans ce qui suit.

A. L'interblocage

L'interblocage est le processus dans lequel la livraison de paquets est retardée indéfiniment du fait qu'un ensemble de paquets soient toujours bloqués dans le réseau. Une situation d'interblocage se produit lorsqu'un ensemble de paquets demandent en même temps les ressources du réseau en mode circulaire où qu'ils détiennent quelques autres ressources du réseau [71]. La technique de commutation de trou de ver (*Wormhole*) est plus encline à des interblocages par rapport aux autres techniques de commutation, car un paquet peut tenir plusieurs tampons dans de nombreux routeurs simultanément.

Une solution pour éviter une situation d'interblocage consiste à donner des priorités aux paquets et ainsi permettre la communication préventive. De même, l'absence d'interblocage peut être garantie par l'algorithme de routage. L'une des façons d'obtenir un algorithme de routage sans interblocage pour la topologie Mesh, par exemple, est de limiter quelques tours, comme dans le cas des algorithmes de routage sur la base du modèle de tours. Ainsi, l'absence d'interblocage peut être réalisée dans d'autres topologies en évitant les communications circulaires dans les graphes de dépendances de canal [71].

Exemple : *Dans cet exemple, nous considérons quatre routeurs "A", "B", "C", "D". Le routeur "A" a un paquet 1 dans son tampon d'entrée qui doit être envoyé au routeur "C". De même, les routeurs "B", "C" et "D" ont les paquets* 2, 3 *et* 4 *dans leurs tampons destinés aux routeurs "D", "A" et "B", respectivement. Le paquet 1 ne peut pas être transféré du routeur "A" au routeur "B" parce que le tampon de ce dernier est occupé par un autre paquet et en attente d'une autre ressource indiquée par la flèche en pointillés dans le routeur "B", comme l'illustre la figure* 1.20. *C'est également le cas avec le reste des routeurs. Ainsi, une situation d'interblocage est créée, dans laquelle chaque paquet occupe un tampon tout en demandant un autre tampon déjà occupé par un autre paquet. Dans cette situation, aucun des paquets n'est capable de se déplacer et cela se traduit par un interblocage dans le canal.*

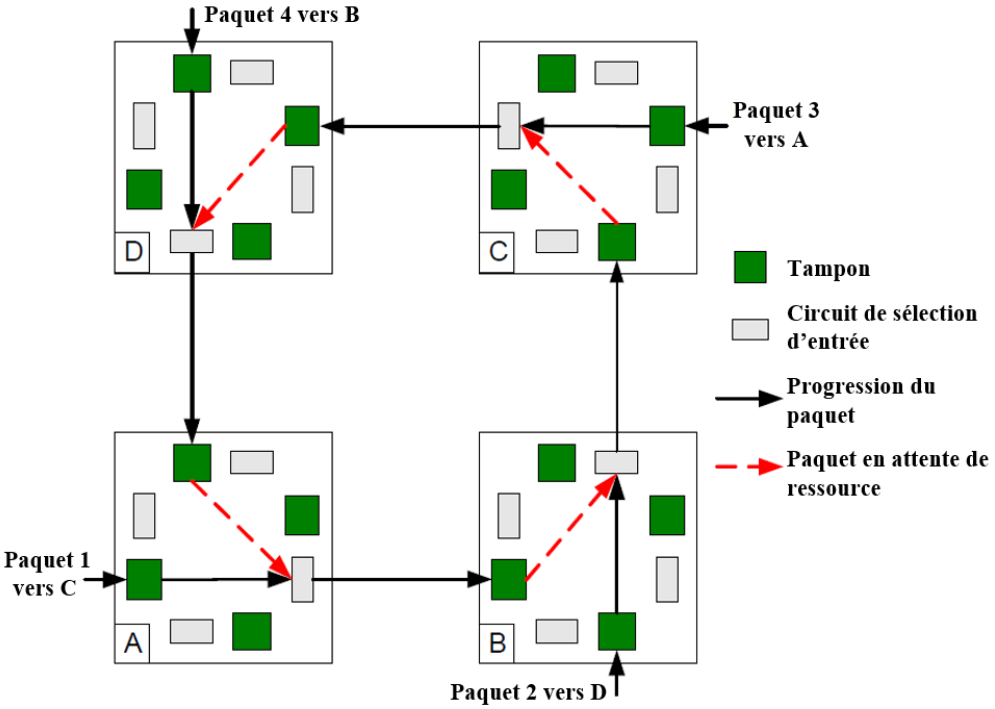


Figure 1.20 : Exemple d'interblocage dans le réseau impliquant quatre paquets

B. L'algorithme de routage XY

Il est l'un des algorithmes de routage des plus simples et des plus couramment utilisés dans le NoC. C'est un algorithme de routage statique, déterministe et sans interblocage. Sur huit tours possibles en topologie Mesh, l'algorithme de routage "XY" permet la moitié des tours en limitant la moitié restante. Selon cet algorithme, un paquet doit toujours être acheminé le long de l'axe horizontal X du Mesh jusqu'à ce qu'il atteigne la même colonne de la destination. Ensuite, il doit être acheminé le long de l'axe vertical Y en direction de l'emplacement de la ressource destinataire.

Tous les tours possibles qu'un paquet peut prendre en topologie NoC de type 2D Mesh sont présentés dans la figure 1.21 (a). De même, les tours restreints dans l'algorithme de routage "XY" sont montrés avec une croix sur la figure 1.21 (c). Dans le cas d'une source située au nœud (0,0) et d'une destination à (1,3) dans une topologie NoC de type 4x4 2D Mesh, un seul chemin est autorisée à l'aide de l'algorithme de routage "XY" et est illustré à la figure 1.22 (a).

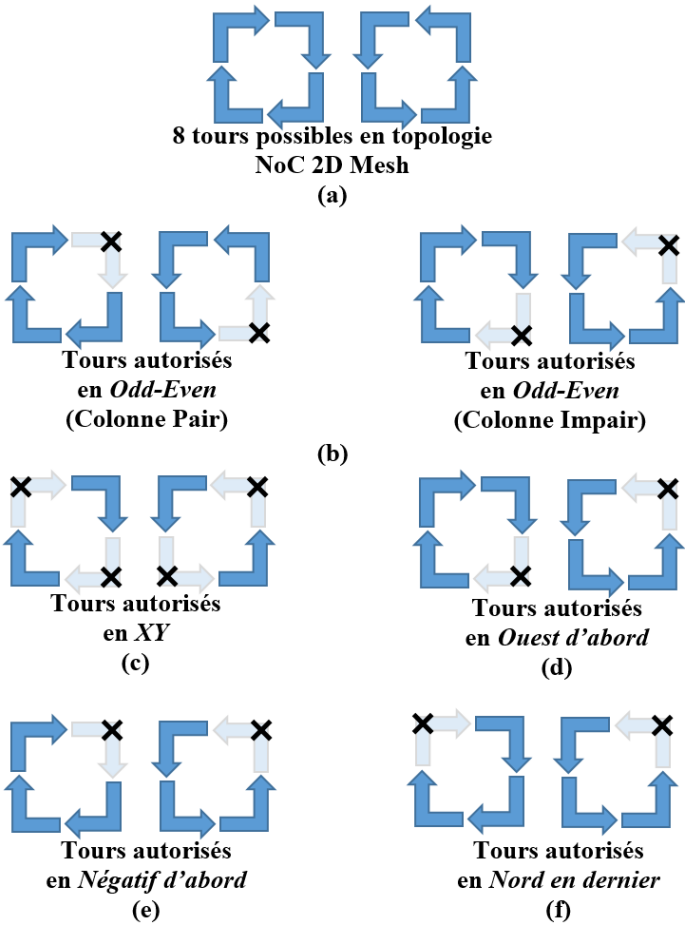


Figure 1.21 : Restrictions dans le modèle tour algorithmes de routage à base de topologie maillée NoC

C. L'algorithme de routage pair-impair (Odd Even)

Il s'agit d'un algorithme de routage partiellement adaptatif. Il restreint les tours Est-Nord ou Est-Sud à n'importe quel nœud situé dans une colonne pair du Mesh. De même, dans une colonne impair, il restreint les paquets de prendre le tour Nord-Ouest ou Sud-Ouest.

Les restrictions dans l'algorithme de routage "pair-impair" sont illustrées dans la figure 1.21 (b). L'algorithme de routage "pair-impair" fournit plus de degré d'adaptabilité par rapport au reste d'algorithmes de routage partiellement adaptatif. Pour une source située au nœud (0,0) et une destination à (1,3) dans une topologie NoC de type 4x4 2D Mesh, les chemins permis par l'algorithme de routage "pair-impair" sont présentés dans la figure 1.22 (b). On notera que les chemins ne sont pas tous autorisés pour cette communication.

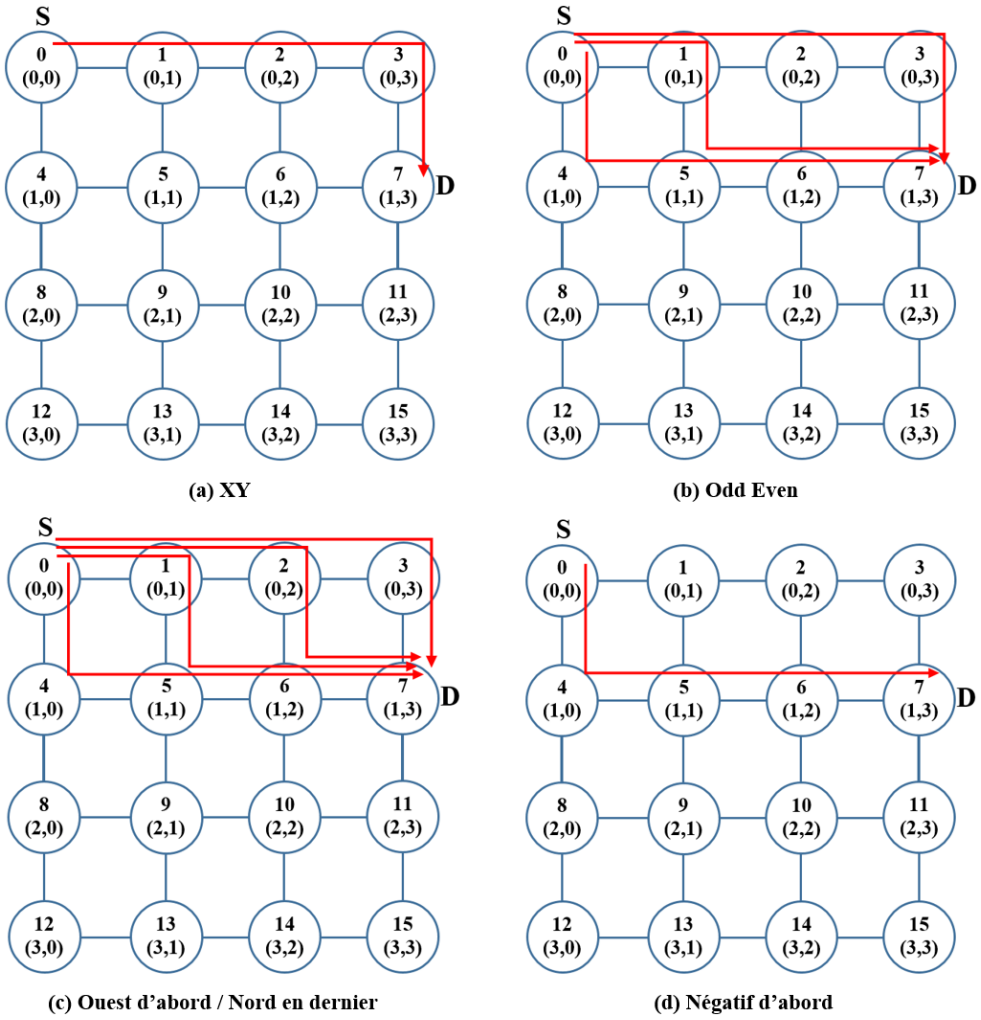


Figure 1.22 : Chemins autorisés dans les différents algorithmes de routage pour une communication spécifique sur la topologie NoC de type 4x4 2D Mesh

D. L'algorithme de routage Source (Source Routing)

L'algorithme de routage "Source" peut être utilisé pour tout type de topologies, régulières ou non régulières. L'entrée nécessaire pour l'algorithme de routage "Source" est un code appelé code de la route (*route-code*), qui représente toute la trajectoire à partir de la source jusqu'à la destination. Tous les tours possibles qu'un paquet peut prendre en topologie NoC de type 2D Mesh sont permis dans le cas de l'algorithme de routage "Source", comme le montre la figure 1.21 (a). A chaque routeur il pourrait y avoir plusieurs directions possibles (selon le nombre de ports de chaque routeur) sur lesquels un paquet peut être routé. Par exemple, le cas d'un routeur à quatre ports, les directions possibles pour un paquet sur ce routeur sont quatre ports vers les routeurs voisins, soit le *North* (*N*), le *South* (*S*), le *East* (*E*) et le *West* (*W*), en plus d'un port primordial pour chaque routeur qui le relié à la ressource attachée à ce routeur, soit le *Core* (*C*) [77].

Chaque direction est représentée par un code décimal, ce code est ensuite représenté sur *n* bits en binaire comme indiqué dans le tableau 1.2. Le nombre de bits *n* sur lesquels le code est représenté dépond du nombre de ports possibles sur les routeurs du réseau NoC. Dans cet exemple, le nombre de bits sera de 3 bits, ce qui donne une possibilité de représenter jusqu'à 8 ports (largement suffisant pour nos 5 ports). Donc, à n'importe quel routeur, les 3 bits (*n* bits dans le cas général) les plus à droite du code de la route déterminent la direction de sortie. L'algorithme de routage "Source" est mis en œuvre par la lecture des 3 bits les plus à droite du code de la route, puis de décaler à droite le code de 3 bits.

Les ports	Le code décimal	Le code binaire
North (N)	0	000
South (S)	1	001
East (E)	2	010
West (O)	3	011
Core (C)	4	100

Tableau 1.2 : Les codes des ports pour le routage source

Exemple : *Nous expliquons la méthode pour générer du code de la route pour un chemin particulier à l'aide d'un exemple. Considérons la topologie NoC de type* 4x4 2D Torus *présentée à la figure* 1.23*. Nous allons calculer le code de la route pour l'envoi de paquet du nœud* "13" *vers le nœud* "1" *le long du chemin* 13→12→0→1*. Ce chemin peut être représenté par une séquence de direction W→S→E→C (c.à.d. West→South→East→Core). Le code de la route peut être obtenu en écrivant les codes de direction pour ces directions dans l'ordre inverse, tout en maintenant l'ordre de 3 bits dans chaque code de direction. Ainsi, le code de la route pour W* (011) →*S* (001) →*E* (010) →*C* (100) *est égale à* 100 010 001 011 = 2187*. Sinon le code de la route en décimal peut être obtenu en multipliant plusieurs fois par* 8 (2^n *dans le cas général*) *et en ajoutant les codes de direction dans l'ordre inverse. Ainsi, le code de route pour W* (3) →*S* (1) →*E* (2) →*C* (4) *est égale à* (((((4 × 8) + 2) × 8) + 1) × 8) + 3 = 2187.

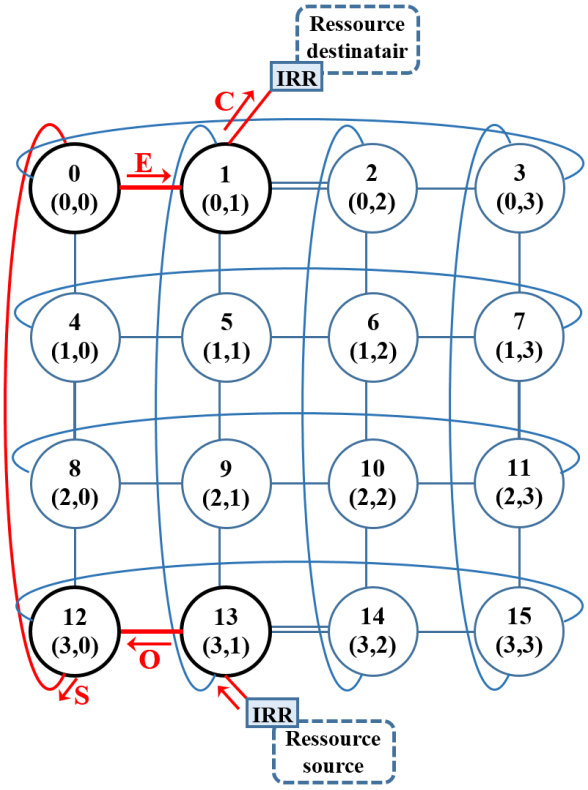


Figure 1.23 : Topologie 4x4 2D Torus

E. L'algorithme de routage Ouest d'abord (West-First)

L'algorithme de routage "Ouest d'abord" est également un algorithme de routage partiellement adaptatif. Il limite les tours Sud-Ouest ou Nord-Ouest à n'importe quel nœud du réseau maillé. Cela signifie que si une communication nécessite le déplacement d'un paquet vers le sud avec une autre direction, le paquet doit être acheminé d'abord vers le sud.

Les restrictions de l'algorithme de routage "Ouest d'abord" sont illustrées dans la figure 1.21 (d). L'algorithme "Ouest d'abord" restreint au moins de la moitié la communication source-destination à un chemin minimal, tandis que le reste des paires peuvent communiquer avec une totale adaptabilité. Par conséquent, l'algorithme de routage "Ouest d'abord" fournit moins de degré d'adaptabilité que l'algorithme de routage "paire-impaire" [64]. Pour la source située au

niveau du nœud (0,0) et la destination à (1,3) dans une topologie NoC de type 4x4 2D Mesh, les chemins autorisés par l'algorithme de routage "Ouest d'abord" sont présentés dans la figure 1.22 (c). Il convient de noter que l'algorithme de routage "Ouest d'abord" fournit une totale adaptabilité en permettant tous les plus courts chemins possibles pour cette communication.

F. L'algorithme de routage Négatif d'abord (Negative-First)

Il s'agit d'un algorithme de routage partiellement adaptatif. Il limite les tours Nord-Ouest ou Est-Sud à n'importe quel nœud du réseau en topologie maillée. Cela signifie que si une communication nécessite le déplacement d'un paquet vers n'importe quel axe négatif, horizontal ou vertical, avec toute autre direction, le paquet doit être acheminé d'abord dans cette direction de l'axe négatif, puis à l'extrémité vers l'autre direction.

Les restrictions dans l'algorithme de routage "Négatif d'abord" sont présentées dans la figure 1.21 (e). Pour une source située au nœud (0,0) et une destination à (1,3) dans une topologie NoC de type 4x4 2D Mesh, l'algorithme de routage "Négatif d'abord" ne fournit aucune adaptabilité et donc seul un chemin d'accès est disponible pour cette communication, comme le montre la figure 1.22 (d).

G. L'algorithme de routage Nord en dernier (North-Last)

C'est un autre algorithme de routage partiellement adaptatif. Il limite les tours Nord-Ouest ou Nord-Est à n'importe quel nœud du réseau de topologie maillée. Cela signifie que si la communication nécessite le déplacement d'un paquet vers le nord suivi par une quelconque autre direction, le paquet doit être acheminé d'abord vers l'autre direction et à la fin vers le nord. Les restrictions dans l'algorithme de routage "Nord en dernier" sont illustrées à la figure 1.21 (f). Cet algorithme restreint au moins la moitié des communications source-destination à un chemin minimal, tandis que le reste des paires peuvent communiquer avec une totale adaptabilité. Par conséquent, l'algorithme de routage "Nord en dernier" fournit moins de degré d'adaptabilité par rapport à l'algorithme de routage "paire-impaire". Pour une source située au nœud (0,0) et une destination à (1,3) dans une topologie NoC de type 4x4 2D Mesh, l'algorithme de routage "Nord en dernier" offre une complète adaptabilité en permettant tous les chemins les plus courts possibles pour cette communication, comme illustré à la figure 1.22 (c).

V.4. Paramètres de performance du NoC

Comme d'autres réseaux, la performance d'un NoC est évaluée par de nombreux critères tels que le débit, la distribution de la charge des liens, le nombre de sauts, la latence, la probabilité de perte de paquets, la tolérance aux pannes, la superficie du routeur, etc. Ces performances sont fortement liés à l'architecture du réseau NoC lui-même, il est primordial d'avoir une bonne compréhension de ces métriques et de la façon de les évaluer, afin d'être en mesure d'optimiser l'architecture du réseau. Certains de ces paramètres sont présentés ci-dessous.

V.4.1. La latence

La latence est définie par le délai moyen (en nombre de cycles) nécessaire pour transférer une unité de données (message, paquet ou flit) de la source vers la destination. Plus précisément, pour un seul paquet, la latence est le nombre de cycles écoulés depuis la première tentative d'injection du flit d'entête (*header*) jusqu'à la réception du dernier flit (*tail*) par la destination [73].

La latence est sans doute l'un des critères les plus importants servant à la caractérisation des NoCs. Dans les réseaux NoC, il y a beaucoup de facteurs qui s'ajoutent à la latence, y compris le délai de routage, le délai d'occupation des canaux, le délai de conflit et les délais de paquetisation/depaquetisation, de flitisation/deflitisation, et de synchronisation entre les routeurs.

La valeur de la latence va dépendre de deux choses : la position relative de l'émetteur par rapport au récepteur et la congestion dans le réseau. En effet, cette dernière aura des

conséquences quant aux choix d'arbitrages et de routages, qui pourront faire augmenter la latence des paquets. En outre, la taille des files d'attente dans les interfaces réseaux sont aussi impactées par la latence car plus les paquets mettront du temps à être transmis et plus le nombre de paquets en attente dans les interfaces sera grand.

Dans un cas plus général, il peut être intéressant d'estimer la latence moyenne du réseau, la latence est exprimée ici comme fonction de la charge injectée à l'entrée du réseau. En plus des caractéristiques statiques du réseau, la latence moyenne tient compte du comportement dynamique sous régime conflictuel pour lesquels plusieurs flots de données se trouvent dans l'obligation de partager les mêmes ressources (tampons, port de sortie, etc.) [73].

V.4.2. La superficie

La surface est une des premières métriques qui est évaluée car elle contribue directement au coût du circuit à fabriquer. Cependant, à partir d'une certaine échelle, sa miniaturisation génère des effets de bord (diaphonie, délais de propagation, ...) qui doivent être pris en compte (évités ou gérés) pour ne pas altérer le comportement du système. Par exemple, un algorithme de routage pour le réseau NoC peut être évalué sur la base de la taille du routeur ou de sa superficie. Donc, le meilleur algorithme de routage sera celui dont la superficie du routeur est plus petite et exige moins de blocs matériels et logiques.

V.4.3. Le débit

C'est le nombre total de paquets qui atteignent leur destination par unité de temps.

V.4.4. La charge du lien

La charge du lien est définie comme la quantité de données circulant sur chaque lien à chaque direction, à condition que les liens soient considérés comme bidirectionnels. Le retard du réseau dépend fortement de la charge des liens et augmente de façon exponentielle en fonction de la charge des liens.

V.4.5. La bande passante utile

La bande passante utile ou le débit de données effectif est une métrique caractérisant les performances d'un réseau [78]. En effet, le débit de données simple est une caractéristique permettant de quantifier le volume de données transitant dans le réseau. Ainsi, chaque élément du réseau (liens, nœuds) peut alors être caractérisé par sa bande passante qui est couramment exprimée en bit par seconde (bps). Cependant, cette valeur ne reflète pas le fonctionnement du réseau. C'est pourquoi, la bande passante utile est une métrique plus pertinente. Elle correspond au débit de données en réception cumulé sur l'ensemble des ressources connectées au réseau.

V.4.6. La bande passante de bissection

Il existe un deuxième type de bande passante permettant de caractériser un réseau. Il s'agit de la bande passante de bissection qui est calculée à partir de la largeur de bissection du réseau. Cette largeur représente le nombre minimum d'arcs permettant de diviser le graphe du réseau en deux sous-ensembles de nœuds de même cardinalité (plus ou moins un nœud). Ainsi, la somme des bandes passantes sur les fils associés aux arcs représente la bande passante de bissection. Cette-dernière est souvent utilisée pour mesurer la localité dans les communications ainsi que les goulots d'étranglement susceptibles de se produire.

V.4.7. La consommation d'énergie

La consommation d'énergie a toujours été un paramètre important de la performance dans tous types de réseaux. Les algorithmes de routage du NoC sont également évalués sur la base de l'énergie consommée par les routeurs correspondants.

V.4.8. La diversité de chemin

La diversité de chemin [23] est une métrique qui va caractériser la capacité du réseau à être tolérant aux fautes. En effet, un réseau offre de la diversité de chemin si pour tous (ou presque tous) les couples (source/destination), il existe plusieurs chemins entre eux. C'est pourquoi, lorsque la diversité de chemin est faible ou inexistante, cela peut entrainer des problèmes de contentions à travers tout le réseau.

V.4.9. La perte de paquet

La perte de paquets est également utilisée pour évaluer la performance des algorithmes de routage dans le NoC. Elle est définie comme la probabilité de perdre de paquets dans le NoC.

V.4.10. La tolérance aux pannes

La tolérance aux pannes est la capacité d'un algorithme de routage d'acheminer les paquets sans la présence de défauts dans les liens. Il s'agit d'une mesure du nombre et de type de fautes tolérées par l'algorithme.

V.4.11. Livraison de paquets en ordre

Une autre mesure pour évaluer les performances d'un NoC est la livraison dans l'ordre des paquets. Les résultats de sortie de la livraison des paquets dans l'ordre est, en fonction des cycles supplémentaires, nécessaires à l'ordonnancement des paquets à la destination. Plus le temps d'ordonnancement des paquets est long, plus les performances du réseau baissent.

V.4.12. Le diamètre

Le diamètre d'un réseau est défini comme étant la distance maximale entre deux nœuds du réseau.

Cette première métrique de performances sert aussi à classer les topologies de réseau. En effet, il existe trois types de diamètre [2] :

- à longueur de ligne fixe. C'est le cas des topologies qui ont un diamètre évoluant en O(1) en fonction du nombre de nœuds interconnectés dans le réseau. Un exemple typique est le crossbar. Ce genre de topologie n'est pas facilement extensible et est donc plus approprié pour les réseaux avec peu de nœuds ;
- linéaire. L'exemple le plus courant est celui de la topologie maillée, qui a l'avantage d'être facilement étendue ;
- logarithmique. Nous pouvons citer le cas des réseaux hyper cubiques, en arbre ou encore multi-étages. Les topologies à diamètre logarithmique conviennent mieux lorsque le nombre de nœuds à connecter devient important.

V.4.13. L'extensibilité

Avec l'évolution des technologies, l'intégration des systèmes sur puce ne cesse d'augmenter et avec elle, le nombre de composants à interconnecter. En conséquence, il est important que le réseau sur puce soit capable d'être étendu. Cette extensibilité du réseau [35] représente son aptitude à augmenter efficacement les performances en fonction du nombre d'éléments communicants. En particulier, la bande passante utile doit augmenter proportionnellement lorsque de nouveaux éléments de traitement sont rajoutés au réseau.

V.4.14. La flexibilité

La dernière métrique de caractérisation de performances, la flexibilité [79], représente le degré d'adaptation du réseau à différents éléments communicants hétérogènes et à sa capacité de réutilisation dans de futurs systèmes. Cependant, il est assez difficile de concilier ces deux exigences avec des besoins en performances. En effet, typiquement un réseau conçu spécifiquement pour une application donnée afin d'offrir des performances optimales ne pourra

pas être réutilisé pour d'autres applications. A l'inverse, un réseau plus générique sera moins performant mais plus flexible.

V.5. Synthèse

Dans ce premier chapitre, nous avons présenté les principales caractéristiques des réseaux sur puce. Puis nous avons identifié les problématiques et les limitations des architectures de communication intégrées sur le bus partagé, largement utilisés jusqu'à maintenant, avec l'accroissement des besoins en bande passante, en performances ainsi que les contraintes liées aux nouvelles technologies d'intégration. Ces limitations sont dues principalement au caractère séquentiel des communications sur de tels supports. Les crossbars quant à eux supportent très bien des charges élevées de trafics mais leur implémentation matérielle devient quasiment impossible au-delà d'une certaine taille (très couteux, réservés à des applications spécifiques). Nous avons ensuite introduit le paradigme NoC comme une solution alternative pour la communication dans les systèmes sur puce, il permet de construire des architectures flexibles et extensibles de systèmes sur puce. Dans le NoC, nous avons montré que les communications et les calculs sont bien séparés grâce au protocole réseau. L'intégration des unités de traitement est ainsi facilitée grâce à des interfaces réseaux bien définies. Puis, nous avons étudié les protocoles de communication et les concepts de base des NoCs. Les réseaux sur puce avec leur support intrinsèque au parallélisme ainsi que la régularité de leurs architectures qui constituent donc l'évolution naturelle des infrastructures de communications sur puce. Enfin, étant donné que pour une exploration efficace il est tout aussi important de pouvoir évaluer les performances d'un réseau, nous avons fait une étude exhaustive des paramètres d'évaluation de performances et de simulation usuellement employées.

Dans le chapitre suivant, nous allons présenter un large état de l'art d'une part sur les approches de personnalisation et d'optimisation de performances des NoCs, et d'autre part sur les méthodes et les outils d'évaluation des NoCs.

Pour la suite des travaux de cette thèse, nous avons choisi quelques paramètres à utiliser, comme le montre le tableau 1.3.

Type	Paramètres
Routage	L'algorithme de routage "*Source*" et/ou XY
Commutation	Commutation en trou de ver (*Wormhole Switching*)
Génération de trafic	Génération de trafic synthétiques CBR et/ou Bursty
Patterns trafic	*Uniform*, *HotSpot*, *Transpose*, *Bit-Reversal* et *Shuffle*
Contrôle de flux	Contrôle de flux par rétroaction de flux
Topologies	2D Mesh, 2D Torus, Ring, Spidergon, X-Mesh et WK

Tableau 1.3 : Paramètres NoC sélectionnés

III. Chapitre 2 : Etat de l'art : Approches d'optimisation & outils d'évaluation des NoCs

Nous présentons dans ce chapitre un état de l'art sur les approches d'optimisation et de personnalisation des NoCs (section VI.1). Ensuite, nous présentons un panorama des méthodes et des outils d'évaluation et d'analyse des performances des NoCs (section VI.2), cette section est divisée en deux sous-sections : la première sous-section concerne la présentation des méthodes d'évaluation par des modèles de systèmes, que ce soit par évaluation analytique ou par simulation (section VI.2.1), et l'autre sous-section concerne la présentation des différents outils d'évaluation par des systèmes existants (section VI.2.2). Enfin, nous présentons une synthèse qui résume nos choix de méthodes et d'outils d'évaluation, ainsi que nos objectifs et nos contributions en ce qui concerne les approches d'optimisation et de personnalisation des NoCs.

VI.1. Approches d'optimisation de performance du NoC

Les architectures d'interconnexion sur puce existantes s'appuient sur des concepts hérités des systèmes parallèles et distribués pour interconnecter des cœurs IPs de manière structurée et évolutive, ils peuvent souffrir d'une latence élevée, d'un faible débit et d'une forte consommation d'énergie lorsque ils sont adaptées aux applications SoC. Les architectures d'interconnexion sur puce adoptées pour les SoCs se composent d'un certain nombre de cœurs IPs (ex., CPU, DSP, mémoire, etc.) qui communiquent via un réseau d'interconnexion sur puce. La difficulté de la conception d'un NoC réside dans un compromis entre une qualité de service (QoS) optimale (en matière de besoins en latence, en débit et en charge de communication), une bande passante élevée et une flexibilité d'utilisation importante, tout en limitant la consommation d'énergie et la surface de silicium.

Après avoir exploré les mesures de performance pour les architectures NoC communes, d'autres paramètres tels que les besoins en énergie et de la surface de silicium doivent être étudiées, car ils sont d'une importance primordiale dans la conception de NoC par rapport à des réseaux informatiques et à des systèmes parallèles et distribués. Dans les réseaux informatiques réguliers, l'énergie consommée par les routeurs est très limitée par rapport à la taille du réseau, tandis que les exigences de la surface de silicium ne sont pas considérées vues que l'espace, couvert par le réseau, est généralement très grand par rapport à la taille des principaux composants du réseau. Le tableau 2.1 compare les architectures NoC à des réseaux informatiques réguliers et des systèmes parallèles [80].

Réseaux sur puce (NoC)	Systèmes parallèles et distribués	Réseaux informatiques
Un dégrée élevé d'hétérogénéité	Un faible dégrée d'hétérogénéité	Hétérogène (multi-plateformes)
Adapter à des applications	Usage générale d'applications	Usage générale d'applications
Importantes contraintes d'énergie (des techniques sophistiquées de routage et de commutation sont nécessaire)	L'énergie n'est pas le souci majeur	L'énergie n'est pas le souci majeur
Une topologie prédéfinie (fixée par le concepteur)	Une topologie prédéfinie (extensible)	La topologie n'est pas prédéfinie
Routeurs sophistiqués	Pas de routeurs spécialisés	Routeurs et switches

Tableau 2.1 : Comparaison des NoC à des réseaux informatiques et des systèmes parallèles

Ces dernières années, plusieurs recherches se sont concentrés sur les approches pour des applications NoC, spécifiques ou pas. Ces approches sont paramétrées au moment de la conception et/ou au moment de l'exécution.

De nombreuses études ont montré que, pour améliorer les performances d'un domaine d'application spécifique, les architectures d'interconnexion sur puce peuvent être personnalisées au moment de la conception. Par exemple, on peut insérer des longs liens entre des commutateurs stratégiques, ou allouer la bande passante nécessaire selon le trafic ou encore allouer la taille des tampons nécessaire selon une application spécifique [4] [81]. Ces approches sont généralement adaptées pour une application spécifique SoC [82]. Ils traitent la sélection d'une architecture d'interconnexion sur puce spécifique à l'application attendue lors de la phase d'exploration d'espace. La conception de réseau sur puce spécifique à l'application est récemment devenue impérative pour beaucoup de groupes de recherche et industriels. Bien que de nombreux travaux ai été présentés dans le passé sous forme d'évaluation générale des architectures NoC en utilisant un stimulus aléatoire [38] [83], la conception des applications spécifiques NoC est au centre des recherches récentes dans le domaine de la communication sur puce [84] [85].

Alors que les techniques matérielles de déploiement de circuit/architecture sont certainement très importantes pour la réduction de la consommation d'énergie des NoC, le logiciel peut également jouer un rôle important. Plusieurs équipes de recherches de par le monde explorent des solutions d'optimisation des NoCs au moment de l'exécution, ainsi l'activité du SoC est surveillée lors de l'exécution. L'optimisation au moment de l'exécution peut réduire le coût de défaillance de la conception, pour échapper à toute vérification au moment de la conception, afin de détecter la présence et la prévention de la corruption du SoC, de la perte de données et/ou la défaillance de l'ensemble du SoC. Leur coût, cependant, inclut la surface de silicium pour le suivi de l'exécution et de la récupération, l'effort de conception dédié et souvent l'impact de la performance sur le système global due aux activités de surveillance continue. Pour une utilisation efficace des ressources NoC, l'architecture de celui-ci doit être adaptative, par exemple, en re-routant au moment de l'exécution le trafic depuis les zones du NoC congestionnées, ou bien par l'allocation dynamique de la bande passante des liens selon l'évolution du trafic.

Dans [86], une approche hybride, qui divise la tâche de réduction de la consommation d'énergie entre un compilateur et l'environnement d'exécution, est présentée. Dans cette approche, le compilateur modifie le code source de l'application dans les premières itérations afin de recueillir des statistiques sur l'utilisation des liens et de décider de la tension appropriée et les niveaux de fréquence pour les liens de communication. Les itérations restantes de la boucle s'exécutent avec ces tension/niveaux-de-fréquence sélectionnées de telle sorte que la consommation d'énergie sera réduite et que la performance ne soit pas affectée de façon excessive. Cette approche est automatisée dans un compilateur et testée à l'aide des applications de MediaBench [62]. Les résultats obtenus montrent que cette approche hybride économise beaucoup plus d'énergie que le schéma de réduction d'énergie à base du matériel et/ou compilateur pur.

Les approches d'optimisation de performance du NoC, dans la phase de conception ou dans la phase d'exécution, peuvent être classées en fonction du niveau auquel l'optimisation s'effectue, c'est à dire au niveau physique, au niveau communication et au niveau application, comme l'illustre la figure 2.1 [82] [87] [88] [89] [90].

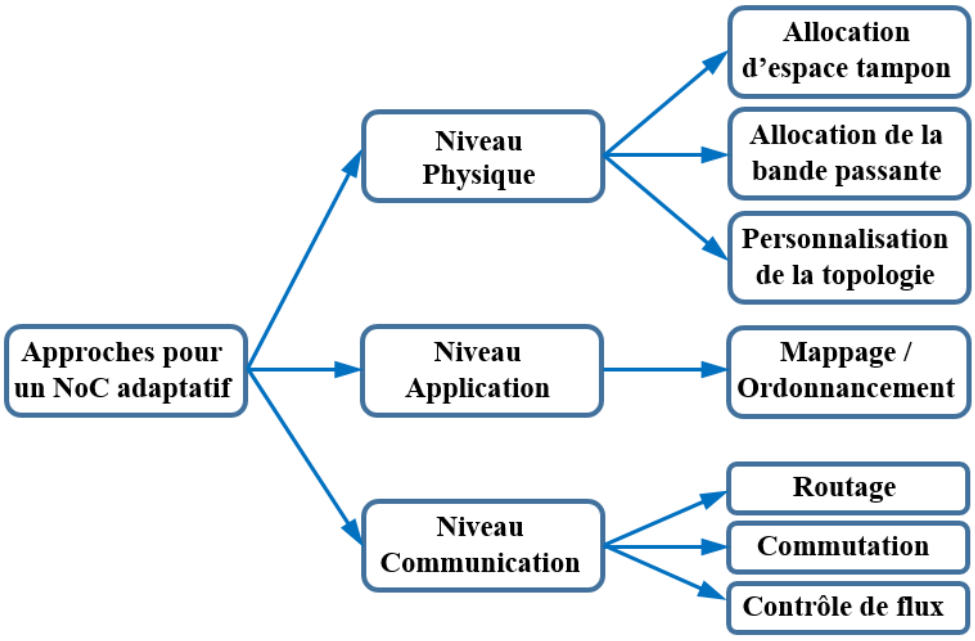


Figure 2.1 : Classification des approches de personnalisation et d'optimisation des architectures d'interconnexion sur puce

Au niveau physique, la configuration d'interconnexion sur puce, l'allocation de la bande passante et la minimisation de la mémoire tampon sont les trois principaux problématiques qui

devraient être étudiés. Au niveau communication, la commutation, le contrôle de flux du réseau et les techniques de routage des données doivent être conçues avec un grand soin. Au niveau application, le mappage (*mapping*) et l'ordonnancement (*scheduling*) des tâches des applications tout en optimisant les coûts et les mesures de performances constituent les principaux enjeux.

VI.1.1. Les approches au niveau physique

Dans le niveau physique, la performance et l'efficacité du NoC dépendent fortement de l'architecture d'interconnexion sur puce. Les commutateurs sont les composants actifs qui influencent fortement la latence et le débit de la communication sur puce, tout comme les liens qui véhiculent les données entre les commutateurs. Cependant, une allocation efficace de leur bande passante, une optimisation de leurs distances, ou une augmentation de leur nombre peut améliorer la performance des applications SoC.

Par conséquent, la personnalisation de l'interconnexion sur puce (topologie), l'allocation de la bande passante, et la minimisation de la taille des tampons sont les trois principales problématiques qui doivent être étudiées lors de la conception du NoC. Dans la suite de cette section, nous citons les approches principalement proposées pour étudier ces problématiques.

VI.1.1.1. La personnalisation de la topologie

Dans [91], les auteurs présentent une méthodologie pour la personnalisation du crossbar STBUS (*STBUS*: *Serial Telecom BUS*). Dans [4] [85], la cible de la personnalisation est la topologie du réseau, en utilisant une méthode de décomposition pour une entière personnalisation et l'insertion de longs liens sur la topologie standard 2D Mesh. Dans [92], une approche pour adapter les liaisons transversales afin de réduire la distance moyenne du réseau entre les nœuds mappés (augmentation de la performance), sans dépasser le nombre maximal prédéfinis de ports des routeurs STNoC (contrôle de la surface silicium et de la surconsommation d'énergie). L'évaluation de cette approche a été effectuée en utilisant deux topologies, le Ring et le Spidergon. D'autres approches structurelles pour la personnalisation de la communication sur puce pour une application spécifique ont été présentées dans [85] [91].

Des approches pour des réseaux sur puce reconfigurables qui permettent l'adaptation en cours d'exécution de la topologie aux exigences des applications sont proposées dans [93] [94] [95]. Ces approches sont compilées sur des composants FPGA (*Field-Programmable Gate Array*) et sont, cependant, limitée à des composants reconfigurables. Une architecture NoC, appelée ReNoC (*ReNoC*: *Reconfigurable NoC*), est proposée dans [8], pour permettre à la topologie du réseau d'être configurée par l'application exécutée sur l'architecture SoC. ReNoC combine la commutation de paquets et la commutation de circuits dans la même topologie. Cela permet de créer des topologies d'applications spécifiques dans une plateforme générale de SoC à base de NoC.

D'autres approches de personnalisation de la topologie par insertion de liens ont été proposées. Ces approches peuvent être classées en deux catégories principales : des approches de synthétisation et des approches de personnalisation [82].

La première classe concerne les techniques proposées pour personnaliser entièrement l'architecture, elles sont plus appropriées pour une application spécifique, et elles ne sont pas nécessairement adaptées aux topologies régulières (ex., 2D Mesh) [96] [97] [98] [99]. Par exemple, dans [97], un algorithme génétique est présenté pour la synthèse d'une application spécifique pour le SoC avec l'objectif d'optimiser la consommation d'énergie et de l'espace de conception tout en maximisant les performances (ex., la latence, le débit, etc.). Cet algorithme est itératif et permet la production et la sélection d'un nombre constant de configurations intermédiaires qui satisfont aux critères de fitness proposés. Cependant, trouver la meilleure configuration peut poser de sérieux problèmes, car la précision dépend de la fonction de fitness. En outre, malgré les capacités de ces techniques de maximiser la performance, en offrant des

techniques pour incorporer les architectures d'interconnexion sur puce synthétisées qui sont plus adaptées pour une application spécifique, ils peuvent souffrir du fait de fournir des architectures avec une grande régularité dans un délai raisonnable [100].

L'autre classe concerne les approches de personnalisation partielle de la topologie. En revanche, elles commencent principalement par une architecture d'interconnexion sur puce régulière comme le 2D Mesh. Puis, elles personnalisent partiellement l'architecture de base en insérant des longs liens. Par exemple, une approche de personnalisation d'un 2D Mesh standard en insérant quelques longs liens spécifiques à l'application a été proposée dans [101]. L'algorithme commence par une architecture régulière de 2D Mesh et ajoute progressivement des liens à longue portée entre les commutateurs selon la fréquence de communication la plus élevée jusqu'à ce que toutes les ressources (liens) disponibles sont utilisées. Les résultats analytiques et de simulation ont été présentés pour montrer que cette approche permet de réduire considérablement la latence moyenne des paquets. Toutefois, le choix stratégique des liens à insérer sur le réseau d'interconnexion sur puce qui convient le mieux aux besoins de plusieurs applications est difficile sans connaître à l'avance leurs correspondances avec les modèles de trafic.

VI.1.1.2. Allocation de la bande passante

Les approches de personnalisation de la topologie suppose que tous les liens du NoC soient identiques, c'est à dire, avec des capacités/bande passante uniformes. Dans le NoC, si les liens ont la même bande passante, certains d'entre eux pourraient être beaucoup plus grands que ce qui est nécessaire, en consommant de l'énergie et de l'espace silicium.

Par conséquent, des approches d'allocation de bande passante ont été proposées, elles assignent une capacité suffisante pour des liens selon le trafic afin de réduire la latence et de minimiser la consommation d'énergie. Par exemple, une approche basée sur un modèle analytique de retard pour les réseaux à commutation en mode trou de ver à canal virtuel avec des liens non-uniformes est proposée en [102]. Le groupe de recherche de Philips a développé un flot de conception complet autour de son réseau Æthereal [103] [104]. Il est basé sur des méthodes et outils pour automatiser les étapes de son flot de conception. Il offre ainsi un flot prenant comme contrainte la bande passante et la latence des communications.

Dans [105], un réseau sur puce à bande passante adaptative a été proposé pour répondre aux problèmes de complexité du routage adaptatif et aux problèmes de performance du routage inconscient (en anglais, *oblivious routing*), lorsque les chemins sont choisis à l'avance pour toute paire de source-destination et tout message doit emprunter l'un de ces chemins. Dans ce réseau à bande passante adaptative, la largeur de la bande passante du réseau peut s'adapter aux conditions changeantes du réseau. La reconfiguration de chaque lien peut être faite très rapidement en réponse à l'évolution des besoins de trafic. La méthode est évaluée sur un exemple concret de liens bidirectionnels d'un 2D Mesh. Les résultats ont montré que les liens bidirectionnels adaptatifs offrent de meilleures performances.

Dans [86], une méthode de conception de bande passante est proposée pour personnaliser la bande passante du lien dans le NoC selon le trafic. La méthode permet au NoC de satisfaire aux exigences de performance à moindre coût de bande passante, et ainsi d'obtenir le même niveau de performance qu'un NoC avec une bande passante uniforme des liens toute en utilisant moins de ressources de bande passante, ce qui diminue la surface silicium du NoC et la consommation d'énergie.

Une autre approche, appelée 2X-Links, a été proposée dans [89] pour permettre aux liens de changer, en cours d'exécution, leur bande passante supportée. La capacité des liens n'est pas uniquement paramétrable en cours de conception, mais peut être aussi adaptée en cours d'exécution en fonction des conditions de trafic réels.

VI.1.1.3. Allocation de l'espace tampon

Plusieurs approches analytiques d'allocation de l'espace tampon pour la conception des routeurs NoC ont été proposées dans la littérature. Par exemple, dans [106], un modèle à l'aide d'un diagramme cyclo-statique de flux de données a été utilisé pour le dimensionnement des tampons pour les applications NoC. Pour montrer l'efficacité de ce modèle, les résultats analytiques ont été comparés à ceux de la simulation. Dans [107], un algorithme a été proposé pour attribuer automatiquement l'espace de mémoire tampon pour chaque canal d'entrée pour correspondre au modèle de communication d'une application cible et au budget d'espace tampon initial. La plupart des approches stochastiques considèrent le trafic entrant et sortant comme une distribution de probabilité (ex., trafic de Poisson). Toutefois, les applications NoC fournissent des modèles de trafic qui sont très différents par rapport au modèle de Poisson comme dans la théorie des files d'attente [15] [108]. Plus précisément, le modèle de Poisson ne parvient pas à capter certaines caractéristiques importantes des réseaux comme l'autosimilarité ou la dépendance à long terme [100].

Dans [109], les auteurs ont suggéré la physique statistique et la théorie de l'information pour étudier le comportement des tampons. La physique statistique peut modéliser les interactions entre les différentes composantes tout en considérant les effets de la mémoire à long terme. Le principal concept de ce modèle est que les paquets dans le réseau se déplacent de nœud en nœud d'une manière similaire à des particules qui se déplacent dans un gaz de Bose et leur migration entre différents niveaux d'énergies à la suite de variations de température. Plus précisément, l'approche repose sur un réseau virtuel de plus en plus aléatoire, qui décrit l'évolution des tampons NoC en fonction du taux d'injection de paquets. L'analyse de la performance du modèle proposé prédit que l'occupation de mémoire tampon suit une distribution en loi de puissance.

D'autres approches basées sur la théorie de systèmes, qui est appliquée avec succès à la conception des circuits électroniques, ont été proposées. Par exemple, dans [110], *Network Calculus* [111] [112] a été utilisé pour analyser et évaluer les mesures de performance des architectures d'interconnexion sur puce. Il a été démontré que la théorie de *Network Calculus* peut être utilisée pour trouver la taille des tampons de chaque canal qui conviennent le mieux aux caractéristiques du trafic. Cependant, *Network Calculus* était utilisé pour l'analyse de pire-cas et peut fournir la profondeur des tampons et la latence de bornes supérieures (*upper-bounds*).

Un modèle espace-état, basé sur la modélisation par réseau à compartiments, pour le NoC avec un contrôleur de l'observateur d'état est proposé dans [113]. Les auteurs se sont focalisés principalement sur le contrôle des débits d'entrée et de sortie et sur la surveillance des débits intermédiaires des routeurs sur puce afin de réduire la congestion et stabiliser le réseau. Dans [16], un algorithme d'allocation d'espace tampon au niveau système est présenté, il peut être utilisé pour personnaliser la conception du routeur dans le NoC. Plus précisément, selon les caractéristiques du trafic d'une application cible et un budget d'espace de mémoire tampon initial, l'algorithme alloue automatiquement la taille de tampon nécessaire pour chaque canal d'entrée en fonction du mode de communication, de façon à ce que la performance globale du NoC sera maximisée. Les résultats de simulation d'une application audio/vidéo ont montré un gain de 85% sur l'utilisation des ressources tampons, sans diminuer les performances du NoC.

Dans [114], un nouveau modèle de gestion dynamique des tampons d'entrée du routeur est présenté, il ajuste en permanence la rigidité de la boucle de rétroaction du contrôle de flux selon les conditions de trafic observées. Ce modèle conduit à une distribution plus efficace de l'espace tampon et améliore l'isolation entre plusieurs charges de travail s'exécutant simultanément avec différentes caractéristiques de performance. Les résultats de simulation montrent que ce modèle améliore la stabilité du réseau, conduisant à une augmentation moyenne du débit de 2,6 fois et une réduction de 31% de la latence moyenne.

VI.1.2. Les approches au niveau communication

Au niveau communication, trois questions principales sont définies dans la conception de NoC adaptatif, la commutation, le contrôle de flux du réseau et le routage des données. Par exemple, une latence plus faible peut être obtenue par la mise en œuvre d'un routage adaptatif qui évite les routeurs défectueux ou surchargés, ou en utilisant une technique de commutation adaptative ou bien en utilisant une technique sophistiquée de contrôle de flux.

VI.1.2.1. Le routage

Compte tenu de l'architecture NoC qui détermine la performance globale du système, le routage est devenu la stratégie la plus importante à considérer pour la conception du NoC. Les stratégies de routage peuvent être classées en régimes déterministes et adaptatifs.

Dans une stratégie de routage déterministe, la source et la destination déterminent le trajet à traverser. Les systèmes de routage déterministes populaires pour le NoC sont le routage source et le routage XY [23]. Dans le routage source, la source spécifie le chemin vers la destination. Dans le routage XY, le paquet suit les lignes d'abord, puis se déplace le long des colonnes vers la destination, ou vice versa. Cependant, le routage XY est limité à des topologies de réseau régulières.

Dans une stratégie de routage adaptatif, le chemin de la traversée est décidé sur une base par-saut. Les systèmes adaptatifs impliquent un arbitrage dynamique et des mécanismes de sélection du saut suivant, c'est à dire, sur la base de la congestion locale du lien. Le calcul du coût minimum (ou le plus court chemin) est fondamentale pour les différentes stratégies dynamiques de routage. L'idée de base est que l'algorithme de routage choisit toujours le chemin le moins congestionné vers la destination grâce à la planification de trajectoire optimale. L'itinéraire moins congestionné peut être trouvé sur la base du calcul de chemin le plus court où le coût du chemin est obtenu lors de l'exécution. Depuis l'état du réseau, tels que l'intensité et les conditions de trafic, qui changent au moment de l'exécution, l'algorithme de routage dynamique devrait être en mesure de découvrir les commutateurs congestionnés et d'effectuer le calcul du plus court chemin en même temps.

Il existe plusieurs algorithmes de routage adaptatif qui ont été proposés dans le contexte du NoC [60]. Par exemple, une méthode qui met l'accent sur le routage adaptatif sans interblocage a été proposée dans [75], elle fournit un cadre pour concevoir des tables de routage qui peuvent surperformer l'algorithme de routage sans interblocage, fondée sur le modèle de tour. D'autres systèmes, tels que le routage adaptatif pair-impair [115] [116] et la sélection adaptative de nœud-sur-chemin (*NOP* : *Node-On-Path*) [117], fournissent également de l'adaptabilité au routage mais ils exploitent seulement le trafic local ou les conditions des voisins. Il y a un grand potentiel pour améliorer l'efficacité de la communication en tenant compte du trafic global au moment de l'exécution en utilisant le routage adaptatif, comme la surveillance du trafic global [118] et le routage adaptatif global [119]. Dans [120], les auteurs ont évalué une technique de contournement qui permet à un trafic sélectionné d'éviter les fonctionnalités complètes de routage de nœuds sélectionnés dans le NoC. Les résultats ont montré que cette technique réduit considérablement la consommation globale d'énergie. Une approche de routage, appelée routage dynamique XY (*DyXY*), est proposée dans [121] pour éviter que les paquets soit transmets aux commutateurs congestionnés. Dans [122], une architecture de communication sur puce adaptative AdNoC (*AdNoC*: *Adaptative NoC*) a été proposée. Elle fournit un algorithme adaptatif d'allocation de routes pour répondre aux différentes garanties de bande passante et un schéma d'attribution sur demande de bloc de tampon pour l'établissement de connexion au moment de l'exécution entre un producteur et un consommateur de données. Ces stratégies utilisent des approches à base de règles ou des heuristiques pour l'adaptation de trafic. On utilisant un calcul de chemin le plus court à la demande pourrait améliorer l'optimalité du routage et l'efficacité de l'adaptabilité. Une approche de routage dynamique basé sur un réseau dynamique de programmation (*DP-Network*: *Dynamic Programming Network*) est proposée

dans [123]. Cette architecture fournit, en temps réel, le calcul du plus court chemin et la planification de la trajectoire optimale.

VI.1.2.2. La commutation

Le mode de commutation, utilisé pour spécifier comment les données et les contrôles sont acheminés, est un facteur pour déterminer la performance du NoC [70]. Deux modes de commutation sont distingués, la commutation de paquets et la commutation de circuits [7].

La commutation de paquets consiste à découper les données à communiquer en plusieurs paquets et à envoyer chacun d'entre eux sur un lien de façon déférente afin d'en accélérer le transfert. Les délais de communications sont non-déterministes, du fait des contentions des ressources de communications. Ce type de commutation nécessite un contrôle de flux et de congestion.

La commutation de circuits, quant à elle, est basée sur la construction d'un chemin unique exclusif entre la source et la destination, lors de l'établissement d'une séquence de dialogue entre ces deux ressources communicantes. Le chemin, appelé un circuit, ainsi créé reste jusqu'à la clôture de la séquence de dialogue.

Dans [124], un NoC appelé SoCBUS est proposé et utilise une méthode hybride de commutation de paquets-circuits appelé PCC (*PCC*: *Packet Connected Circuit*). Cette approche utilise le routage d'un petit paquet qui traverse le réseau d'interconnexion sur puce et met en place un itinéraire en circuit commuté pour que les flits de données utiles le suivent. Dans SoCBUS, quand un flit est bloqué, il ne sera pas stocké dans le tampon du commutateur, mais il sera rejeté par le contrôle de flux de la commutation de paquets. Cette technique élimine le besoin en taille de tampon et les pénalités de la latence de communication.

Une autre technique hybride de commutation de circuits est proposée dans [125], cette technique surmonte avec succès certains des inconvénients associés à la commutation de circuits, en particulier, en éliminant le temps de la configuration et la reconfiguration des circuits instantanément et le temps du chevauchement des flits à commutation de paquets avec des flits à commutation de circuits. Les auteurs ont évalué la commutation de circuit traditionnelle et ils l'on comparée avec la combinaison hybride de la commutation de circuit et la commutation de paquets. Les résultats obtenus montrent que cette technique hybride délivre des bénéfices de latence pour la commutation de circuits, toute en maintenant les bénéfices de débit de la commutation de paquets, avec peu de surface silicium et peu d'énergie.

Un routeur à commutation de circuit reconfigurable est proposé dans [126], trois applications sont évaluées et les résultats montrent que le routeur NoC à commutation de circuits satisfait les exigences pour les trois différentes applications. En plus, le routeur à commutation de circuit a une faible consommation d'énergie, une petite surface de silicium et un maximum débit par direction comparé à la commutation de paquets.

Dans [127], une autre approche a été proposée afin de permettre au NoC de se configurer dynamiquement par rapport aux modes de commutation tenant compte des exigences de changement de communication du système. De même, un mécanisme de communication sur puce en intégrant un réseau commuté à commutation de paquets et de circuit en une seule architecture a été proposé dans [9]. La partie à commutation de paquets fournit la commutation de paquets traditionnelle tandis que le sous-réseau à commutation de circuits dirige les paquets sur des circuits qui contournent certains nœuds intermédiaires. Les résultats de l'évaluation démontrent une faible consommation d'énergie et une amélioration des performances, par rapport à un NoC classique à commutation de paquets.

VI.1.2.3. Le contrôle de flux

Les techniques de contrôle de flux de réseau traitent l'espace tampon limité dans les commutateurs. Dans la commutation de circuits, les techniques de contrôle de flux ne sont pas nécessaires tant que le circuit est affecté dans la phase de configuration. Ces techniques ne sont nécessaires que lorsque le mode de commutation de paquets est utilisé, parce que les paquets doivent être tamponnés avant de les envoyer à d'autres commutateurs ou des cœurs IP. Les techniques de contrôle de flux peuvent être classées en fonction de leur granularité de l'allocation de la bande passante du canal et de l'allocation de la mémoire tampon [23]. Les deux principales techniques de contrôle de flux de haut niveau sont le contrôle de flux en mode trou de ver (*wormhole flow control*) et le contrôle de flux en mode canal virtuel (*virtual-channel flow control*).

Des études récentes ont montrées que le la technique du trou de ver nécessite peu d'espace tampon, ce qui diminue la latence quand il n'y a pas de problème de blocage. Cependant, quand le premier flit est bloqué, tous les flits de ce paquet seront aussi bloqués sur des commutateurs intermédiaires. Une approche de régulation dynamique des canaux virtuels (*ViChaR : Virtual Channel Regulator*) est proposée dans [88] pour allouer dynamiquement des canaux virtuels et les ressources de mémoire tampon en fonction des conditions de trafic du réseau. Dans cette technique, le nombre de canaux virtuels VCs (*VC*: *Virtual Channels*) et la taille de tampon par VC sont dynamiquement ajustés en fonction de la charge du trafic. Les résultats présentés montrent que *ViChaR* réalise une performance similaire avec la moitié de la taille de la mémoire tampon d'un routeur générique. Alors que *ViChaR* peut obtenir une bonne performance, l'arbitrage des VCs et sa logique sont plus compliquées et l'augmentation arbitraire du nombre de VCs peut augmenter la latence. Ce problème à motivé le travail du [128], le travail présenté adopte une approche dynamique basée sur une table de VCs, avec un nombre fixe de VCs. Les résultats présentés montrent une réduction de 30% de l'ensemble de la consommation de l'énergie du réseau et une réduction de 41% dans la surface silicium où la moitié des tampons d'entrée dans le routeur sont supprimés.

Les auteurs du [129] présentent un mécanisme d'évitement de congestion par contrôle de flux dans un NoC avec une communication au meilleur effort (*Best-Effort*). La méthode proposée utilise la combinaison de mécanismes de contrôle locaux et globaux, c'est-à-dire que l'information de congestion au niveau du nœud local, où la congestion se produit, fournit des vitesses d'injection prévues pour les sources de trafic correspondant. Les résultats de simulation montrent que cette méthode peut obtenir à peu près le même débit pour différents scénarios de trafic (*HotSpot*, *Bit-Complement* et *All2One*). Cette méthode est d'au moins 76% plus rapide que la méthode contre-pression (*back-pressure*) standard, pour informer la source sur le problème de la congestion dans le réseau (*congestion-Aware Time*). La méthode montre également une amélioration de la latence d'au moins 35% (en fonction de scénarios de trafic) par rapport à un mécanisme de contre-pression standard.

Dans [130], un nouveau système de contrôle de flux qui admet le critère d'équité Max-Min pour toutes les sources a été proposé. Le contrôle de flux a été modélisé comme une solution algorithmique simple à un problème d'optimisation. L'algorithme peut être mis en œuvre par un contrôleur qui admet une communication surchargée et non-surchargée. Dans [131], une nouvelle technique de contrôle de flux réseau, canal virtuel expresse avec des robinets (*EVC-T: Express Virtual Channel with Taps*), est proposée. Cette technique permet de transmettre les paquets de diffusion et les paquets de données de manière efficace. Cette technique a été évaluée par simulation, en utilisant à la fois un trafic synthétique et des benchmarks. Les résultats obtenus montrent que cette technique réduit la latence moyenne du réseau de 24% avec une variation négligeable de consommation d'énergie pour les benchmarks synthétiques. Pour les applications parallèles, cette technique permet d'augmenter les instructions par cycle de 9% en moyenne avec une augmentation minimale de la consommation d'énergie.

Les auteurs de [132] ont proposé un mécanisme de contrôle de flux dynamique des canaux pour réaliser un partage global des ressources du canal, ce qui peut augmenter le débit et le taux d'utilisation du canal. Un NoC 8x8 2D-Mesh est mise en œuvre sur un simulateur à cycle précis. Les résultats de simulation montrent que le dispositif proposé permet de réduire la taille du tampon de 30% comparé avec le contrôle de flux de canal virtuel avec le même débit. En outre, le débit peut être amélioré par 20 par rapport à un contrôle de flux en mode trou de ver.

VI.1.3. Les approches au niveau application

Au niveau application, le mappage (*mapping*) et l'ordonnancement (*scheduling*) des tâches de l'application dans les plateformes NoC, tout en optimisant les mesures de coûts et de performance, constitue le principal point à prendre en considération [108]. Par exemple, dans [90], la performance de plusieurs heuristiques de mappage (mise en correspondance) en cours d'exécution avec des charges de travail dynamique a été étudiée. Basé sur des demandes de communications et sur la charge des liens du NoC, les tâches sont remappées à la volée. Les résultats présentés ont montré une réduction en cours d'exécution de la congestion du NoC. Dans [133], une approche décentralisée en utilisant des clusters virtuels qui sont construits au moment de l'exécution est proposée. Les clusters sont reliés à un sous-ensemble de nœuds du NoC et ils ont une taille variable qui peut être ajustée au moment de l'exécution. Chaque cluster a un composant qui est responsable du (re)mappage. Un algorithme heuristique décentralisé est proposé dans [134] pour permettre à chaque élément IP de migrer les tâches individuelles dans les IPs voisins, en se basant sur la charge de travail local, la taille des taches et les exigences de communication des tâches à faire migrer.

Les auteurs de [122] [135] ont étudié le mappage topologique des cœurs IPs sur les architectures NoC pour épargner la bande passante et l'énergie de communication. Un middleware orienté service en temps réel, appelé CARISMA, a été proposé dans [136]. Ce middleware forme une couche homogène au-dessus des nœuds interconnectés et permet de faire fonctionner le système et les services d'application. Dans cette approche, les tâches sont réparties de manière autonome aux nœuds les plus appropriés.

Dans [84] [137], le problème de mappage des cœurs IPs spécifique à l'application sur le réseau sur puce est présenté. Ce problème a été largement exploré et des approches différentes ont été proposées. Dans [84] et [135], deux approches heuristiques dérivées des deux techniques *Branch and Bound* et *Greedy* ont été présentées. Dans [138] [139], le problème de mappage a été modélisé en utilisant une approche génétique. Les auteurs de [137] étendent le problème de mappage aux cas de multiples utilisations d'applications pour les boitiers décodeurs de la réception de la télévision numérique (*set-top boxes*) et la conception de TV-SoC. Dans [140], une méthodologie pour le mappage et la personnalisation de la topologie du NoC appliquée à une étude de cas industriel a été présentée, STNoC par STMicroelectronics. En particulier, l'approche proposée tente de trouver la meilleure topologie de l'application spécifique pour la famille des topologies STNoC allant du Ring jusqu'au Spidergon.

VI.1.4. Synthèse

Cette section donne un aperçu de l'état de l'art sur les approches d'optimisation du NoC. Contrairement aux approches en phase de conception dans lesquelles tous les paramètres/protocoles sont optimisés/sélectionnés au préalable pour cibler des applications spécifiques, dans les approches en phase d'exécution, le processus d'adaptation fonctionne en permanence pour faire évoluer le système. Sur la base de ce panorama d'état de l'art, des critères de performance ont été définis dans le contexte du NoC. Ces critères peuvent fournir un schéma de base pour développer un NoC performant sur tous les niveaux ; application, communication et physique.

VI.2. Méthodes et outils d'évaluation de performance du NoC

Les architectures d'interconnexion sur puce adoptées pour les systèmes sur puce sont caractérisées par un compromis de qualité de service de la communication entre la latence, le débit, la charge des liens, la consommation d'énergie et les exigences de la surface de silicium. Les architectures NoC peuvent être évaluées soit par expérimentation avec les systèmes existants, soit par expérimentation avec des modèles de systèmes (modèles de simulation ou d'évaluation analytique), comme le montre la figure 2.2. Cependant, l'expérimentation des performances des NoCs avec des vrais systèmes sur FPGA est très coûteuse, voire impossible. Par conséquent, la modélisation est la solution la plus efficace pour l'évaluation et l'optimisation de l'architecture du NoC avant sa mise en œuvre sur la puce.

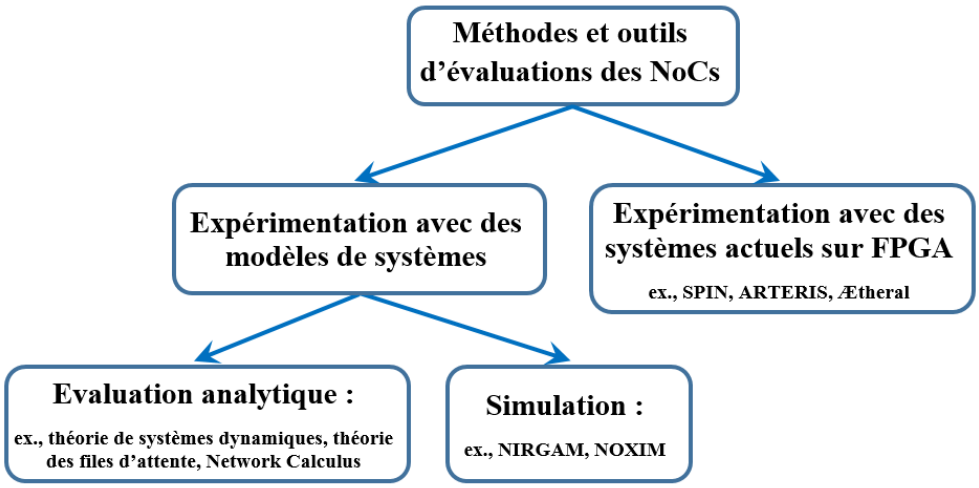


Figure 2.2 : Classification des méthodes et des outils d'évaluations des NoCs

VI.2.1. Evaluation des NoCs par des modèles de systèmes

VI.2.1.1. Evaluation analytique

Il existe plusieurs modèles analytiques d'évaluation de performance appropriés, ils peuvent estimer très tôt dans la phase de conception des indicateurs de performance souhaités en une fraction de temps que prendrait la simulation. Bien que l'utilisation des modèles de haut niveau recèle beaucoup d'aspects technologiques complexes, elle facilite l'exploration rapide de l'espace de conception NoC. En outre, les modèles analytiques fournissent non seulement des propriétés de synchronisation de système, mais également des informations utiles sur le comportement du système. Par conséquent, ils peuvent être invoqués dans une boucle d'optimisation des NoC pour une estimation rapide et précise de la performance [13]. Parmi les modèles les plus couramment utilisés dans la littérature, on peut citer :

- la théorie des files d'attente (*QT*: *Queuing Theory*),
- le Network Calculus (*NC: Network Calculus*),
- l'analyse d'ordonnançabilité (*SA*: *Schedulability Analysis*),
- l'analyse du flot de données (*DF*: *DataFlow Analysis*).
- La théorie des systèmes dynamiques (*DS*: *Dynamique Systems*)
- La modélisation par réseau à compartiments (*CFF*: *Compartimental Fluid-Flow*)

A. Théorie des files d'attente

La théorie des files d'attente est de grand intérêt pour les systèmes de calcul et les réseaux de transmission. C'est la méthode formelle qui considère certains composants du système (ex., processeurs, interconnexions, etc.) comme centres de services. C'est une théorie mathématique relevant du domaine des probabilités, qui est concernée par la modélisation mathématique et

l'analyse des systèmes qui fournissent des services aux demandes stochastiques. De plus, la théorie des files d'attente fournit l'ensemble d'approches et de formules pour résoudre analytiquement certains systèmes simple en obtenant leurs paramètres de performance.

Depuis que la théorie des files d'attente s'accorde avec des modèles de probabilité, elle est utilisée pour évaluer les indicateurs de performance moyen-cas de performance tels que la latence moyenne des paquets, le débit moyen, la consommation moyenne d'énergie et l'utilisation moyenne des ressources. Les concepteurs de systèmes utilisent ces indicateurs pour prendre des décisions de manières à résoudre certains problèmes tels que : la configuration de la mémoire tampon [107] et la capacité des liens [102].

Les auteurs de [141] ont développé un modèle basé sur la théorie des files d'attente pour quantifier le comportement de la performance et de l'énergie des réseaux sur puce, afin de fournir des estimations rapides de performance pendant la phase de conception. Ils ont posé comme hypothèse que, les arrivées de paquets à tous les canaux d'entrée suivent la propriété de Markov.

Dans [107], en résolvant une série d'équations non linéaires dérivées du modèle de files d'attente *M*/*M*/1/*K*, un modèle analytique est dérivé pour analyser rapidement la configuration actuelle de la taille de la mémoire tampon et de détecter les goulots d'étranglement dans les canaux du routeur. Ce modèle est ensuite utilisé pour résoudre le problème de dimensionnement de la mémoire tampon dans les NoCs à commutation de paquets. Plus précisément, compte tenu des caractéristiques de trafic de l'application cible et le budget total de l'espace de mémoire tampon disponible, le modèle proposé attribue automatiquement la profondeur de la mémoire tampon pour chaque canal d'entrée, dans les différents routeurs à travers la puce, de telle sorte que le temps d'attente moyen des paquets est réduit au minimum dans le système.

Dans [102], un modèle analytique de retard pour les réseaux à commutation en mode trou de ver (*wormhole switching*) équipé de canaux virtuels a été proposé pour l'allocation de la capacité des liens dans les systèmes SoCs. Ce modèle de files d'attente est basé sur *M*/*M*/1. Cet algorithme d'affectation alloue les ressources du réseau de manière efficace afin que la qualité de service (QoS) et les exigences de performance soient remplies.

Un modèle de performance analytique pour les NoCs à commutation en mode trou de ver (*wormhole switching*) a été proposé dans [142]. Avec l'utilisation du modèle de files d'attente *M*/*G*/1, le nombre moyen de paquets à chaque tampon est calculé. Ce modèle fournit trois indicateurs de performance, à savoir l'utilisation moyenne de la mémoire tampon, la latence moyenne d'un paquet et le débit du réseau.

Un autre modèle analytique pour estimer la performance de la communication du NoC à commutation en mode trou de ver (*wormhole switching*) est présenté dans [143]. Ce modèle prend en charge une topologie arbitraire de réseau avec des canaux virtuels. Pour résoudre la dépendance inhérente des liens successifs occupés par un paquet, les auteurs utilisent l'approche de routage à décomposition de chemin pour générer une série de catégories de liens ordonnés. Ensuite, ils utilisent les modèles de files d'attente *M*/*M*/1 et *M*/*M*/1/*K* pour calculer le temps de latence de transmission des composants du réseau.

Dans [144], les auteurs ont introduit une méthode d'analyse statique de synchronisation de niveau paquet pour les NoCs à commutation en mode trou de ver avec des canaux virtuels. Elle s'appuie sur une chaîne de Markov réduite pour représenter l'état du réseau, y compris l'occupation de tous les tampons. Le modèle gère n'importe quelle topologie, la capacité des liens et la taille de mémoire tampons, et il fournit une analyse de retard par flux.

Dans [145], un modèle analytique de performance en utilisant un processus semi-Markov est présenté pour estimer le temps moyen de latence des paquets dans le NoC. Plus précisément, un procédé semi-Markov est utilisé pour décrire le comportement de chaque lien dans le réseau et le retard du flit d'en-tête est calculé. Dans [146], les auteurs ont présenté un modèle analytique générique pour estimer les latences de communication et les utilisations lien-

tampons pour les NoCs à commutation en mode trou de ver avec une application donnée mappée sur elle. Ce travail modélise correctement les interdépendances qui en résultent entre les routeurs. En outre, une étude de cas d'utilisation d'une méthode d'analyse, basée sur des processus stochastiques de chaîne de Markov, pour l'évaluation de la latence d'un NoC 2D-Mesh, avec un algorithme de routage déterministe et un modèle de trafic *Uniforme*, est présentée dans [147]. Les auteurs de [148] présentent une analyse de la performance des NoCs hards et softs pour les FPGA. Ils ont appliqué le modèle de files d'attente de Jackson [149] pour analyser les performances d'un multiprocesseur SoC. Ensuite, ils ont utilisé le modèle de Jackson pour analyser des NoCs à commutation de circuits et les résultats obtenus montrent que les NoC câblés (hards) sont nettement meilleurs que les NoC softs classiques.

Pour analyser un système de files d'attente, il est nécessaire de connaitre un peu les lois régissant le processus d'arrivée, la logique qui régit le comportement de la file d'attente, et les caractéristiques du processus de service. La théorie des files d'attente concerne l'analyse mathématique de ces systèmes soumis à des exigences dont les occurrences et les longueurs sont spécifiées de façon probabiliste.

B. Network Calculus

Le *Network Calculus* (*NC*) est une théorie qui a été développée pour calculer des bornes déterministes (au pire-cas) sur les délais dans les réseaux de communication. C'est Cruz qui a écrit les articles fondateurs de cette théorie : il a traité d'abord des calculs de délais et de charge dans les éléments de réseaux pris isolément [150] ; puis de la mise en réseau des différents éléments [112]. De nombreuses questions concernant la modélisation du trafic, la topologie du réseau ou la gestion de la taille des paquets de données sont déjà soulevées, mais le formalisme du *NC* n'été pas encore présent.

Le but du *NC* est d'analyser les comportements critiques du réseau avec des méthodes permettant de calculer des bornes sur les délais, la charge ainsi que d'autres paramètres de qualité de service. En particulier, le *NC* s'intéresse aux performances pire-cas : soit des performances locales (taille maximale d'un tampon au niveau d'un nœud du réseau), soit des performances de bout-en-bout (temps de trajet de bout-en-bout pire-cas). Les informations sur le système sont modélisées grâce à des fonctions, telles que les courbes d'arrivée qui modélisent le trafic ou les courbes de service qui quantifient le service garanti en chaque nœud du réseau.

Parmi les applications du *NC* pour le NoC, on trouve l'analyse des SoCs [151]. C'est une alternative à d'autres approches d'étude de performance pire-cas comme : les méthodes holistiques [152], l'approche par trajectoire [153] ou la vérification de modèles (*models checking*) [154]. Il a des avantages a priori, en termes de modularité et de passage à l'échelle, qui permettent des études de réseaux complexes [155].

Dans [156], les auteurs ont examiné plusieurs disciplines de services qui sont proposés dans la littérature pour fournir des garanties de performance par des connexions de bout-en-bout dans les réseaux à commutation de paquets. Diverses questions et compromis dans la conception de disciplines de service pour la garantie de performance de service sont discutées, et un cadre général pour l'étude et la comparaison de ces disciplines est présenté. Ce travail donne un excellent aperçu de garantie de service dans les réseaux, qui est directement applicable aux NoC, et définit les bases qui ne sont pas différents en réseaux hors puce et en réseaux sur puce.

Le *NC* peut être utilisé pour estimer les pires-cas des délais et des retards de flux dans un système donné. Les bornes de retard au pire-cas par flux de flits et de paquets dans les NoC à commutation en mode trou de ver sont étudiées dans [157]. Les auteurs proposent d'abord des modèles d'analyse pour le contrôle de flux, le lien et le partage de la mémoire tampon, puis à partir de ces modèles d'analyse, ils obtiennent un modèle d'analyse ouverte de service pour capturer l'effet combiné de contrôle de flux, le lien et le partage de la mémoire tampon. Avec le modèle d'analyse de service, ils calculent les courbes de services équivalentes pour les flux individuels, puis ils dérivent leurs bornes de retard de flits et de paquets.

Dans [148], sur la base du *NC*, un système de régulation de débit est proposé pour réduire les délais et les bornes de retard dans les architectures SoC. Les auteurs ont ensuite formulé un problème d'optimisation pour minimiser les tampons totaux en vertu des exigences de qualité de service. L'analyse de la régulation est effectuée pour les réseaux à meilleur effort. Dans [158], une méthode fondée sur *NC* est présentée pour analyser et évaluer les interconnexions sur puce en termes de performance et des métriques de coût telles que les exigences de latence, de consommation d'énergie et de surface silicium. Les topologies 2D Mesh, Spidergon, et WK-récursives ont été comparées en utilisant un modèle de trafic donné. Les auteurs montrent que WK-récursive surpasse le 2D Mesh et le Spidergon dans tous les indicateurs considérés.

Le *NC* est appliqué pour analyser les bornes des délais et des retards de flux pour un trafic auto-similaire dans le NoC [159]. Les auteurs montrent tout d'abord que le trafic auto-similaire ne peut pas être limité par une quelconque courbe d'arrivée déterministe. Ensuite, ils prouvent que le trafic auto-similaire peut être limité par des courbes d'arrivée affines si un paramètre supplémentaire, l'excès de probabilité, est utilisé pour capturer son rafale qui dépasse l'enveloppe d'arrivée.

Dans [159], les auteurs dérivent les bornes de retards au pire-cas d'un flux individuel dans des NoCs à meilleur effort et à commutation de paquets. Les auteurs montrent que les retards simulés sont totalement contraints par les bornes de retard calculées et les bornes sont toutes exactes (*tight*). Dans ce travail, les auteurs ont pris des tampons suffisamment grands dans les routeurs et leur approche ne prend pas en charge les réseaux avec des tailles de mémoire tampon données et avec les tampons organisés comme des canaux virtuels. Plus tard, ils étendent leur travail pour des tampons bornés [160] et des canaux virtuels [161].

Le *NC*, qui est fondé sur l'utilisation d'enveloppes pour modéliser les contraintes du réseau et l'utilisation du dioïde (*min;+*) [162] pour combiner ces contraintes, a intrinsèquement certains avantages et inconvénients par rapport à d'autres méthodes d'évaluation de performances d'un réseau. Ces caractéristiques sont :

- un formalisme mathématique précis (l'utilisation de (*min;+*)) ; ce qui permet a priori, par rapport à des modèles basés sur l'interprétation au cas-par-cas des mouvements de données dans le réseau, d'éviter les erreurs d'interprétation dans des exemples pratiques.
- la possibilité d'utiliser des types d'enveloppes plus ou moins précises pour les contraintes. Par exemple, pour accélérer les calculs, on peut décider d'utiliser des courbes d'arrivée/service de type concave/convexe : certes les bornes seront moins précises, mais on est assuré d'obtenir des bornes qui sont toujours valides.
- la possibilité, en théorie, d'étudier des réseaux à n'importe quelle échelle. En effet, les notions de trajectoire entrée/sortie et de système ne font pas d'hypothèse sur l'objet étudié, et la combinaison de plusieurs sous-systèmes pour former un système est aussi tout à fait possible. Pratiquement, il est cependant difficile d'obtenir des bornes précises.
- le pessimisme des bornes, qu'il est difficile de contrôler. C'est justement l'objet des recherches sur le *NC*...

On pourra trouver des comparaisons plus précises sur les avantages et les inconvénients d'autres modèles [161] [162].

C. Analyse d'ordonnançabilité

L'analyse d'ordonnançabilité (*SA*: *Schedulability Analysis*) est un formalisme mathématique utilisé pour étudier les propriétés de synchronisation (s'assurer que toutes les échéances seront respectées) dans les systèmes temps réel. La *SA* a été proposé pour analyser les systèmes de calcul [163] [164] [165], puis elle a été appliquée aux plateformes de communication telles que les multi-ordinateurs [166] et les NoCs [167]. La *SA* est habituellement appliquée aux tâches de l'application en cours d'exécution sur un ou plusieurs Processeurs/Cœurs, mais c'est une plateforme qui permet d'évaluer les comportements aux pires-cas des systèmes.

Dans la *SA*, les tâches sont modélisées avec des modèles périodiques et sporadiques. Les tâches périodiques et sporadiques sont libérées à plusieurs reprises. Une tâche périodique est libérée à des intervalles réguliers et une tâche sporadique est libérée à des temps arbitraires mais avec un intervalle de temps minimum spécifié entre les versions [168] [169].

Les systèmes temps réel sont équipés d'un test d'ordonnançabilité [170], qui détermine si chacune des tâches admises peut respecter son échéance. Une nouvelle tâche ne sera pas admise que si elle passe le test d'ordonnançabilité. Le test d'ordonnançabilité peut être direct ou indirect. Dans un test d'ordonnançabilité direct le temps de réponse au pire-cas des tâches est calculé et un ensemble de tâches est planifiable si et seulement si le temps de réponse au pire-cas de chaque tâche est inférieur ou égal à son échéance. Ce type de test est précis, mais le coût de calcul des temps de réponse est très élevé. Dans [171], une formule itérative a été proposée pour calculer le temps de réponse au pire-cas d'un ensemble de tâches périodiques. La complexité de ce test est pseudo-polynomiale, donc il peut être inadapté pour le contrôle d'admission en ligne, en particulier dans les applications temps réel, comprenant de grands ensembles de tâches. Les auteurs de [172] ont proposé quelques méthodes pour réduire le nombre d'itérations dans le calcul des temps de réponse des tâches. Cependant, la complexité du pire-cas de leur test est encore pseudo-polynomiale.

La plateforme de communication SoC doit fournir différents niveaux de service pour les différents composants de l'application dans le même réseau. La communication en temps réel a des exigences très strictes, l'exactitude repose non seulement sur le résultat de la communication, mais aussi la limite de temps de réalisation. Un paquet de données reçu par une destination trop tard pourrait être inutile. La métrique de temps acceptable du pire-cas est définie comme étant l'échéance du paquet. Un ensemble de trafic en temps réel des flux sur le réseau sont appelés ordonnançable si tous les paquets appartenant à ces flux de trafic respectent leur échéances sous n'importe quel ordre d'arrivée de l'ensemble des paquets. Dans un tel système, l'analyse d'ordonnançabilité concerne les enquêtes d'ordonnancement des flux dans le réseau. Ce formalisme utilise une approche itérative pour estimer la latence maximale de bout-en-bout des flux dans un système en réseau.

Pour prendre en charge la communication temps réel dans des réseaux d'interconnexion, plusieurs mécanismes de contrôle de flux ont été proposés dans la littérature pour étudier le mécanisme d'ordonnancement de paquets en fonction de la priorité [166] [173] [174]. Dans [166], un mécanisme de régulation de débit est proposé, dans lequel il y a le même nombre de canaux virtuels que le nombre de niveaux de priorité, et un paquet peut demander seulement un canal virtuel qui est numéroté inférieure ou égale à sa priorité. Un autre mécanisme de contrôle de flux accéléré et anticipé [174] a été proposé pour éviter le problème d'inversion de priorité dû au blocage du contrôle de flux dans les routeurs à commutation en mode trou de ver. L'inversion de priorité est désignée comme une situation dans laquelle un paquet de priorité supérieure doit attendre que le transfert d'un paquet de priorité plus basse se termine. Le contrôle de flux accéléré et anticipé interdit les paquets à faible priorité d'utiliser les tampons d'entrée au-delà de leur limite autorisée, de sorte que les paquets de haute priorité peuvent toujours devancer les paquets de faible priorité pour utiliser les canaux, si nécessaire. Par conséquent, ce contrôle de flux ne provoque pas d'inversion de priorité. Cependant, la limite supérieure de la latence du réseau pour chaque paquet dans le réseau n'est pas délivrée par cette méthode.

Dans [175], les auteurs ont analysé la communication inter-processeurs des systèmes temps réel avec un réseau direct utilisant la commutation en mode différé (*store and forward*). Ils présentent une méthode pour garantir la durée maximale de livraison de bout-en-bout des paquets, et un test d'ordonnançabilité pour assurer que les paquets temps réel respectent leurs échéances. Plusieurs méthodes de contrôle de flux pour les réseaux temps réel acheminés en commutation en mode trou de ver ont été proposées dans [176]. Ces méthodes augmentent la probabilité de paquets temps réel pour respecter leurs échéances, mais ils n'ont pas de garanties sur la faisabilité de paquets. C'est pour cette raison qu'ils sont plus adaptés aux systèmes soft-

échéance qu'aux systèmes hard-échéance. Un test de faisabilité *off-line* pour les paquets temps réel acheminés en commutation en mode trou de ver est présenté dans [177]. Ce test fonctionne pour n'importe quelle méthode d'affectation statique de priorité. Passer ce test de faisabilité est suffisant, mais ce n'est pas une condition nécessaire pour la faisabilité. Tous les liens utilisés pour former un itinéraire pour le flux sont regroupés comme une ressource partagée (comme une structure de bus). Du coup, ils considèrent juste la concurrence directe et ils ignorent la concurrence indirecte, leur résultat est optimiste. Dans [173], les auteurs utilisent le même modèle proposé dans [177] et ils considèrent les concurrences indirectes. Cependant, les concurrences directes et indirectes sont considérées comme les mêmes, leur résultat est pessimiste. Ils montrent que la complexité de calcul de l'algorithme d'analyse d'ordonnancement proposé est en $O(n^2)$, où n est le nombre de flux dans le système.

Dans [178], un graphe de dépendance de blocage est utilisé pour exprimer les concurrences qu'un flux peut rencontrer et dérivé la borne supérieure de livraison de paquets. Les auteurs dans [179] ont formulé un arbre de concurrences pour prendre en compte les concurrences directes et indirectes et pour capturer l'utilisation simultanée des liens. Dans [167], les auteurs proposent une approche d'analyse d'ordonnançabilité hors ligne pour discuter en temps réel les services de communication sur puce avec une politique de commutation en mode trou de ver à base de priorité. Les auteurs montrent que le problème général de la détermination de l'ordonnancement exact en temps réel des flux de trafic sur le réseau sur puce est un problème *NP*-difficile. Cependant, ils donnent un déterminant de borne supérieure dans l'ordonnancement de flux de trafic en temps réel en évaluant diverses interrelations entre les flux de trafic. Ils proposent une méthode pour prédire la latence de paquets du réseau sur la base de concurrence directe et indirecte des flux de trafic de priorité plus élevée.

L'objectif de l'analyse d'ordonnançabilité est les systèmes temps réel, elle détermine que le système temps réel peut respecter l'échéance ou non. En outre, l'analyse d'ordonnançabilité essaie de donner la priorité à des tâches de sorte que chaque tâche respecte son échéance.

D. Analyse de flot de données

L'analyse de flot de données réfère à un ensemble de techniques qui infèrent des informations sur le flot de données le long des chemins d'exécution d'un système logiciel [180]. L'exécution d'un système logiciel peut être vue comme une série de transformations de l'état du système (constitué à partir de l'ensemble des valeurs de toutes les variables du système). Chaque exécution d'une instruction intermédiaire transforme un état d'entrée à un nouvel état de sortie.

Le graphe de flot de données est un modèle de calcul, où un certain nombre de processus concurrents communiquent entre eux par des canaux PAPS (Premier Arrivé, Premier Servi) non bornés [181]. Un programme de flot de données est un graphe orienté comprenant des nœuds (acteurs) qui représentent la communication et des arcs représentant les séquences ordonnées (flux) d'unités de données (jetons) comme illustré sur la figure 2.3 [13]. Les cercles représentent les nœuds, les flèches représentent les flux et le point représente un jeton. L'exécution d'un graphe de flux de données est une séquence de tirs. Lors de chaque tir un acteur consomme des jetons d'entrée et produit des jetons de sortie. Le nombre de jetons consommés et produits peut varier pour chaque tir et il est défini dans les règles de tir d'un acteur de flux de données. Une propriété importante des graphes de flot de données est que l'acteur de tir ne dépend que de la disponibilité des données. Cela implique que les graphes de flot de données ne sont pas temporisés (synchronisés), ce qui signifie que rien n'est précisé sur les points dans le temps au cours de lequel se produisent des tirs. Lorsque vous exécutez plusieurs acteurs sur une seule ressource, une séquence de tirs, aussi appelé un planning, est nécessaire. Pour les modèles généraux de flot de données, il ne peut pas être décidé si un tel planning existe parce qu'il dépend des données d'entrée [13] [106].

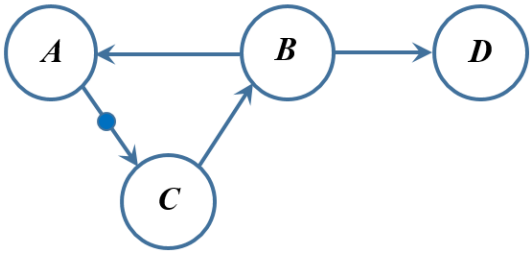


Figure 2.3 : Le graph de flot de données

Selon la façon dont la consommation, la production et les règles de tirs sont spécifiées, il existe une variété de différents modèles de calcul de flot de données. Ils diffèrent par leur expressivité, leur concision, leur analysabilité et leur efficacité de mise en œuvre [182]. L'expressivité et la concision d'un modèle de calcul indique comment il peut expliquer les caractéristiques du système et comment il est compacté. L'analysabilité est déterminée par la disponibilité des algorithmes d'analyse et de synthèse et de l'environnement d'exécution nécessaire à un algorithme sur un graphe avec un nombre donné de nœuds. L'efficacité de mise en œuvre d'un modèle de calcul est influencée par la complexité du problème d'ordonnancement de l'exécution.

Il existe plusieurs types de flot de données. Parmi les modèles les plus importants, on peut citer les graphes synchrones de flot de données (*SDF*: *Synchronous DataFlow*) [183] [184] [185], ils constituent un formalisme issu de la conception de systèmes électroniques. Ils ont été introduits par Lee et Messerschmitt en 1988 et ont été utilisés intensivement dans le monde académique et industriel, donnant naissance à de nombreuses extensions. Etant donné que dans *SDF*, les débits de données sur différentes canaux ne sont pas les mêmes, il est aussi appelé un modèle de flot de données multi-débit [186]. Le graphe *SDF* à taux unique ou homogène (*HSDF*: *Homogeneous Synchronous DataFlow*) [183] [187] [188] est une forme restreinte du modèle *SDF* dans lequel la consommation et la production sur chaque arc est d'un seul jeton. Un jeton est franchissable s'il existe au moins un jeton sur l'ensemble de ses arcs entrants. Pour toute *HSDF*, un planning statique peut être facilement construit par les outils de planification de compilation. Un algorithme qui transforme tout graphe *SDF* en un graphe *HSDF* est décrit dans [189]. Il y a aussi le modèle de flot de données cyclo-statique (*CSDF*: *Cyclo-Static DataFlow*) [190], dans lequel le nombre de jetons consommés et produits par un acteur varie de manière cyclique. Il y a un nombre fixe de phases dans un cycle et chaque acteur produit ou consomme un nombre fixe de jeton dans chaque phase, mais différentes phases peuvent avoir un comportement différent. Bien que le *CSDF* soit plus compact que le *HSDF*, il est aussi expressif que *HSDF* [191]. Comme le taux de chaque canal de données n'est pas fixe, l'analyse et la planification des *CSDF* sont plus complexes par rapport aux *SDF*. Tout de même, en raison de certains comportements asynchrone et dépendant des données, quelques applications de streaming ne peuvent être exprimées par *SDF* et *CSDF* [192]. Ce problème peut être résolu en étendant le modèle *SDF*. Une autre généralisation du modèle de flot de données est le Booléenne *SDF* (*BDF*: *Boolean DataFlow*) [193] où le nombre de jetons consommés et produits dépend de la valeur d'un jeton lit depuis une entrée de contrôle dédié. Puisque les productions et les consommations de jetons dépendent des valeurs de données lors de l'exécution, un réseau *BDF* n'est pas complètement statiquement planifiable. Le modèle de flot de données dynamique (*DDF*: *Dynamic DataFlow*) [181] est un modèle de flux de données booléen avec une variation supplémentaire ; les acteurs du contrôle mentionnés dans le modèle *BDF* sont capables de lire les valeurs de multiples jetons et les acteurs de données peuvent être tirés conditionnellement à base du contrôle des acteurs de lecture. En raison de la connaissance incomplète au moment de la compilation, les modèles de calcul *BDF* et *DDF* ont besoin d'un mécanisme de planification d'exécution pour déterminer quand un acteur devient exécutable. En outre, il n'est pas toujours possible de prédire si un planning avec des longueurs de tampon bornées peut être construit. Par conséquent, le planning d'exécution et le mécanisme de détection de blocage sont nécessaires pour implémenter ces modèles de calcul. Cela rend leur

mise en œuvre moins efficace par rapport aux *SDF* et *CSDF*. Pour surmonter ce problème, plusieurs modèles de calcul de flux de données ont été proposés dans la littérature tels que le flot de données synchrone paramétré (*PSDF*: *Parameterized Synchronous DataFlow*) [194], le flot de données à taux variable (*VRDF*: *Variable Rate DataFlow*) [195], le flot de données variable par phase (*VPDF*: *Variable Phased DataFlow*) [196] et le flot de données à scénario conscient (*SADF*: *Scenario-Aware Dataflow*) [197]. Ces modèles de calcul offrent un compromis entre l'analysabilité et l'efficacité de mise en œuvre. Par conséquent, ils peuvent exprimer un certain dynamisme tout en permettant l'analyse au moment de la conception et un faible débordement de la mise en œuvre.

Les modèles de flot de données classiques ne sont pas temporisés/synchronisés. Pour traiter les propriétés temporelles d'un système, un temps d'exécution au pire-cas peut être associé à chaque acteur [198]. Cette extension nous permet d'évaluer le comportement temporel du système à base de NoC comme le débit et la latence. Un temps d'exécution au pire-cas est ajouté à chaque acteur comme le montre la figure 2.4 [185]. Un nombre spécifié de jetons est consommé et produit au même temps de l'exécution de l'acteur. Une boucle d'arc d'un acteur est utilisée pour modéliser que l'exécution précédente doit être terminée pour que la prochaine exécution peut commencer. Les politiques d'ordonnancement peuvent être modélisées indirectement en transformant le temps d'exécution au pire-cas pour le temps de réponse au pire-cas [185].

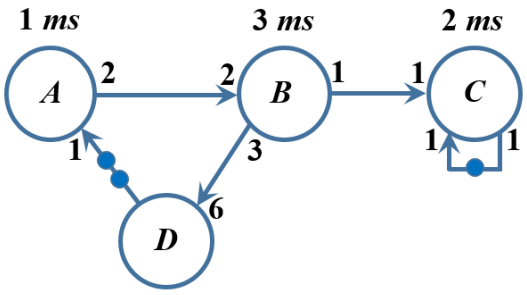


Figure 2.4 : Un exemple de graphique *SDF* avec le temps d'exécution

Le débit est un indicateur de performance important pour les applications de streaming. Il a été bien étudié dans la littérature sur les modèles de flot de données [199] [200] [201] [202]. Toutes ces études ont porté sur l'analyse de *HSDFs* et sont applicables aux *SDFs* que par une conversion à *HSDF* [183] [198]. La latence est un autre indicateur de performance très important. Cependant, peu de recherches ont été faites sur la latence. Les auteurs de [198] étudient la latence pour les *HSDFs*. Bien qu'il soit possible de calculer le temps de latence pour *SDF* par une conversion à *HSDF*, la conversion peut conduire à une augmentation exponentielle du nombre de nœuds dans le graphe qui rend prohibitif de prédire des mesures de performance [182]. Dans [203], une technique de minimisation de la latence est présentée qui travaille directement sur *SDF*. Cette technique calcule la latence minimale possible pour un *SDF* et fournit un schéma d'exécution qui donne le minimum de latence.

Dans [204] une architecture à base de multiprocesseur NoC et un modèle *HSDF* des jobs sont présentés, ils permettent de raisonner sur le comportement temporel du système dans lequel le NoC fournit des connexions virtuelles point-à-point avec un débit garanti et une latence maximale. Les auteurs ont modélisé chaque tâche par un acteur avec un arc qui boucle sur le nœud lui-même. Dans [205], les auteurs ont montré que les serveurs à taux de latence [206] peuvent être inclus dans un modèle de flot de données par deux acteurs. L'un des acteurs modélise le taux, et l'autre modélise la latence. Dans [185], les modèles *SDF* sont utilisés pour calculer le comportement temporel de bout-en-bout des jobs dans un système temps réel de multiprocesseurs embarqués. Dans [207] et [208], les auteurs ont montré comment construire un modèle *CSDF* qui modélise de façon conservatrice les connexions d'un NoC. Ensuite, ils ont utilisé le modèle de flot de données proposé pour le dimensionnement de la taille de la mémoire tampon dans les interfaces du réseau pour garantir la performance du système, et ils

ont montré que la taille des tampons est déterminée avec un temps d'exécution comparable aux méthodes analytiques, et des résultats comparables à la simulation exhaustive.

Les auteurs de [209] ont proposé un algorithme qui détermine de façon approximative les capacités tampons minimales pour les graphes *CSDF* tels que les exigences de débit et les contraintes sur les capacités tampons maximales sont satisfaites. En outre, ils ont montré que le modèle *CSDF* peut conduire à des besoins en ressources réduites par rapport à un modèle de *SDF*.

La Figure 2.5 montre la comparaison des différents modèles de calcul de flot de données examinés sur la base des aspects de l'expressivité et de la concision, de l'analysabilité et de l'efficacité de mise en œuvre [182]. Les modèles de flot de données sont classés en fonction de leur capacité à capturer le comportement dynamique d'une manière compacte dans l'expressivité et l'axe de la concision.

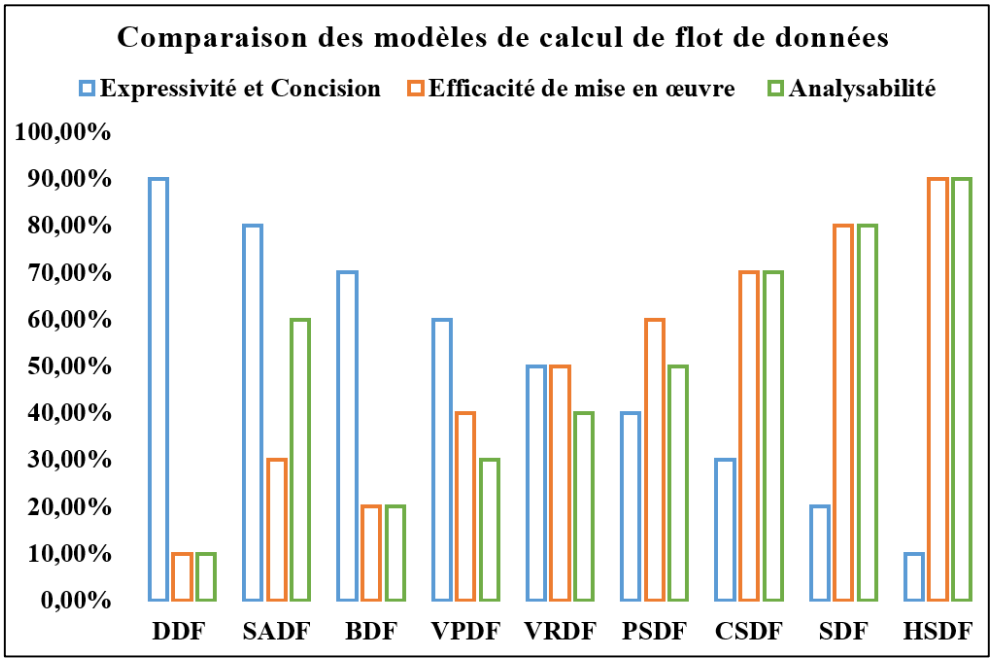


Figure 2.5 : Comparaison des différents modèles de calcul de flot de données (Adapté de [182]).

Une conclusion générale est que l'expressivité est typiquement troquée contre l'analysabilité et l'efficacité de mise en œuvre.

E. Théorie des systèmes dynamiques

Issue de la physique, la théorie des systèmes dynamiques, encore appelée "dynamique non-linéaire" ou "théorie des systèmes complexes" traite de grands types de comportements ou d'évolutions, sans référence directe aux éléments matériels qui constituent ces systèmes, ce qui lui permet de présenter un haut degré de généralité et d'universalité [210]. L'objet de la théorie des systèmes dynamiques est l'étude de la formation des patterns et des structures, dans les systèmes complexes. Il s'agit d'une méta-théorie, proposant un cadre formel d'analyse, indépendant des substrats ou niveaux d'analyse sur lesquels elle peut être appliquée. La théorie de systèmes dynamiques cherche à expliquer le comportement d'un système complexe, dynamique, social, technique, économique et politique (systèmes sociaux, technologiques, économiques et politiques) pour améliorer la prise de décision. Ses racines remontent à environ 35 ans en arrière, à la dynamique industrielle où Forrester [211] a expliqué les problèmes qui découlent des applications industrielles, telles que l'instabilité de production et du travail, l'instabilité du développement des affaires et du partage des marchés. Cette théorie est développée aussi dans d'autres domaines d'intérêt, comme la gestion de projets de recherche et de développement, le développement urbain, la gestion des ressources énergétiques, l'astrophysique, la physique des particules, l'étude des turbulences hydrauliques, l'étude des

champs magnétiques, la météorologie, le développement et le contrôle et l'apprentissage moteur, l'économie, la sociologie, la psychopathologie, la gestion, la prospective et la théorie du chaos. Le nom "dynamique industrielle" est vite devenu la théorie de systèmes dynamiques.

La théorie de systèmes dynamiques exprime les interactions entre des variables d'un système et prédit leur influence dans une certaine période de temps [212]. L'application de la théorie de systèmes dynamiques est un outil moderne dans la procédure de prise de décision [213]. Dans le domaine de la gestion de la qualité totale (*TQM*: *Total Quality Management*) l'application de la théorie de systèmes dynamiques a été largement étudiée dans la littérature. Le *TQM* est décrit comme un modèle contemplatif et n'a donc pas un comportement prévisible. Par conséquent, l'application des théories de systèmes dynamiques est imposée, comme il est décrit dans la littérature [211] [214] [215] [216]. Selon ce dernier, un système dynamique est défini en fonction du temps, de la culture et du comportement répété. La théorie de systèmes dynamiques a une approche "holistique", plutôt qu'une approche d'entrée-sortie. Selon cette approche, les changements qui se produisent dans un espace ou dans un sous-ensemble du système influent sur le sous-système lui-même ainsi que le reste des sous-ensembles.

La théorie de système dynamique ne s'intéresse pas aux systèmes, elle s'intéresse plutôt aux problèmes [212]. Les problèmes dans les systèmes dynamiques ont au moins deux choses en commun : premièrement, ils sont dynamiques, ce qui signifie qu'ils contiennent des variables qui changent au fil du temps, par des comportements répétés périodiquement et par des changements complexes. Le facteur "temps" comprend le développement à long terme, les changements actuels et les orientations futures prévisibles. Le facteur "comportements répétés" comprend les comportements non-linéaires, soit avec une influence positive ou négative. Le facteur "changements complexes" va au-delà du concept de causalité (cause à effet) et comprend des phénomènes qui ne suivent pas l'évolution prévisible dans le temps. Par exemple, le taux de chômage local, les augmentations d'impôts et la gestion de la qualité de vie pourrait retarder la construction d'un bâtiment, le développement d'une économie, etc. La définition correcte du problème est la première étape dans la théorie de systèmes dynamiques [217]. Deuxièmement, les problèmes comprennent la notion de rétroaction, comme les systèmes de servomécanique dans les moteurs (un servomoteur est capable de maintenir une opposition à un effort statique et dont la position est vérifiée en continu et corrigée en fonction de la mesure) et les systèmes humains [213]. La théorie de systèmes dynamiques met l'accent sur la structure et le comportement des rétroactions interconnectés.

F. Modélisation par réseau à compartiments

Une alternative aux méthodes de modélisation de systèmes dynamiques mentionnées précédemment, est la modélisation par des systèmes à compartiments. Les réseaux à compartiments [218] ont été utilisés dans de nombreux domaines, comme les réseaux de communications [219] [220], les procédés industriels et économiques [221] [222], les systèmes de santé [223], etc. Le principal intérêt de ces réseaux consiste en l'élaboration d'une représentation dynamique du comportement du système à modéliser. En nous appuyant sur l'approche des réseaux à compartiments, nous présentons dans ce paragraphe une méthode pour la mise en équation d'un modèle dynamique pour les réseaux NoCs. Dans un régime de fonctionnement dynamique les paramètres sont variables dans le temps.

Les réseaux à compartiments sont des réseaux constitués de réservoirs de stockage (réels ou virtuels) qui contiennent une certaine quantité d'une espèce matérielle ou immatérielle déterminée (masse, énergie, information, etc.), dénommés "compartiments", qui sont interconnectés par des mécanismes de transfert (flux de matière, d'énergie ou d'information). Des modèles dynamiques pour décrire le trafic dans le NoC peuvent être proposés en adaptant des modèles de la mécanique des fluides et en s'appuyant sur ces modèles, de concevoir et mettre en œuvre un algorithme de contrôle de flux. Un exemple d'une représentation d'un réseau NoC dynamique peut être donné par la figure 2.6.

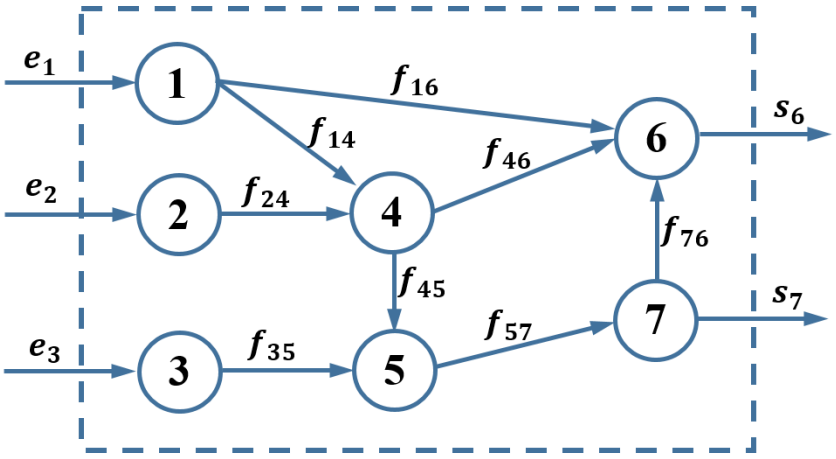


Figure 2.6 : Représentation par réseau conservatif

Les nœuds du graphe, qui représentent les commutateurs du NoC, contiennent chacun un réservoir de quantité variable appelé "compartiment". Ce réservoir représente l'état du nœud, dans notre cas, c'est le nombre de flits dans la file d'attente correspondant au nœud. Les compartiments sont interconnectés par les arcs. Chaque arc constitue un lien entre deux nœuds différents. Les flux de trafic envoyés par les nœuds traversent les arcs (liens) pour aller des sources jusqu'aux destinations des flits. Le flux de transfert entre deux nœuds i et j est noté par f_{ij}. Les interactions du réseau avec son milieu extérieur (entrées/sorties) sont notées par : λ_i envoyées depuis les nœuds sources, et e_i envoyées vers les nœuds destinations. La variable d'état $x_i(t)$ représente la quantité actuelle contenue dans chaque compartiment i du réseau NoC au temps t. Le vecteur $x(t) = \big(x_1(t), x_2(t), \dots, x_n(t)\big)$ est le vecteur d'état du système, son ordre est équivalent à tous les nœuds du réseau NoC. Les flux de transfert $f_{ij}\big(x(t)\big)$ entre les nœuds i et j, et les flux de sortie vers les destinations $e_i\big(x(t)\big)$, sont des fonctions des variables d'état. Ainsi, la dynamique de chaque compartiment est décrite par l'équation différentielle suivante.

$$\dot{x}_i = \big(\sum_{k\neq i}^{q} \lambda_{ki} + \sum_{j\neq i}^{n} \alpha_{ji}(x)\big) - \big(\sum_{k\neq i}^{p} \mathrm{e}_{ik}(x) + \sum_{l\neq i}^{m} \alpha_{il}(x)\big) \qquad (2.1)$$

L'évolution de l'état de chaque compartiment est décrite par la différence entre la somme des flux d'entrée (parenthèse de gauche de l'équation 2.1) et celle des flux de sortie (parenthèse de droite de l'équation 2.1).

Nous rappelons ci-dessous certaines propriétés structurelles de ces réseaux [219] [224] :

- Les réseaux à compartiments sont des systèmes positifs et conservatifs,
- Un système régi par une loi de conservation de masse s'appelle conservatif,
- Pour chaque compartiment d'un système à compartiments les débits qui interviennent sont des flux non négatifs, donc le système est positif,
- Les flux sortant de ou entrant dans un compartiment ont tous des valeurs positives, mais ils sont déduits ou additionnés quand la balance de flux est écrite pour un tel compartiment. En plus, un réseau à compartiments est conservatif dans le sens que la quantité totale contenue dans le système est conservée.

Pour l'utilisation des modèles dynamiques pour la conception d'un algorithme de contrôle de flux il faut en déduire une représentation par des modèles d'entrée/sortie (équations différentielles ou fonctions de transfert) ou par des équations d'état.

G. Synthèse sur les modèles d'évaluation analytique

Le compromis général entre l'abstraction et la précision est peut être observé dans la comparaison entre ces quatre formalismes. Comme chacun des formalismes examinés a des avantages et des inconvénients comme le montre le tableau 2.2, et ils sont également

partiellement différents dans le principe, aucun d'entre eux ne peut facilement remplacer tous les autres. Il y a certainement des problèmes de point pour chaque formalisme qui nécessitent de les étudier encore, mais la recherche sur des approches pour les problèmes de l'analyse de la performance du système est la plus urgente.

La théorie des files d'attente, le Network Calculus, et l'analyse d'ordonnançabilité peuvent trouver une relation de forme proche entre les paramètres du système et les paramètres de performance du système. Toutefois, en cas de la théorie des files d'attente et du Network Calculus, il est difficile d'en tirer des modèles mathématiques d'un réseau donné, parce qu'ils utilisent des modèles à événements complexes (et aussi les modèles nœuds). D'autre part, l'analyse d'ordonnancement utilise des modèles à événements simples. Par conséquent, le modèle de performance est facilement extrait avec moins de précision. Un problème commun des formalismes mentionnés ci-dessus est qu'ils ne peuvent pas modéliser un système dans lequel il existe des dépendances entre les flux de données. Parmi ces quatre formalismes mathématiques étudiés, le modèle de flot de données est le seul formalisme capable de modéliser les dépendances de flux dans un système avec précision. Cependant, nous devons utiliser des modèles de flot de données restreintes, comme *SDF* et *CSDF*, qui ne peuvent pas modéliser tous les systèmes [13].

Formalismes	Avantages	Inconvénients
La théorie des files d'attente	• formes d'équations proches	• difficile de dériver des modèles précis • ne peut pas représenter les dépendances de flux
Le Network Calculus	• formes d'équations proches	• difficile de dériver des modèles précis • ne peut pas représenter les dépendances de flux
L'analyse d'ordonnançabilité	• formes d'équations proches • facile à configuré les nœuds et à modéliser les évènements	• ne peut pas représenter les dépendances de flux • précision limité
L'analyse de flot de données	• modélisation naturelle • peut exprimer les dépendances de flux et le contrôle de flux	• utilisable qu'avec des modèles restreints, comme *SDF* et *CSDF*
La théorie des systèmes dynamiques	• Synthétique et abstraite • modélisation temps réel • peut exprimer les dépendances de flux et le contrôle de flux • déterministe	• système causal (dépend que de phénomènes du passé/présent) • difficile à décrire mathématiquement • non-autonome
Modélisation par réseau à compartiments	• Synthétique • modélisation temps réel par équations différentielles • peut exprimer les dépendances de flux et le contrôle de flux	• traite que les moyens-cas • difficiles à modéliser des réseaux de grande taille • non-autonome • haut niveau d'abstraction

Tableau 2.2 : Avantages et inconvénients des quatre formalismes

Bien que chaque formalisme peut être étendu dans diverses directions, ces extensions fonctionnent généralement sur des problèmes mathématiques complexes ou ils sont perçus comme étant contre nature et encombrant. Par conséquent, nous pensons qu'un modèle complet qui combine deux ou plusieurs formalismes seraient très souhaitable.

VI.2.1.2. Evaluation par simulation

A. Classification des simulateurs NoC

Cette section a pour but d'analyser les différentes solutions de simulation de NoC existantes afin de choisir l'une d'entre elles qui offre un bon rapport de simplicité, rapidité, possibilité de personnalisation de l'architecture NoC (topologie, espace tampon, etc.), possibilité d'évaluation des performances (latence et consommation d'énergie) du NoC et prise en charge des topologies, des algorithmes de routage et des applications les plus couramment utilisés pour le NoC. Ces contraintes sont considérées pour le choix du simulateur utilisé dans la suite pour la simulation de nos approches pour le NoC.

Il existe de nombreux simulateurs disponibles dans la littérature qui peuvent être utilisés pour simuler un réseau MSoC/NoC. Il est possible de les classifier en trois catégories. Les simulateurs les plus répondus seront présentés plus en détail ci-dessous (section VI.2.1.2.B).

a. Simulateurs de systèmes

Ils proposent des plateformes de simulation MPSoC mais ils se focalisent sur un point particulier de l'architecture (processeur, tolérance aux fautes...) et ne permettent pas d'explorer tous les paramètres de la plateforme. En général, ces simulateurs implémentent une solution unique et simple du système d'interconnexions ce qui restreint l'exploration.

SimOS (Simulator Operating System) [225] est un vrai simulateur complet de système. Il simule le code utilisateur et le code système. Initialement développé par le groupe de Stanford flash pour le système MIPS/IRIX. Maintenant, il existe d'autres versions comme le système Alpha/OSF et le système PPC/AIX. Le simulateur original a été intégré avec succès avec Sicosys [226] [227] pour exécuter des charges de travail de base de données.

Simics [228] est un système à niveau, simulateur de jeu d'instructions, capable de simuler des systèmes cibles haut de gamme avec une fidélité et une vitesse suffisante pour démarrer et exécuter des systèmes d'exploitation et des charges commerciales. Il est commercial mais un peu cher (à partir de 5000 € à 18000 € l'année).

Les simulateurs Simics [228], SimOS [225] proposent une modélisation simplifiée de l'architecture afin d'être assez rapide pour simuler des systèmes sur différentes architectures. Dans les deux cas, leur vocation n'est pas de permettre l'exploration architecturale mais de permettre une mise en situation du système.

RSIM [229], MicroLib [230], SESC [231], UNISIM [232], CASSE [233], ROSES [234], CAAM [235] ou encore GEMS [236] sont des simulateurs qui permettent l'exécution d'applications sur une structure MPSoC mais pas l'exploration de l'ensemble de la structure.

RSIM [229] est un environnement de simulation multiprocesseur à mémoire partagée (et monoprocesseur). Ce simulateur est destiné à exploiter les processeurs offrant un parallélisme par niveau d'instructions (*ILP: Instruction-Level Parallelism*). Ainsi, il offre la possibilité de modifier de nombreux paramètres au niveau du processeur. Une plateforme multiprocesseur est également disponible, elle propose une structure multiprocesseur homogène contenant une cache cohérente qui s'organise autour d'un réseau 2D Mesh. Cette plateforme n'utilise pas d'interface standard ce qui ne permet pas une grande modularité. De plus, seule une topologie, dont le temps de traversée est fixe, est disponible, et il ne prend pas en charge l'OS. Par contre il est facile à intégrer avec le simulateur de réseau d'interconnexion Sicosys [226] [227] et de l'employer dans des applications d'évaluations du réseau NoC.

UNISIM [232] remédie au problème d'interface. En effet, il est basé sur la librairie SystemC [237] et utilise l'extension TLM (*TLM*: *Transaction Level Modeling*) associée [238]. Il est donc facilement modulable du fait de l'utilisation de la librairie standard TLM. En fait, étant donné que tous les modules utilisent la même interface, il devient simple de rajouter, retirer et

combiner des modules dans le simulateur. Mais comme pour RSIM, une seule interconnexion est proposée, il s'agit d'un PCI-Bus.

D'autres simulateurs introduisent de nouvelles fonctionnalités, par exemple, ReSP [239] utilise la même base de librairie que le simulateur UNISIM mais il propose en plus d'encapsuler le noyau de simulation SystemC dans un environnement de contrôle écrit en Python. Ceci a pour but d'augmenter les services en permettant un contrôle non intrusif. Par exemple, ReSP permet d'arrêter, de mettre en pause et de redémarrer une simulation à tout moment, ce qui n'est pas directement possible avec le noyau SystemC. De plus à la compilation, les modules SystemC que l'on veut contrôler sont encapsulés dans une couche d'adaptation C++/Python. Cette surcouche permet, par exemple, d'introduire des fautes dans les communications et ainsi de tester la tolérance aux fautes d'un module sans devoir modifier le code SystemC.

D'après les auteurs de [239], l'impact n'est pas significatif comparé à une simulation seule. Cette surcouche n'est pas nécessaire pour l'exploration et surtout elle aura sûrement un impact si l'on simule un système beaucoup plus gros. Cette catégorie de simulateur permet principalement l'étude d'un processeur élémentaire dans un environnement multiprocesseur, la vérification du portage de l'application et le début du dimensionnement architectural (nombre de processeurs, hiérarchie mémoire). Mais le manque de précision dans le modèle d'interconnexions ne permet pas l'exploration de celui-ci.

b. Simulateurs de réseaux d'interconnexion

Cette catégorie de simulateurs autorise l'exploration du réseau d'interconnexions. En effet, ces simulateurs sont, par nature, conçus pour explorer les réseaux selon un maximum de paramètres. On peut citer Atlas [240], μSpider [28] [241], NoCGen [242], Noxim [243], Sicosys [226] [227], Nostrum [244] [245], Nirgam [77], gpNoCSIM [246], Garnet [247], NNSE [248], BookSim [249], Worm_sim [250], HNoCS [251], Chaos [252], etc.

Chaos [252] est un simulateur de routeur. Il simule un modèle détaillé de réseau de type *k-ary n-cube*. Il s'agit notamment des topologies courantes comme les réseaux 2D Mesh et 2D Torus et les réseaux hyper-cube. Il support les réseaux à commutation de paquets et à commutation de trou de ver (*wormhole switching*), et achemine les paquets en utilisant soit le routage adaptatif chaotique ou bien le routage dimensionnel (XY, XYZ, e-cube) en utilisant une variété de modèles de trafic. Par contre, il est assez vieux et pas très commode.

Noxim [243] est un simulateur SystemC synchronisé sur une horloge qui décrit précisément le routeur d'une topologie 2D mesh. Il suffit donc de relier ces routeurs entre eux afin de former une grille (un mesh) de taille variable.

De même, le simulateur de réseau NS2 [253] [254] (Network Simulator 2) peut également être utilisé pour la simulation de NoC basée sur le routage distribué. Comme Noxim, il peut également générer des trafics et des résultats automatiquement.

Il est possible de modifier un certain nombre de paramètres dans ces routeurs afin d'explorer les différentes possibilités de réseaux. Ainsi, l'algorithme de routage, la profondeur des FIFO et la politique d'arbitrage dans les routeurs peuvent être aisément remplacés. L'émission et la réception de messages dans le réseau sont effectuées par des générateurs de trafic paramétrables. Tous ces simulateurs sont synchronisés par une horloge afin de fournir une précision au cycle, sauf HNoCS qui est à base d'évènements. Dans tous les cas, il est nécessaire de définir le routeur de base et de connecter correctement l'ensemble de ces routeurs selon la topologie choisie. Mais, malheureusement, leur défaut principal est de ne pas proposer un modèle de processeur précis se connectant sur le réseau, ce qui ne permet pas de réaliser une exploration complète.

c. Simulateurs mixtes

Cette catégorie de simulateur permet une exploration complète d'une architecture multiprocesseur, du réseau d'interconnexions aux processeurs. Dans cette catégorie, on trouve deux approches afin de permettre une simulation totale.

La première approche consiste à coupler un simulateur de la première et de la deuxième catégorie. Ainsi on obtient un simulateur complet qui permet une simulation de toutes les parties du multiprocesseur. On peut citer l'association Sicosys+RSIM [226] ou encore GEMS+GARNET [247]. Dans les deux cas, le point délicat consiste à fournir une interface adaptant les protocoles de communication des deux simulateurs. En effet, dans le cas de Sicosys+RSIM, les deux simulateurs n'utilisent pas la même façon de se synchroniser. Sicosys est synchronisé à une horloge alors que RSIM se synchronise sur des évènements afin d'être plus rapide. Du fait de cette adaptation, la vitesse de simulation chute d'un facteur 3 [226]. On retrouve également une perte de performance dans le cas de GEMS+GARNET. Cette perte de performance peut en partie s'expliquer par l'ajout de communications entre deux processus différents sur puce sur le processeur hôte. En effet, les communications entre les processus sont plus consommatrices en temps que des lectures dans la pile du même processus. Le problème des assemblages comme ceux-ci réside dans le fait que les simulateurs ne sont pas conçus dès le départ comme un seul simulateur et que donc leurs interfaces ne sont pas adaptées.

La deuxième approche rectifie cette faiblesse en proposant des simulateurs complets destinés à l'exploration architecturale. Mentor Graphics [255]et Coware [256] proposent, chacun, une plate-forme complète mais payante permettant l'exploration architecturale. ASIM [257], MC-SIM [258], MPARM [259] et Soclib [260] sont des simulateurs MPSoC complets. Toutefois, ASIM, MC-SIM et MPARM ne proposent qu'une seule interconnexion chacun, ce qui limite les possibilités d'exploration. Pour sa part, SoClib, proposé par le Lip6, est un simulateur MPSoC complet. Ce simulateur basé sur la librairie SystemC est à l'origine au niveau CABA (*CABA*: *Cycle Accurate, Bit Accurate*) et à ensuite proposé un niveau TLM. Il dispose d'un large choix de processeurs, d'interconnexions et d'IPs diverses. Les connexions entre les modules se font à travers une interface VCI (*VCI*: *Virtual Component Interface*) afin de normaliser les connexions. L'écriture de chaque module est contrainte par des règles d'écriture qui permettent l'uniformisation des modules.

B. Simulateurs NoC existants

L'étude de l'existant de la section précédant nous permis de voir les différentes approches pour simuler une plate-forme MPSoC/NoC. Il apparait qu'au moment du choix aucune solution ne correspondait exactement à nos besoins de simulations de réseau d'interconnexion. Cette étude met en évidence que dans la majorité des cas, la modélisation du réseau d'interconnexions est réalisée en décrivant chaque élément de cette interconnexion. Ceci peut poser un problème si on envisage de simuler un réseau de grande taille. Par exemple en SystemC, il en résulte un grand nombre de threads SystemC ce qui risque de ralentir la simulation.

Pour cette raison nous avons voulais voir plus en détail les simulateurs de réseau d'interconnexion les plus proches de nos besoins de simulation, leur description est présentée dans cette section, afin de pouvoir choisir un simulateur qui convient le plus à la simulation de nos travaux sur le NoC.

a. BookSim

BookSim [249] est un simulateur à cycle précis de réseaux d'interconnexion. Initialement développé et introduit avec les principes et pratiques d'un livre de réseaux d'interconnexion, depuis sa première apparition sa fonctionnalité a été étendue en permanence. La version actuelle, BookSim 2.0, prend en charge un large éventail de topologies comme le 2D Mesh, le 2D Torus et les réseaux de papillon aplati [261] (flattened butterfly), fournit divers algorithmes de routage et inclut de nombreuses options de personnalisation de la microarchitecture du

routeur du réseau. Le code source de BookSim 2.0 est accessible via son navigateur référentiel sur le Web [249].

b. Worm_sim

Worm_sim [115] [250] [262] est un simulateur à cycle précis de réseaux d'interconnexion développé en C++ basé sur la bibliothèque de modèles standards STL (*Standard Template Library*). Worm_sim peut être utilisé pour simuler un large éventail d'architectures NoC avec la commutation de trou de ver (*wormhole switching*), (ex., NoCs ayant différentes topologies et différents algorithmes de routage, etc.), à l'aide des paramètres de performance pouvant être commandés et contrôlés par l'utilisateur (par exemple la taille de la mémoire tampon du canal, le délai de routage, le délai d'arbitrage du crossbar, etc.).

Worm_sim intègre des générateurs de trafic qui permettent aux utilisateurs de simuler le système sous plusieurs modèles de trafic populaires, comme les modèles de trafic Uniforme, point-chaud (*Hotspot*), Transpose, etc. Worm_sim fournit également un moyen efficace pour permettre aux utilisateurs de spécifier les conditions de trafic arbitraire pour le NoC en vertu de simulation. Plus précisément, l'utilisateur a le contrôle sur le taux de génération de paquets à chaque nœud IP individuellement, la taille du paquet et sa distribution, etc. Bien plus, il permet à l'utilisateur d'attacher un fichier de suivi pour chaque nœud IP, de sorte que le système sous simulation peut imiter l'état du trafic exacte des applications réalistes en réutilisant les fichiers de trace extraites de ces applications.

En plus, pour reporter les données de performance après la simulation, Worm_sim permet également à l'utilisateur de collecter des données sur la consommation d'énergie de communication en matière de simulation. Il supporte actuellement deux modèles de puissance, le modèle Ebit tel que présenté dans [115] [262], et le modèle de puissance d'Orion [263]. La bibliothèque de modèle de puissance d'Orion extraites de projet Orion Princeton supporte un ensemble riche d'API et a été compilé avec Worm_sim pour mettre en œuvre le soutien de simulation de l'énergie en utilisant le modèle d'alimentation Orion.

Worm_sim utilise une interface simple en ligne de commande pour configurer différents paramètres pour les systèmes sous simulation. Les paramètres configurés dans la ligne de commande sont, dans un sens global, pour l'ensemble NoC en vertu de simulation. Un moyen plus puissant pour la configuration du système est assuré par l'utilisation d'un fichier de configuration. Cela permet à l'utilisateur de contrôler les paramètres au routeur individuel ou au niveau IP.

c. HNoCS

HNoCS [251] est une implémentation open-source d'une plateforme de simulation NoC en utilisant OMNeT++ [264]. Le moteur de simulation à base d'événements OMNeT++ offre des APIs C++ pour un ensemble de services qui peuvent être utilisés pour modéliser, configurer, décrire la topologie, recueillir des données de simulation et effectuer des analyses. OMNeT++ fournit une complète documentation à l'utilisateur et un matériel de formation permettant de faire de nouvelles recherches en un court délai. La plateforme HNoCS utilise les fonctionnalités du module d'interface d'OMNeT++ pour supporter une sélection en temps réel de modules de simulation à partir d'une bibliothèque de composants paramétrés. Les modèles fournis supportent des configurations hétérogènes de NoC en termes de capacité de liaison et le nombre de canaux virtuels (*VC*: *virtual channel*). Les modules HNoCS disponibles aujourd'hui implémentent la commutation en mode de trou de ver (*wormhole switching*), avec la technique d'arbitrage à priorité égale (*Round-Robin*) ou bien le-vainqueur-prend-tout (*WTA*: *Winner-Takes-All*).

d. Atlas

Atlas [240] est un outil MSoC développé par GAPH - *Hardware Design Support Group* (voir l'architecture Atlas en figure 2.7). Atlas est écrit en Java et peut donc être compilé pour

tous les systèmes d'exploitation qui supportent Java. Il est doté d'une interface utilisateur simple.

L'environnement logiciel Atlas est entièrement dédié à la génération et à l'évaluation de performances du NoC Hermes. Atlas permet la génération de l'architecture NoC en VHDL (*VHDL*: *VHSIC* (*very-high-speed integrated circuits*) *Hardware Description Language*) synthétisable à partir des paramètres du NoC, Atlas génère le code VHDL synthétisable de l'architecture de communication en vue d'une implantation directe sur FPGA. Les paramètres d'entrée sont le choix du contrôle de flux ("*Handshake*" et "*Credit-Based*"), le nombre de canaux virtuels "*Virtual Channels*" en mode "*Credit-Based*", la taille du NoC, la taille des flits (8, 16, 32 ou 64 bits) et la taille des tampons pour chaque routeur (4, 8, 16 et 32 flits). Ainsi, Atlas peut générer, à la demande de l'utilisateur, le "*Test-Bench*" en SystemC relatif au NoC pour des simulations fonctionnelles et temporelles hors ligne au moyen des outils de simulation traditionnels. Les paramètres de la simulation consistent à définir les générateurs de trafic, la fréquence d'horloge du NoC ou de la fréquence pour chaque routeur, la sélection des émetteurs, récepteurs et liens entre eux (quel émetteur vers quel récepteur), le nombre de paquets à envoyer au(x) routeur(s) cible(s), la taille des paquets (plus précisément le nombre de flits par paquet, ainsi que la distribution des données dans les routeurs).

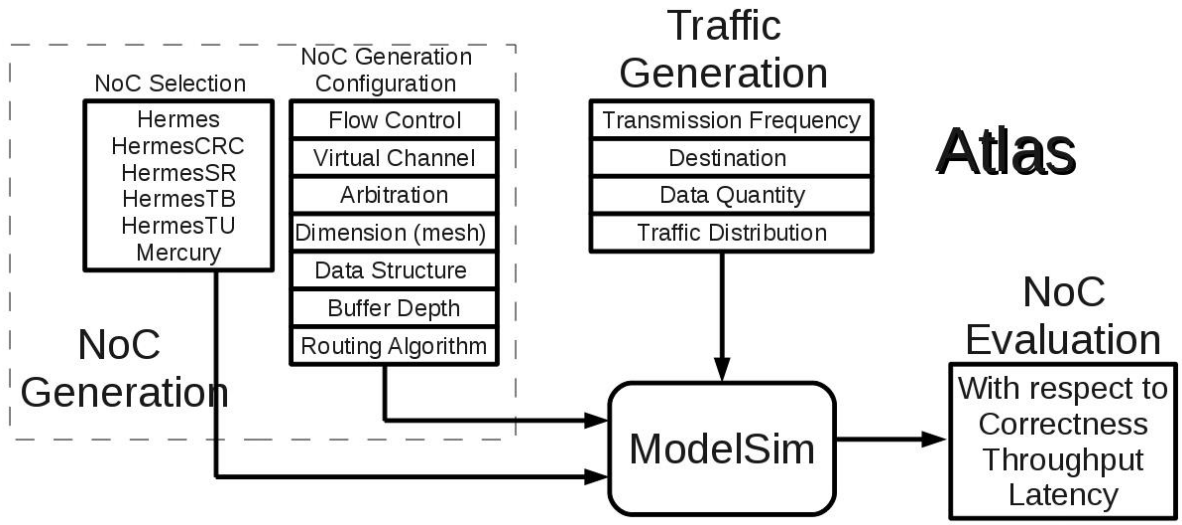


Figure 2.7 : Architecture Atlas

Toutes ces caractéristiques sont alors prises en compte pour la génération du code VHDL synthétisable et de ses modèles de simulation et de test en SystemC. Les fichiers VHDL synthétisables obtenus sont les suivants :

- Un package contenant tous les types et les sous-types permettant de dimensionner le NoC
- Une entité décrivant le fonctionnement des ports d'entrée et de leurs tampons
- Une entité décrivant le fonctionnement du bloc logique (arbitre et aiguilleur) et donc l'algorithme de routage
- Des entités pour chaque type de routeurs (routeurs centraux, routeurs en bas à droite, à gauche, en haut à droite,…)
- L'entité globale du NoC

En ce qui concerne l'évaluation des performances du NoC, l'outil génère les résultats obtenus par la simulation sous forme de tableaux et de graphiques. Grâce à cet outil, il est possible de voir comment évoluent plusieurs paramètres tels que la latence moyenne, le débit global du NoC, le débit dans chaque routeur, le nombre de cycles nécessaire pour transmettre un flit dans chaque routeur et le nombre de flits transmis par unité de temps.

Toutefois, la personnalisation du réseau sur puce telles que la taille de la mémoire tampon est difficile. En outre, l'outil ne peut pas soutenir la simulation des applications réelles.

e. µSpider

µSpider [28] [241] [265] est un outil de génération de MSoC/NoC (cf. figure 2.8), qui est développé par le groupe de recherche CEA-List. L'objectif des travaux de ce groupe est de fournir un outil pour faire face aux questions de prototypage multi-cibles complexes, mais cet outil ne supporte pas encore la simulation et l'évaluation du NoC. En d'autres termes, l'outil *µSpider* CAD vise seulement le domaine de la création et de l'optimisation du NoC, il ne comprend pas les options pour générer du trafic et d'évaluer les exécutions d'essais. Pour la génération de trafic, les algorithmes doivent être écrits dans l'outil Xilinx EDK et doivent être implémentés dans la conception du NoC à la main. En outre, ni le code source ni un exécutable compilé de l'outil est disponible sur Internet.

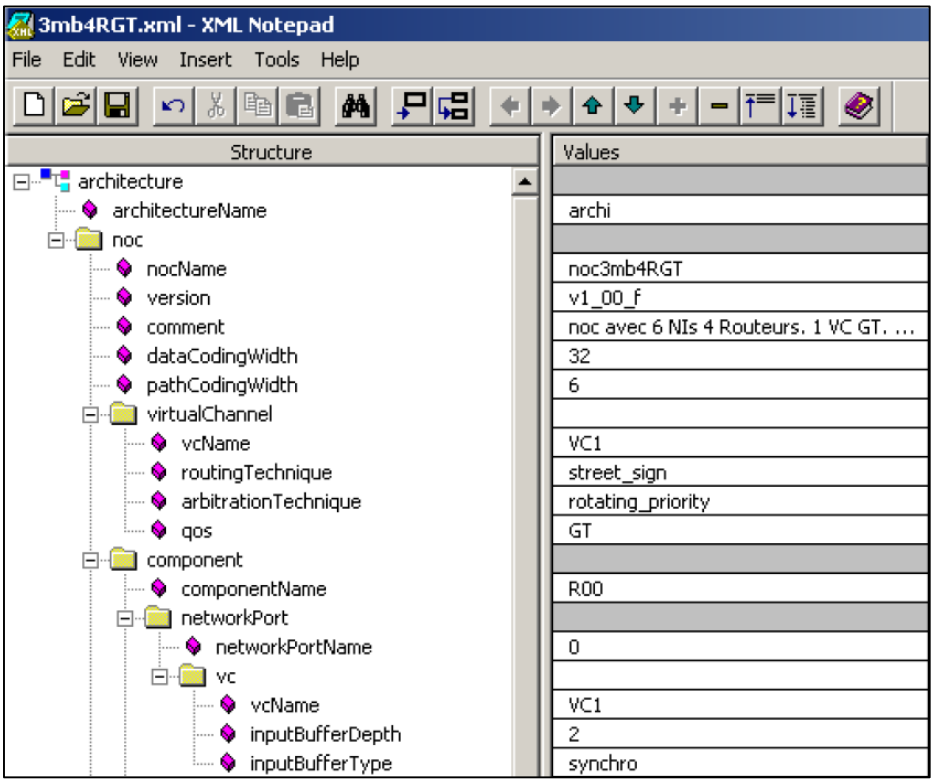


Figure 2.8 : Interface utilisateur *µSpider*

f. NoCGen

NoCGen [242] est un outil qui a été développé par l'Université de New South Wales, en Australie. Cet outil est basé sur des bibliothèques et il permet l'évaluation par synthèse et par simulation (cf. figures 2.9 et 2.10). NoCGen vise la génération d'un NoC comme une description matérielle (*HDL: Hardware Description Language*), où l'objectif est basé sur la description du routeur et la génération de trafic dans les routeurs. L'idée est de fournir une bibliothèque extensible de composants qui peuvent être utilisés pour construire un routeur. La méthodologie NoCGen augmente la flexibilité en permettant la création de différents réseaux d'interconnexion à partir du même modèle de routeur (cf. figure 2.10). Cet outil fournit des avantages similaires à ceux fournis par le mini-bus "*Sonics Silicon Backplane*" mais appliqué aux NoC et aux réseaux commutés.

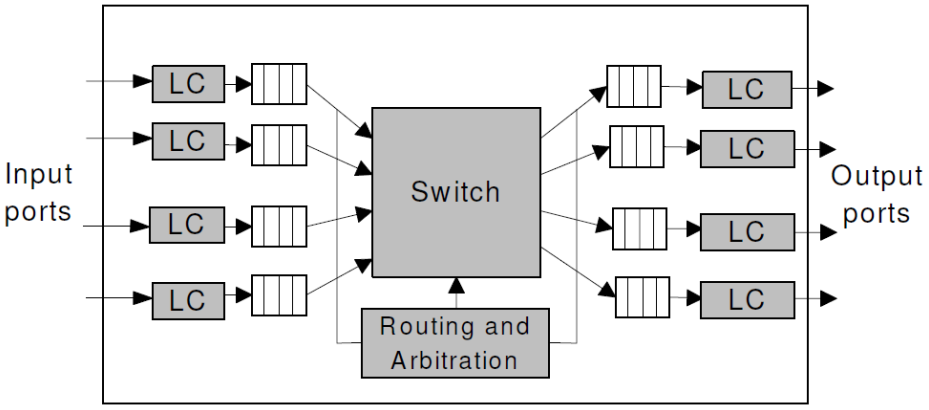


Figure 2.9 : Architecture du routeur canonique

Le NoCGen offre les possibilités suivantes: a) la mise en œuvre des différents routeurs NoC en utilisant une bibliothèque préconçue de sous-composants, b) la méthodologie de génération de HDL synthétisable pour configurer rapidement les routeurs haut dans le circuit NoC, compte tenu de la description d'un graphique de haut niveau, et c) un mélange d'environnement de simulation VHDL et SystemC pour simuler le circuit NoC généré avec une charge de travail réaliste. Le circuit NoC créé peut être synthétisé pour donner une estimation précise de la consommation d'énergie, la superficie et la latence.

Cependant, l'outil n'est pas disponible au public sur Internet, et ne considère pas l'utilisation des applications réelles pour tester le système.

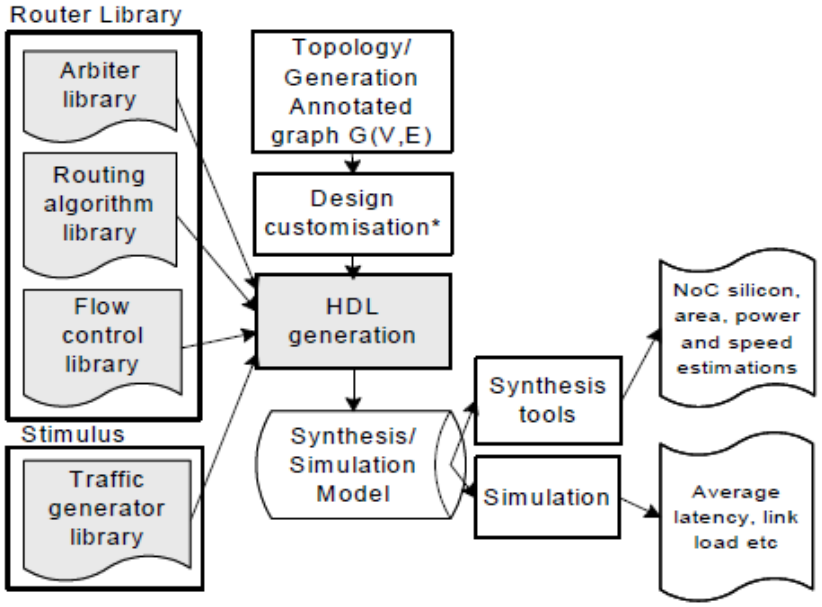


Figure 2.10 : Architecture NoCGen

g. Noxim

Noxim [243] est un outil de simulation MSoC/NoC développé à l'Université de Catane (Italie). L'outil est disponible gratuitement sur le site Noxim via *SourceForge* sous conditions de la licence GPL. Il est compilé pour un environnement Linux. Noxim est écrit en SystemC, il utilise le moteur de simulation SystemC ainsi que la bibliothèque de composants de réseau de niveau comportemental pour la simulation rapide. Il a une interface de ligne de commande pour définir plusieurs paramètres du NoC. En particulier, l'utilisateur peut personnaliser la taille du réseau, la taille de la mémoire tampon, la distribution de la taille des paquets, l'algorithme de routage, la stratégie de sélection, la vitesse de l'injection de paquets, la distribution de temps de trafic, le modèle de trafic, la distribution de trafic point-chaud (*hotspot*), etc.

L'outil Noxim permet la conception, la simulation et l'évaluation d'un NoC (cf. figure 2.11). Le simulateur permet d'évaluer le NoC en termes de débit, de latence et de consommation d'énergie. Ces informations sont fournies à l'utilisateur à la fois en termes de résultats moyens globaux et de résultats pour chaque communication. Plus précisément, l'utilisateur est autorisé à recueillir différents paramètres d'évaluation, y compris le nombre total de paquets/flits reçus, le débit moyen global, la latence max/min globale, la consommation totale d'énergie, et latence/débit/énergie pour chaque communication, etc.

Cependant, l'outil ne tient compte que de la topologie 2D Mesh et ne supporte pas la personnalisation de l'architecture NoC. De véritables applications ne sont pas également supportées par cet outil.

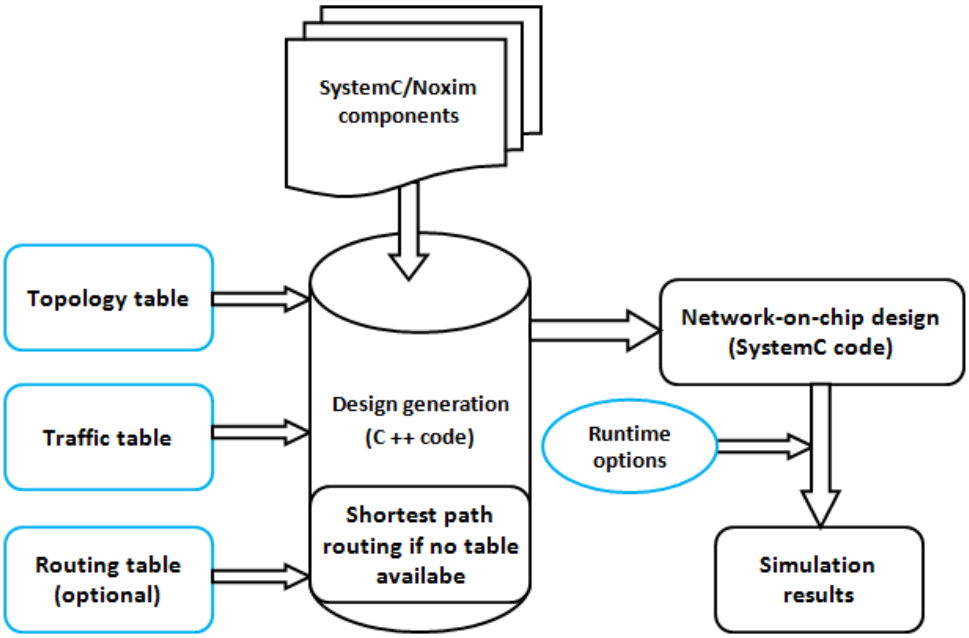


Figure 2.11 : Architecture Noxim

h. SICOSYS

Sicosys [226] [227] est un outil Linux avec une interface utilisateur visuelle pour la simulation de MSoC/NoC et avec une présentation visuelle des résultats calculés. La topologie, les routeurs et les paramètres de simulation sont définis dans des fichiers SGM (*SGM*: *Standard Generalized Markup*) (cf. figure 2.12).

Sicosys est une interconnexion d'usage général de simulateur de réseau qui permet la modélisation d'une large variété de routeurs de messages d'une manière précise. Les résultats sont très proches de ceux obtenus en utilisant des simulateurs de matériel mais à un coût de calcul plus faible. Afin de rendre l'outil facilement compréhensible, extensible et réutilisable. La mise en œuvre du simulateur est basée sur une technologie intimement liée à la conception orientée objets. En particulier, environ 110 classes distribuées dans environ 50.000 lignes de code ont été nécessaire. La portabilité est très élevée et peut être exécuté en toute plate-forme UNIX avec un standard compilateur C++.

Malheureusement, il y a seulement une version compilée Linux disponibles. L'outil est utile pour une simulation logique du réseau conçu, mais ne fournit qu'une faible quantité de structures prédéfinies. En outre, il ne crée pas une description au niveau de transfert de registres RTL (*RTL*: *Register Transfer Level*) du réseau et les scénarios de trafic de simulation sont saisis manuellement par l'utilisateur.

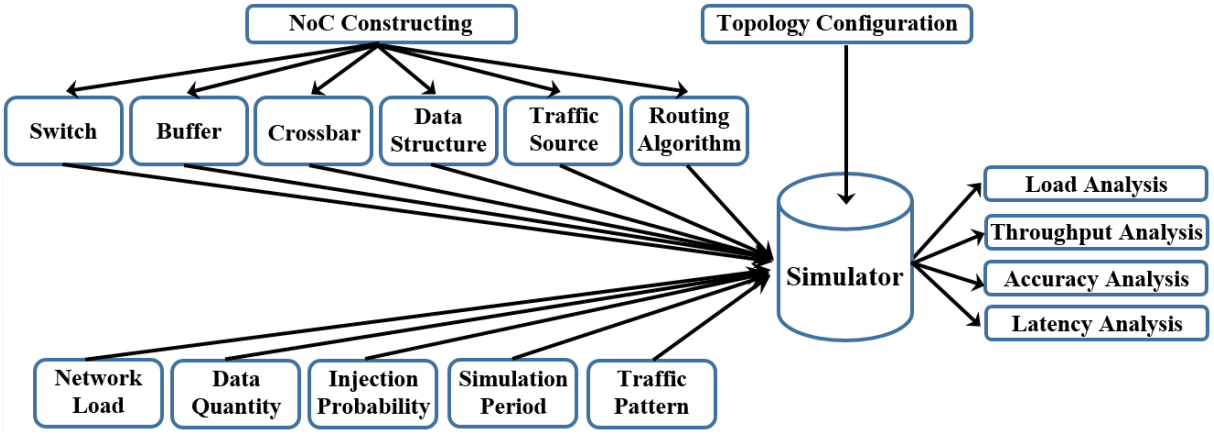


Figure 2.12 : Architecture SICOSYS

i. Nostrum

Nostrum [244] [245] est un environnement de simulation MSoC/NoC (*NNSE*: *Nostrum NoC Simulation Environment*), cet outil est basé sur le SystemC, il est développé à l'Institut Royal de Technologie de Stockholm (Suède). L'outil est disponible sur Internet et peut être téléchargé à partir du site Nostrum (http://www.ict.kth.se/nostrum/NNSE/). L'outil fournit une interface graphique qui permet la configuration des réseaux sur puce et leur évaluation avec différents modèles de trafic. La topologie, les routeurs, la répartition du trafic, et les paramètres de simulation sont définis dans des fichiers internes et les résultats de simulation peuvent être affichés dans une variété de graphiques (cf. figure 2.13). Nostrum est compilé pour les environnements Linux et dispose d'une interface utilisateur visuelle, mais la variété des options configurables décrites est très limité. En outre, les modèles de routeurs sont décrits comme des routeurs à tampon-moins (*buffer-less routers*). L'outil ne fournit pas une description RTL du NoC configuré. Donc, il est seulement utile pour une simulation théorique.

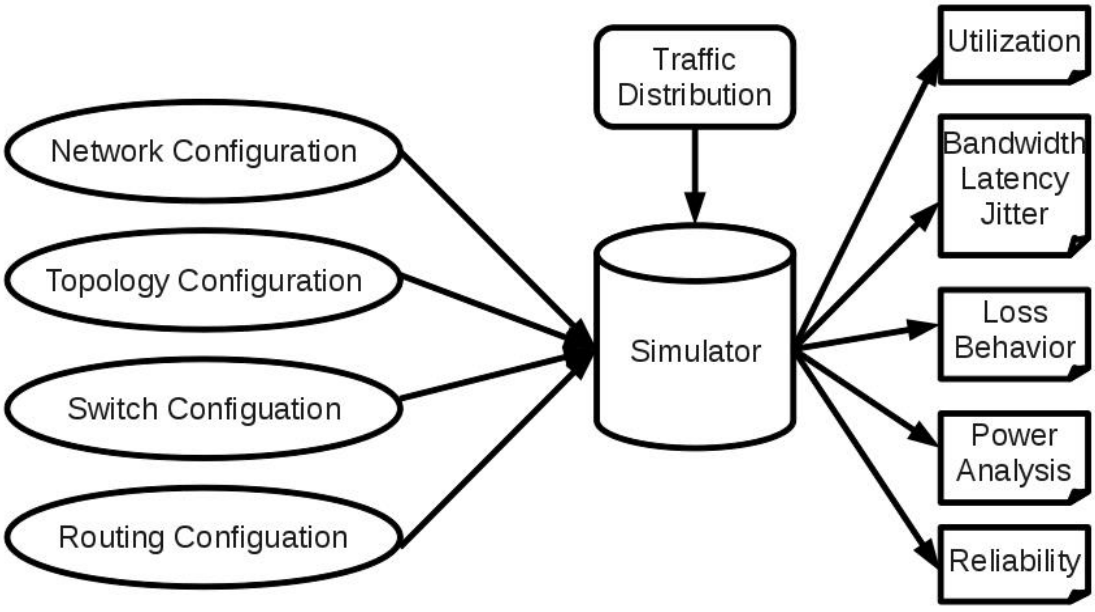


Figure 2.13 : Architecture Nostrum

j. NIRGAM

NIRGAM [77] est simulateur SystemC à base d'événements discrets et de cycle précis pour la conception de NoC en termes d'algorithmes de routage et d'applications sur diverses topologies (2D Mesh et 2D Torus actuellement mises en œuvre). NIRGAM est le fruit d'une recherche collaborative entre le groupe d'électronique de conception des systèmes à l'École

d'électronique et d'informatique de l'Université de Southampton au Royaume-Uni et le département d'informatique et de génie informatique de l'Institut national de technologie à Malaviya, à Jaipur en Inde. Ce projet est financé par l'EPSRC (UK).

NIRGAM permet la génération de NoC, de la simulation jusqu'à l'évaluation du NoC (cf. figure 2.14). Il est écrit en SystemC et peut être exécuté en toute plate-forme UNIX avec un standard compilateur C++. Il support diverses distributions et scénarios de trafic. Il est disponible gratuitement sur son site Internet (http://nirgam.ecs.soton.ac.uk/), et il est open-sources, ce qui permet aux concepteurs de pouvoir l'adapter à leurs besoins de simulation, comme par exemple implémenter d'autres topologies, ou d'autres patterns trafic, etc.

Cependant, l'outil en sa version originale ne considère que les topologies 2D Mesh et 2D Torus et ne supporte pas la personnalisation de l'architecture NoC. D'importantes applications ne sont pas également prises en compte dans cet outil.

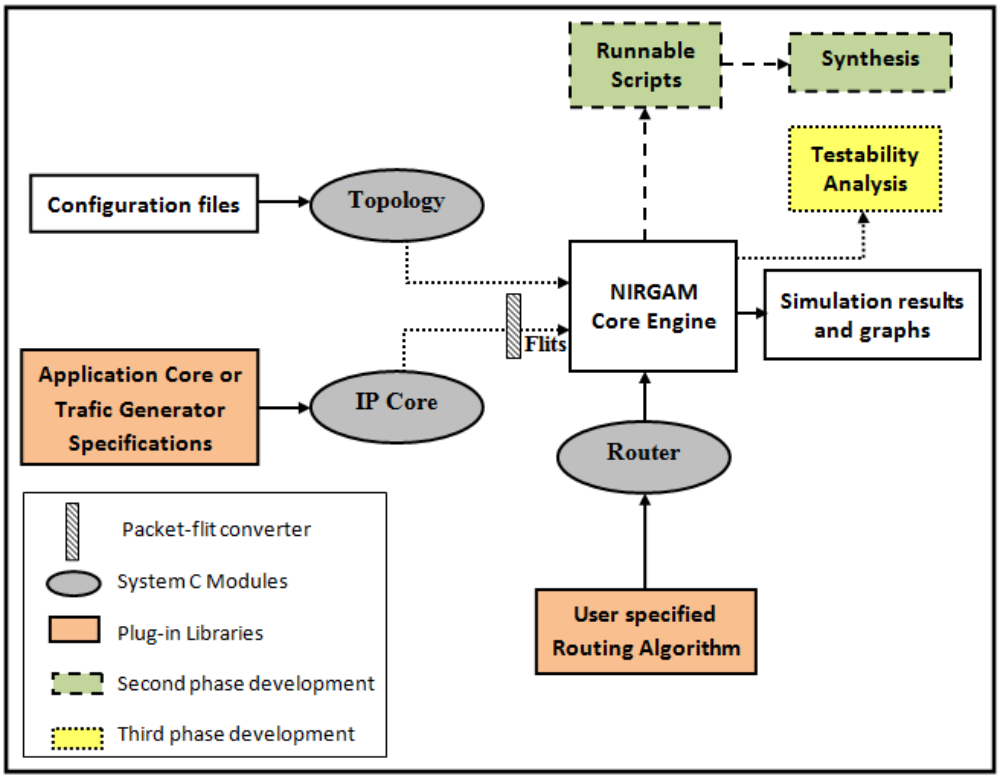


Figure 2.14 : Architecture Nirgam

C. Synthèse

Cette section est focalisée sur l'étude comparative de quelques-uns des simulateurs décrits ci-dessus (section VI.2.1.2.B), la comparaison est basée sur différents paramètres de configuration du NoC tels que la topologie, l'algorithme de routage, le contrôle de flux, et les paramètres d'évaluation tels que la latence, le débit, la consommation d'énergie, et la charge des liens (cf. tableau 2.3).

Fonctionnalité	*Atlas*	*µSpider*	*NoCGen*	*Noxim & Noxim++*	*SICOSYS*	*Nostrum*	*Nirgam*
Interface utilisateur	Visuel	Visuel	Ligne de commande	Ligne de commande	Visuel	Visuel	Ligne de commande
Génération NoC	✓	✓	✓	✓	✓	✓	✓
Topologie	2D Mesh / Torus	2D Mesh / Personnalisé	Personnalisé	2D Mesh / irrégulière	2D Mesh / Torus / irrégulière / (Square) 2D Mesh à minimal distance avec une couverture autour des liens	2D Mesh / Torus / Arbre / Ring	2D Mesh / Torus
Algorithme de routage	Ouest d'abord / Négatif d'abord / XY / Personnalisé	XY / *Streetsign*	Dimensionnel	XY / West d'abord / Nord d'abord / Négatif d'abord / Pair-Impair / *DyAD* / entièrement adaptatif / table de routage personnalisé	Adaptatif / Déterministe / Minimal / Non-Minimal	Aléatoire / XY / Delta-XY / déformation minimale	XY / Pair-Impair / Source
tampon de routeur	(4, 8, 16 ou 32 flits)	✓	✓	✓	Taille / délai de données	✓	✓
le numéro de port du routeur	4 + interface réseau	✓	✓	5 à 10	✓	✓	✓
Arbitrage	*Round Robin* / priorité	*Round Robin* / *FIFO*	Matrice / *Round Robin* / Priorité à base de loterie / statique	Aléatoire / Le niveau de tampon / le chemin voisin	--	--	--
Le contrôle de flux	A base de crédit / *Handshake*	De bout en bout	*Store and forward* / *Wormhole* / *Virtual cut-through* / Demande de subvention point à point / *Handshake* point à point	--	*Wormhole* / *Virtual Cut-through* / *Store and forward*	*Wormhole* / Non Prioritaire / Priorité pondéré / *Straightthrough*	Wormhole

Les canaux virtuels	1/2/4	✓	✓	--	✓	✓	✓
Taille Flit	8 à 64 bits	--	--	✓	✓	--	✓
Interface réseau	--	✓	--	--	--	✓	--
bande passante du lien	= Taille Flit	✓	--	--	--	✓	--
Qualité de service	BE / GT	BE / GT	--	--	--	BE / GL	BE / GT
correction d'erreur / détection	injection d'erreur / *CRCCheck* / *Hamming*	--	--	--	--	Injection d'erreur	--
Améliorations automatiques	--	Câblage sur mesure pour raccourcir les chemins / câblage pour le trafic de l'application spécifique / dimensionnement de la mémoire tampon	--	Dimensionnement de la mémoire tampon / sélection de topologie irrégulière	--	--	--
Les fichiers de sortie	VHDL / SystemC	VHDL	VHDL	SystemC	--	--	texte
les modèles de trafic	✓	--	✓	✓	✓	✓	Uniform/ Hotspot/ Shuffle/ Bit-reversal/ Multimedia
Générateurs de trafic	Uniforme / Normal / Pareto	Aucune option avec l'outil lui-même, les utilisateurs doivent entrer le trafic manuellement	Trafic en fonction d'une bibliothèque, Couverture SystemC à appliquer pour la générateurs de trafic de haut niveau.	Poisson / Rafale / Pareto	Shuffle parfait / Autocréation	Normal / Aléatoire / Uniforme / Spatial	Base CBR (*Constant Bit Rate*) / Bursty / Trace

La répartition temporelle	--	--	--	✓	Autocréation	Constant / aléatoire / normal	Constant / aléatoire
Simulation NoC	avec ***ModelSim***	✓	✓	✓	✓	✓	✓
Évaluation NoC	✓	Ne supporte pas	Ne supporte pas	✓	✓	✓	✓
Débit	Global / local (min, max, moyen)	--	--	Global / local (min, max, moyenne)	Global	Global / local (min, max, moyenne)	Global (moyenne)
Latence	Global / local (min, max, moyenne)	--	--	Global / local (min, max, moyenne)	Global	Global / local (min, max, moyenne)	Global (moyenne)
Paquets transmis	Global / local	--	--	--	✓	Global / local	Local (moyenne)
Latence des tampons	Global / local	--	--	--	Moyenne Global	--	--
Visualisation des résultats	GNU 2D + 3D graph / Table	--	--	Tableau	L'utilisation de fichier texte / utilisation de tampon	Graph 2D	L'utilisation de fichier texte / Graph 2D ou 3D par GNUPLOT ou MatLab
Consommation estimation	✓	Ne supporte pas	Ne supporte pas	✓	--	--	✓
Paramètres considérés	utilisation du routeur	--	--	transmission de lien / sélection de saut / routage de paquet / routeur inoccupé	--	--	--

Tableau 2.3 : Comparaison des simulateurs NoC

Notre choix de simulateur est porté sur le simulateur NIRGAM [77]. Il a été développé par le groupe d'électronique de conception des systèmes à l'École d'électronique et d'informatique de l'Université de Southampton au Royaume-Uni et le département d'informatique et de génie informatique de l'Institut national de technologie à Malaviya, à Jaipur en Inde. L'intérêt principal du choix de NIRGAM est qu'il est un simulateur open-source. Ce logiciel libre est écrit en System C et peut être exécuté en toute plate-forme UNIX avec un standard compilateur C++, nous utiliserons le compilateur g++ dans le cadre de notre projet. Ce qui nous permis d'apporter des modifications et des améliorations à ce simulateur afin de l'adapter à nos besoins de simulation. En plus, il est totalement configurable (type et taille de la topologie, fréquence d'horloge, taille des tampons, taille des flits, les canaux virtuels, pattern de trafic, générateurs de trafic, etc.), il est capable d'évaluer plusieurs paramètres de performance tels que la latence, le débit, la consommation d'énergie et la charge des liens. NIRGAM fournit en sortie des résultats sous forme de graphs Gnuplot ou des codes exécutables en MatLab afin de dessiner les graphs correspondants aux résultats.

NIRGAM est un simulateur à base d'événements discrets et de cycle précis pour la conception de NoC. Il supporte diverses distributions et scénarios de trafic. Dans sa version originale, NIRGAM ne prend en compte que les topologies 2D Mesh et 2D Torus, il ne supporte pas la personnalisation de l'architecture NoC. La version que nous utilisons dans cadre de cette thèse a été modifiée par moi-même et prend en compte bien tous types d'architectures (régulières ou pas).

VI.2.2. Evaluation des NoCs par des systèmes existants

Un grand nombre de différentes architectures NoC ont été proposées par différents groupes de recherche. La topologie, l'algorithme de routage, le contrôle de flux, la technique de commutation, la technique d'arbitrage, la structure des paquets et la pile des protocoles de communication sont les caractéristiques les plus importantes qui distinguent ces diverses propositions de NoC. Dans ce qui suit, nous vous présenterons quelques-uns de ces réseaux sur puce.

ARTERIS [266] est le premier outil de conception de réseaux sur puce à avoir été commercialisé. Il est une suite d'outils développés par la compagnie ARTERIS. Cette suite comprend deux outils d'aide à la conception et à l'exploration architecturale. ARTERIS permet la construction d'une topologie de réseau à partir d'un cahier des charges, où sont spécifiés les besoins en performances de l'application visée ainsi que les limites de cout et de surface pouvant être admis. Cet outil est basé sur un modèle de simulation SystemC du routeur ARTERIS reprenant les mécanismes de routage, de bufférisation ainsi que les mécanismes de contrôle de flux et de qualité de service. Les mesures de performances telles que les débits et la latence sur le réseau sont produit à chaque simulation à partir de modèles de trafic synthétiques. Le concepteur peut modifier manuellement la topologie donnée en entrée en fonction des performances obtenues afin d'arriver à la meilleure adéquation (besoins/couts) [267]. L'intérêt majeur de ce genre d'outils est de permettre la modélisation et la simulation de plusieurs instances de réseaux sur puce de façon visuelle et très rapide, ce qui se traduit par un gain substantiel en temps de conception. Une fois la topologie fixée, ARTERIS permet de générer une description matérielle du réseau. Il est à noter ici que la liberté donnée à l'utilisateur de choisir une topologie arbitraire écarte de fait la possibilité d'utiliser des algorithmes de routage algébriques et impose l'utilisation de tables de routages au niveau de chaque routeur. ARTERIS utilise un protocole de transport propriétaire appelé NTTP (*NTTP*: *NoC Transaction and Transport Protocol*) qui permet d'utiliser les transactions requête/réponse et d'intégrer les fonctionnalités traditionnelles des bus (adresse, transaction *load*/*store*, *burst*, bit de parité). Ceci lui permet d'être compatible avec un grand nombre d'interfaces standards (ex., AMBA AHB (*Advanced High-speed Bus*), AMBA AXI (*Advanced eXtensible Interface*), OCP 2.0 (*Open Core Protocol*)).

Une architecture sur puce appelée "évolutive, programmable et réseau intégré" (*SPIN: Scalable, Programmable and Integrated Network*) a été proposé par Guerrier et Greiner du laboratoire LIP6 en 2000 [44] [268] . Ils ont également proposé la topologie Fat-arbre en raison de son faible diamètre et son efficace implémentation par la technologie d'intégration à très grande échelle (*VLSI*: *Very-Large-Scale Integration*). Elle met en œuvre le modèle de communication à passage de message.

Benini et De Micheli ont proposé un micro-réseau de commutation de paquets basé sur une infrastructure de communication pour la conception des SoCs [25]. De même, le réseau sur silicium Ætheral [103] [104] est proposé par le laboratoire de R&D de Philips (maintenant NxP). Ils proposent de concevoir un SoC à base de réseau de communication pour supporter un débit garanti pour un trafic en temps réel et une communication au meilleur effort (*BE*: *Best Effort*) pour le reste du trafic. Ætheral est un réseau qui définit deux classes de qualité de service, la classe GS (*GS*: *Guaranteed Service*) et la classe BE. Les mécanismes de garantie de services sont implémentés au niveau des routeurs et se basent sur une technique d'allocation statique des ressources TDMA (*TDMA*: *Time division multiple access*) du réseau et cela pour chaque flux de données. L'allocation se fait par des tables appelées "*Slot Tables*" où chaque ligne correspond à un intervalle temporel (ou *slot*), et chaque colonne correspond à un port de sortie du routeur. La valeur à l'intersection ligne/colonne indique le numéro de port d'entrée pouvant émettre durant un *slot* donné. Cette approche évite de fait toute contention mais au prix d'une latence moyenne pouvant être plus élevée à celle de la classe de service BE. Initialement et avant l'allocation des *slots* au flux de données, le réseau fonctionne en mode BE uniquement.

Une autre proposition appelée Linköping SoCBUS combine les avantages de la communication à commutation de circuits avec celles de la communication à commutation de paquets.

La société STMicroelectronics a proposé le réseau à commutation de paquets STNoC [43]. L'originalité du STNoC réside dans sa topologie régulière en Spidergon, supportant des algorithmes de routage très simples nécessitant un minimum de ressources matérielles. Le point fort de la topologie Spidergon découle de l'encombrement minimal des interconnexions qu'elle engendre. En effet, quelle que soit la taille du réseau STNoC il est toujours possible de le transformer en une implémentation planaire avec un minimum de croisements entre les liens. Des interconnexions fortement encombrées avec beaucoup de croisements nécessitent plus de surface et sont à l'origine d'une surconsommation électrique. Une autre caractéristique de la topologie est qu'elle peut être adaptée aux besoins de l'application ciblée en retirant des liens inutilisés sans pour autant modifier les algorithmes de routage, ce qui ajoute un degré de flexibilité non négligeable. Le STNoC utilise une approche se basant sur une séparation stricte entre la logique interne des routeurs et les liens servant au transport des paquets. En plus du support implicite du mode GALS (*GALS*: *Globally Asynchronous, Locally Synchronous*), cette approche donne la possibilité d'avoir plusieurs configurations de liens en fonction des besoins en performances. En effet, selon les contraintes physiques, il est possible d'utiliser des liens séries ou parallèles, il est aussi possible de choisir la fréquence de sérialisation/fonctionnent en fonction des débits souhaités indépendamment de la fréquence utilisée par la logique interne des routeurs.

Le groupe de collaboration de chercheurs entre l'institut royal de technologie KTH de Stockholm et le centre national de la recherche technique VTT (en finnois, *Valtion Teknillinen Tutkimuskeskus*) de Finlande ont proposé la topologie KTH-VTT. La topologie KTH-VTT est un mesh bidimensionnel pour une architecture NoC sur le développement d'un nombre important de systèmes sur puce complexes [269].

Hermes [270] est un réseau avec une stratégie de bufférisation en trou de ver (*wormhole*) et un routage XY sur une topologie 2D Mesh, dont la taille peut être fixée en fonction des préférences utilisateur. Divers paramètres peuvent être modifiés lors de la phase de conception, tels que la largeur des flits, la longueur des paquets, ou encore la profondeur des tampons

internes. Le réseau est conçu pour supporter uniquement le mode de fonctionnement synchrone. C'est d'ailleurs l'une de ses principales limitations. La particularité de ce réseau est qu'il est en accès libre, il est ainsi possible d'ajouter et/ou modifier certaines fonctionnalité directement sur les sources HDL du réseau. Une interface très intuitive est fournie, et permet de générer une instance spécifique du réseau en fixant chaque paramètre. Il est aussi possible à travers la même interface de tester l'instance du réseau sous plusieurs scénarios de trafic (*Uniforme*, *Pareto On/Off*, *Poisson*) au moyen de générateurs écrit en SystemC. Cependant, comme aucune librairie d'IPs n'est fournie avec le réseau, particulièrement des interfaces aux bus standard ; il est nécessaire de développer une interface réseau pour chaque protocole utilisé. Une dernière caractéristique importante du réseau Hermes est qu'il est compatible avec une implémentation matérielle (ex., *FPGA*: *Field Programmable Gate Array*); en effet le code VHDL généré est complètement synthétisable. Toutefois, l'adjonction de moniteurs de performances matérielles (non inclus dans Hermes) est nécessaire afin de permettre une étude des performances sous conditions de trafics réels.

Cette étude de la littérature nous montre qu'un grand nombre de chercheurs ont proposé différentes architectures à base de réseau à commutation de paquets pour la conception d'infrastructure de communication inter-IPs dans les systèmes sur puce.

VI.3. Synthèse

Dans ce deuxième chapitre, nous avons présenté un panorama d'état de l'art sur les approches proposées dans la littérature pour l'optimisation des performances du réseau NoC donné, et nous avons ressorti les difficultés d'optimisation et de conception de telles architectures. En effet la taille de l'espace d'exploration et, par conséquent, la complexité des algorithmes de recherche sont directement liés au nombre de paramètres devant être pris en compte. D'où la nécessité d'avoir des outils d'aide à la conception pour les NoCs. Ces outils, en facilitant la mise en œuvre et l'implémentation d'une instance d'un réseau, accélèrent le processus de recherche et d'optimisation. Dans ce contexte, nous avons présenté les méthodes et les outils de conception et d'expérimentation des NoCs classés en deux grandes catégories, expérimentation par des modèles de systèmes (modèles analytiques ou par simulation) et expérimentation par systèmes existants. Pour des SoC dynamiques, dans lesquels le modèle de trafic n'est pas connu au préalable, une architecture d'interconnexion sur puce personnalisée est nécessaire pour traiter les demandes de charge de travail imprévisibles et les changements imprévisibles des tâches actives et des exigences de communication. Nous avons présenté quelques implémentations de NoCs qui nous semblaient les plus significatives. Il est important de souligner qu'il existe actuellement un grand nombre d'implémentations de NoC avec des fonctionnalités de plus en plus avancées telle que l'évitement de la congestion, des mécanismes de garantie de qualité de service ou encore des fonctionnalités de transport temps réel. Cela montre bien l'engouement actuel que suscitent les NoCs dans la communauté de la recherche en micro-électronique.

Cette étude de l'existant nous a permet de se positionner par rapport à ce qui se fait actuellement dans le domaine des NoCs. Ainsi, nous avons proposé trois approches d'optimisation des performances du NoC qui seront présentées en détail dans les chapitres suivants de cette thèse. Ces trois approches permettent aux concepteurs d'optimiser et de personnaliser une architecture d'interconnexion sur puce afin qu'elle correspond à une grande charge de travail des applications SoCs. L'objectif de ces approches est d'explorer et d'évaluer les performances de l'architecture d'interconnexion sur puce pour mieux répondre aux besoins d'un maximum de type d'applications. Le travail de cette thèse s'inscrit dans deux niveaux de l'architecture NoC, le niveau physique et le niveau communication (cf. figure 2.15).

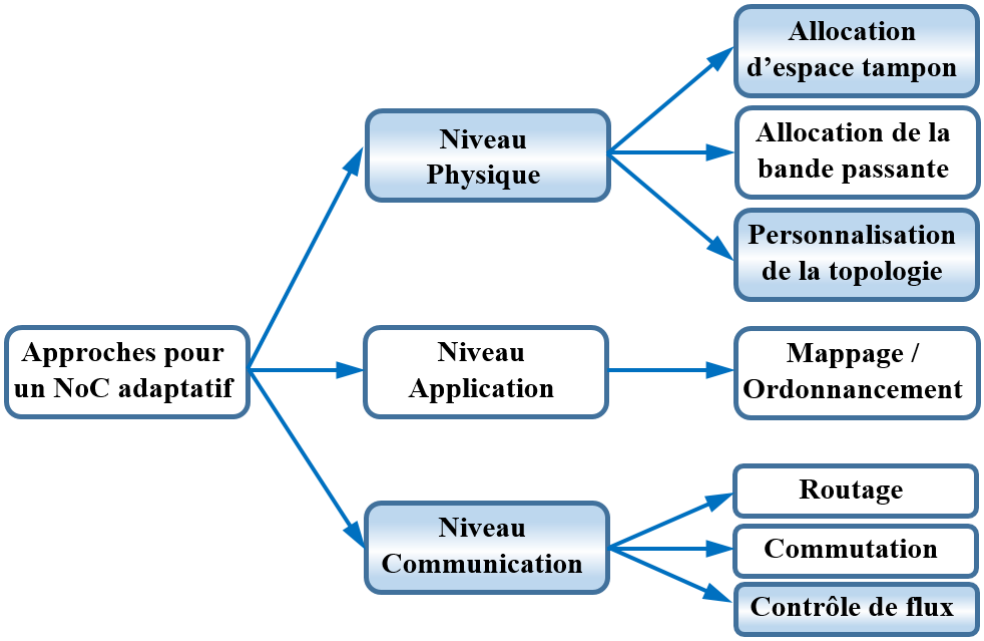


Figure 2.15 : Contribution des travaux de ma thèse

Dans le niveau physique, deux approches sont présentées. La première approche consiste à personnaliser l'architecture d'interconnexion sur puce par l'insertion de longs liens entre des commutateurs stratégiques, sélectionnés selon des critères de performances physiques de l'architecture sur puce (ex., réduire la distance moyenne de l'architecture et augmenté l'inter-connectivité locale, ce qui est connu sous le nom "degré de clusterisation"). Ce travail est présenté en détail dans le chapitre 3. L'autre approche consiste à allouer dynamiquement des tailles de tampons nécessaire à l'application spécifique afin d'améliorer les performances du NoC. Cette approche est basée sur la modélisation par réseau à compartiments présentée ci-dessus (section VI.2.1.1.F), elle est inspirée du travail présenté par Guffens et al. [271] [272]. Ce travail va être également détaillé dans le chapitre 3.

Dans le niveau communication, une approche de contrôle de flux dynamique basée sur la théorie des systèmes dynamiques présentée ci-dessus (section VI.2.1.1.E). Cette approche est présentée en détail dans le chapitre 4.

IV. Chapitre 3 : Approches de personnalisation des NoCs

Dans ce chapitre, nous présentons deux approches de personnalisation de l'architecture d'un NoC candidat. Ces approches se situent au niveau physique, par rapport à notre classification des approches de personnalisation présentée dans le chapitre 2 (section VI.1). La première approche consiste à personnaliser la topologie du NoC afin de l'adapter aux applications SoC, elle est basée sur l'ajout de liens entre des commutateurs stratégiques. Quant à la deuxième approche, elle consiste à personnaliser la topologie du NoC pour l'adapter à une application spécifique, elle est basée sur l'allocation dynamique de l'espace tampon des commutateurs selon les conditions de fonctionnement du NoC.

VII.1. Méthodologie d'exploration de l'espace de conception

Dans cette section, nous présentons une méthodologie d'exploration de l'espace de conception pour la personnalisation ou l'adaptation d'un NoC candidat pour correspondre à un modèle de trafic d'une application particulière sous contrainte d'un budget initial de ressources. Cette méthodologie donne aux concepteurs des NoCs, dans des stades très tôt de la conception, une rétroaction utile plus rapide à la fois sur une granularité de niveau grossier et fin tels que l'utilisation de tampons et la latence par nœud et par flux de données. En appliquant cette méthodologie, les goulots d'étranglement dans le NoC peuvent être déterminés au préalable et peuvent être évités ensuite par exemple par l'insertion des liens supplémentaires pour alléger les commutateurs congestionnés ou bien par l'allocation de la taille de mémoire tampon requise pour chaque canal. Comme le montre la figure 3.1, compte tenu des paramètres de l'architecture comme l'algorithme de routage et la bande passante des liens, et des paramètres de l'application tels que le nombre de tâches et de leurs modes de communication, les concepteurs peuvent analytiquement évaluer les métriques de performance (c.à.d. la latence, le débit et la charge de communication) et le coût (c.à.d. la consommation d'énergie et la surface silicium) [110].

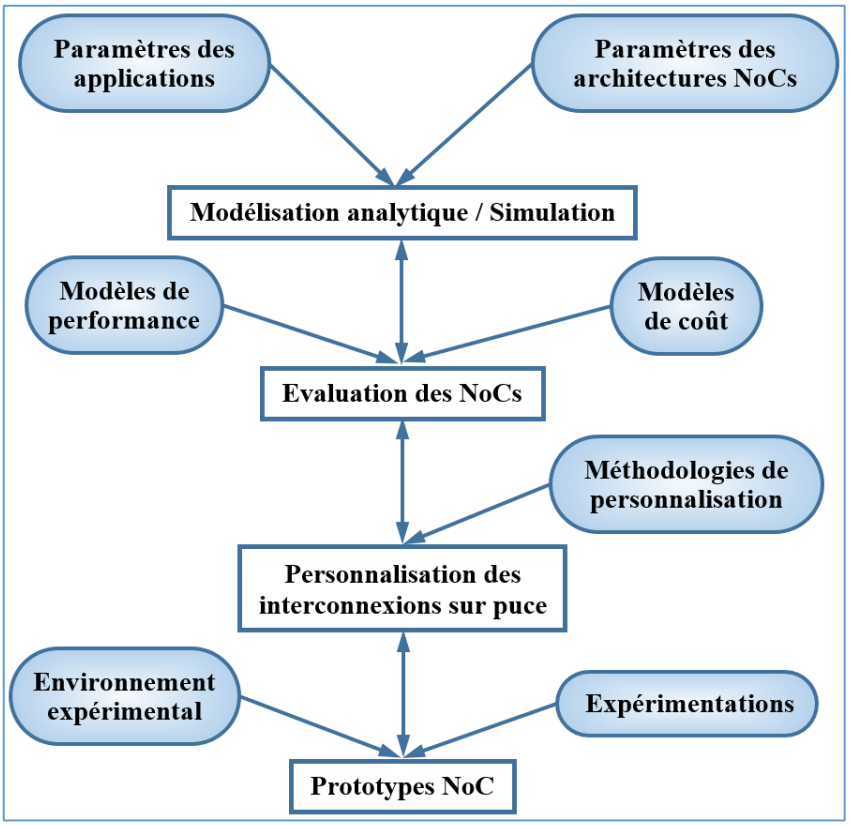


Figure 3.1 : Représentation graphique de la méthodologie DSE

Après la modélisation analytique, les concepteurs peuvent quantitativement explorer les compromis entre la performance, la surface et l'énergie de l'architecture NoC cible, puis la personnaliser, par exemple, par l'allocation de l'espace de mémoire tampon pour chaque canal pour adapter le NoC à une application spécifique, ou par l'ajout de plusieurs liens pour mieux répondre aux besoins en charge de travail de plusieurs applications SoCs. Le prototypage FPGA pour une évaluation précise des performances, de l'énergie et des besoins de surface silicium, est la dernière étape dans la méthodologie d'exploration de l'espace de conception pour évaluer l'efficacité des algorithmes de personnalisation. Il permet d'explorer l'espace de conception et d'identifier les goulets d'étranglement potentiels dans le système, car il permet aux concepteurs d'exécuter des modèles à cycle précis sur un matériel réel, qui est beaucoup plus rapide que la simulation.

Dans ce chapitre, nous démontrons l'efficacité de cette méthodologie à travers deux approches de personnalisation de NoC que nous avons proposées ; la première approche basée sur l'insertion de liens pour personnaliser la topologie du NoC (section VII.1.1), et la deuxièmes approche basée sur l'allocation dynamique de l'espace de mémoire tampon pour la conception des routeurs NoC adaptés à une application spécifique (section VII.1.2). Sur les deux approches nous n'avons pas réalisé la dernière étape de prototypage sur FPGA du NoC candidat, faute de moyen.

VII.1.1. Approche de personnalisation de NoC par insertion de liens

Des études récentes ont montré que, pour améliorer les performances d'un domaine spécifique d'application SoC, l'architecture d'interconnexion sur puce (NoC) sur laquelle le SoC est basé doit être personnalisée au moment de la conception. Les approches de personnalisation du NoC qui existantes dans la littérature sont donc destinées à des applications spécifiques, offrant des SoCs spécifiques à ces applications cibles. Elles traitent la sélection de l'architecture d'interconnexion sur puce qui pourrait accueillir le modèle de trafic de données spécifique à l'application attendue lors de la phase d'exploration d'espace.

Pour concevoir des SoCs dynamiques, dans lesquels le modèle de trafic des applications n'est pas connu ou prévisible à l'avance, un réseau NoC efficace est nécessaire. Afin que le NoC prenne en charge différents types de trafic, il doit être donc personnalisé. Une des principales difficultés est la maîtrise des paramètres permettant d'obtenir le NoC approprié aux besoins de plusieurs applications. En raison de la taille de l'espace de conception (topologie, chemins, routage, etc.), il n'est pas possible d'explorer toutes les solutions en fonction des différents compromis possibles. Il est donc nécessaire de disposer de méthodologies et d'outils pour personnaliser l'architecture du NoC afin de concevoir le NoC adapté aux besoins d'un maximum de types de trafic de différentes applications.

Comme cela a été présenté dans le chapitre 1 (section V.2.4), plusieurs réseaux d'interconnexion sur puce (2D Mesh, Ring, Spidergon, WK, X-Mesh, etc.) de base ont été étudiés et adaptés récemment pour les SoCs. Ces réseaux d'interconnexion sur puce ont des caractéristiques différentes en fonction des critères physiques de leur topologies [3] [4] [7] [38], tels que le diamètre, la distance moyenne, le degré, la bissection, le nombre de liens et le degré de clusterisation (cf. chapitre 1, section V.2.4.2). Ces critères fournissent une première indication sur certains indicateurs de performance et de qualité de service QoS. Par exemple, si l'architecture d'interconnexion sur puce a un petit diamètre, un haut degré, une petite distance moyenne et un degré de clusterisation élevé, donc le réseau d'interconnexion sur puce peut prendre en charge différents modèles de trafic de données tout en assurant une bonne performance de communication. Cependant, il n'y a pas de réseau d'interconnexion sur puce universel, qui peut prendre en charge tous les modèles de trafic des applications SoC avec une garantie de QoS exigée par les utilisateurs.

VII.1.1.1. Hypothèses de base

Pour permettre aux concepteurs d'adapter un réseau NoC donné afin qu'il correspond à une grande charge de travail de différentes applications, nous avons proposé un algorithme d'insertion de liens. Cet algorithme se résume en trois étapes ; la première étape consiste à sélectionner les liens stratégiques à ajouter sur le NoC; la deuxième étape consiste à la mise en œuvre de ces liens sur l'architecture NoC de base souhaitée ; et la troisième étape, quant à elle, consiste à évaluer la performance du NoC personnalisé par le routage des flux d'un modèle de trafic appliqué donné. Toutefois, il est difficile de choisir judicieusement les liens à insérer sur le NoC, qui convient le mieux aux besoins de plusieurs applications, sans connaître à l'avance les modèles de trafic de communication.

Afin de personnaliser la topologie du NoC sans avoir recourt au modèle de trafic, nous avons voulu améliorer les caractéristiques physiques de la topologie du NoC, ce qui aura forcément

un impact direct sur les performances de communication de celui-ci. Nous avons donc mis en place plusieurs hypothèses, comme par exemple :

- Un petit diamètre permet une communication rapide entre les nœuds les plus éloignés. En d'autres termes, la latence maximale est proportionnelle au nombre maximal de sauts, c'est pour cette raison que le diamètre est considéré comme une mesure importante.
- De même, une faible distance moyenne du NoC peut avoir un impact fort sur la latence moyenne de la communication sur puce.
- Un degré moyen élevé des nœuds permet d'avoir des communications rapides entre les voisins proches. Toutefois, plus le degré moyen est faible plus les coûts matériel sont réduits et la capacité d'extension du NoC est augmentée.
- Pour fournir plus de chemins entre deux sous-réseaux, et améliore donc la performance globale du réseau, il est nécessaire d'avoir une grande bissection.
- L'ajout de liens dans un NoC pourrait réduire le diamètre, d'améliorer la communication, et de réduire la latence de communication. En revanche, plus le NoC a de liens, plus le coût en surface de silicium est élevé.
- Un degré de clusterisation plus élevé indique la disponibilité des chemins de routage alternatifs afin de re-router des paquets congestionnés à un nœud donné, ce qui permet de réduire la congestion de la mémoire tampon des nœuds intermédiaires. Un degré de clusterisation plus élevé favorise aussi la communication locale.

Notre algorithme de personnalisation du réseau d'interconnexion sur puce ne prend en compte donc que les propriétés physiques de la topologie, et il les améliore afin que le NoC supporte des applications SoC avec différents modèles de trafic de communication. Parmi les caractéristiques physiques de la topologie du NoC qui nous semble les plus significatives, nous avons choisi de travailler sur la distance moyenne (notée : Dm) et le dégrée de clusterisation (noté : Cn). Plus précisément, notre stratégie vise à maximiser le degré de clusterisation (Cn) et à minimiser la distance moyenne (Dm). Donc, l'objectif est de trouver un compromis d'amélioration de ces deux mesures pour un NoC initial donné.

VII.1.1.2. Formulation du problème

Le problème de trouver un compromis d'amélioration de la distance moyenne et du dégrée de clusterisation ($Max(Cn)$ & $Min\,(Dm)$) peut être considéré comme un problème d'optimisation, Il peut être défini par la fonction de fitness suivante :

$$\text{Max}\,(dégrée\ de\ clusterisation), \text{Min}(distance\ moyenne) \quad avec\ \sum_{l\in Ls} s(l) < S \qquad (3.1)$$

Où Ls est la liste des liens qui peuvent être ajoutés au NoC de base, l est un lien sélectionné appartenant à cette liste de liens, $s(l)$ est la distance du lien sélectionné l et S est le budget initial (la distance globale des liens à insérer).

L'équation (3.1) vise à optimiser les performances physiques de la topologie dans un premier temps, indépendamment des applications qui devraient être exécutées sur le SoC basé sur ce NoC optimisé. Cette équation d'optimisation vise par ailleurs à optimiser les performances de communication de différentes applications qui peuvent être exécutées sur les SoCs basées sur ce NoC personnalisé. En effet, la formule de la fonction de fitness (3.1) peut être convertie sous la forme :

$$\text{Max}\,(débit), \text{Min}(latence), \text{Min}(énergie) \quad avec\ \sum_{l\in Ls} s(l) < S \qquad (3.2)$$

Sachant que le dégrée de clusterisation d'un nœud i est donnée par l'équation suivante :

$$Cn_i = \frac{2\,l_i}{n_i\,(n_i-1)} \qquad (3.3)$$

Où l_i est le nombre de liens entre les voisins du nœud i, et n_i est le nombre de voisins du nœud i. Ainsi, le dégrée de clusterisation moyen est obtenu par la formule suivante :

$$Cn = (\sum_{i=1}^{N} Cn_i)/N \quad (3.4)$$

Où N est le nombre total des nœuds dans le NoC.

Par ailleurs, la distance moyenne du NoC est peut être calculée par la formule :

$$Dm = \left(\sum_{i=1}^{N}\sum_{j=1}^{N} d_{ij}\right)/\left(N(N-1)\right) \quad (3.5)$$

Où d_{ij}est la distance entre le nœud i et le nœud j, et N est le nombre de nœuds dans le NoC. Plusieurs distances peuvent être considérées pour le calcul de la distance entre deux nœuds i et j, notamment le nombre de sauts entre eux ou bien la distance euclidienne.

Dans l'ensemble de notre travail, chaque topologie est modélisée par une matrice d'adjacence. La distance considérée est celle du nombre de sauts. Donc, la distance entre deux nœuds connectés est équivalente à un saut. Si deux nœuds ne sont pas directement connectés, leur distance est équivalente au nombre de sauts du plus court chemin entre eux.

VII.1.1.3. Implémentation de l'algorithme d'insertion de liens

Notre approche de personnalisation du NoC est représentée par un algorithme d'insertion de liens stratégiques. Cet algorithme a été amélioré à plusieurs reprises, nous vous présentons donc la version la plus élaborée de l'algorithme. Il convient à noter que l'approche de personnalisation proposée ici est très différente de celles qui existent dans la littérature et qui ciblent en général une application donnée spécifique et ne cherche pas à exploiter pleinement les fonctionnalités du NoC sous-jacent. Notre approche d'exploration de l'espace de conception pour la personnalisation et l'adaptation d'un NoC candidat pour un budget de ressources donné ne prend pas compte du modèle de trafic de l'application souhaitée.

A partir des deux fonctions d'optimisation (3.1) et (3.2), on a défini deux plans d'actions. En premier lieu, nous avons développé un algorithme itératif pour la recherche des liens stratégiques à insérer sur le NoC. Les liens sélectionnés doivent optimiser au maximum les propriétés physiques de la topologie du NoC, tel que diminuer la distance moyenne et augmenter le degré de clusterisation (équation 1). L'optimisation de la topologie du NoC peut avoir un impact direct sur les performances de QoS de la communication sur puce (débit, latence, énergie, etc.), tout en respectant la contrainte relative au budget en termes de ressources (liens disponibles).

En second lieu, nous avons procéder à évaluer les performances de communication (débit, à la latence, la charge des liens et à la consommation de l'énergie) du NoC optimisé. Plusieurs modèles de trafic (applications) peuvent être considérés pour cet effet, notamment les modèles *Bit-Reversal*, *Transpose*, *Shuffle*, et *Uniform*, *Hot-Spot*, etc. (cf. chapitre 1, section V.3.3). Le routage du flux de trafic d'un modèle donné est effectué par l'algorithme de routage "*Source*" associé à l'algorithme *Dijkstra* pour la sélection du plus court chemin entre la source et la destination.

L'algorithme de personnalisation de NoC présenté ici (cf. figure 3.3) permet de concevoir un réseau d'interconnexion sur puce qui peut servir comme une toile de communication pour le maximum d'applications du NoC tout en optimisant les performances de communication des applications SoC. L'algorithme est composé de plusieurs fonctions, dont les principales sont :

- Unes fonction de génération de la matrice d'adjacence pour chaque type de topologie (2D Mesh, 2D Torus, X-Mesh, WK, Spidergon, Ring, Pyramide, Bus, etc.), en fonction du nombre de nœuds souhaité (ex., 4x4, 8x8, etc.)
- Une fonction de calcul de la liste de liens à insérer Ls (si A_{ij} = 0, et $i \neq j$)
- Une fonction de calcul de la distance moyenne Dm de la topologie

- Une fonction de calcul du dégrée de clusterisation Cn de la topologie
- Une fonction de filtrage de la liste de liens Ls pour ne garder que ceux qui améliorent les performances physiques de la topologie (Cn, Dm)
- Une fonction de conversion de la matrice d'adjacence en une matrice de ports (si $A_{ij} = 1$, c'est-à-dire que le lien L_{ij} existe, alors on remplace la valeur de A_{ij} par le code décimal du port de sortie du nœud i, tel que indiqué dans le tableau 3.1)
- Des fonctions de modèles de trafic, tels que : *Shuffle*, *Bit-Reversal*, *Hot-Spot*, *Uniform*, *Multimedia*, *Butterfly*, etc. Elles donnent en sortie une liste de paires de nœuds communicants du NoC (src, dst)
- Une fonction de calcul des plus courts chemins à base de l'algorithme de *Dijkstra*.
- Une fonction pour le calcul des route-codes, qui prend en entrée la matrice des ports et le chemin de chacune des paires (src, dst)
- Une fonction qui écrit tous les paramètres nécessaires à la simulation dans des fichiers de configuration du simulateur Nirgam, notamment : la topologie, le budget, les liens à insérer, le modèle de trafic choisi, et les route-codes qui correspondent à ce modèle de trafic.

Pour choisir judicieusement les liens à insérer sur le NoC qui convient le mieux au besoin de plusieurs applications SoC, l'algorithme commence à partir d'une configuration standard initiale C_0 du NoC (ex., 2D Mesh) et les ressources disponibles S (ex., $S = 4$). Ensuite, l'algorithme passe par plusieurs étapes :

Etape 1 : lors de la première étape, la matrice d'adjacence (A) de la topologie du NoC est calculée, $A_{ij} = 1$ si et seulement s'il y a un lien entre les nœuds i et j, $A_{ij} = 0$ sinon.

Etape 2 : lors de la deuxième étape, une liste Ls de tous les liens qui pourraient être ajoutés à la configuration initiale C_0 est générée (le lien L_{ij} est ajouté à la liste Ls, si $A_{ij} = 0$ et $i \neq j$). La liste Ls est filtrée par la suite de façon à ne conserver que les liens L_{ij} qui améliorent simultanément le degré de clusterisation (Cn) et la distance moyenne (Dm). En effet, seuls les liens entre les routeurs qui ont au moins un routeur voisin en commun sont considérés (pour s'assurer d'améliorer le dégrée de clusterisation).

Etape 3 : lors de la troisième étape, les liens conservés sont testés un-par-un sur la configuration actuelle C_k, c'est ainsi que leur couple (Cn, Dm) est calculé. En suite la liste Ls sera trié par ordre décroissant, de tel sorte que les liens qui améliore le plus le couple (Cn, Dm) se trouvent en premier. Pour cela, nous avons défini une solution optimale de référence. Pour des performances physiques idéales du NoC, il faut que celui-ci soit complètement connecté. Un NoC complètement connecté offre une meilleure distance moyenne équivalente à "1" saut (un seul saut de n'importe quel nœud vers n'importe quel autre nœud), ainsi qu'un meilleur dégrée de clusterisation équivalent lui aussi à "1". Par conséquence, la solution optimale est définie parS_{opt} (Dm_{opt}, Cn_{opt}) = (1,1). Le meilleur lien est celui qui offre une solution S_i (Dm_i, Cn_i) qu'est la plus proche à la solution optimale S_{opt} (Dm_{opt}, Cn_{opt}), (cf. figure 3.2). Ainsi, nous avons calculé la distance euclidienne de chaque solution (Dm_i, Cn_i) (qui correspond au lien i, i allons de 1 à n, où n est le nombre total de liens dans Ls) par rapport à la solution optimale (Dm_{opt}, Cn_{opt}). La distance euclidienne est peut être obtenu comme suite :

$$DIS\_EUC(S_i, S_{opt}) = \sqrt{(Cn_i - Cn_{opt})^2 + (Dm_i - Dm_{opt})^2} \quad (3.7)$$

Le lien qui offre la plus faible distance euclidienne est celui qui optimise le mieux simultanément les deux paramètres Cn et Dm, à savoir, Min (Dm) et Max (Cn). En cas où plusieurs liens ont la même distance euclidienne, l'un d'entre eux sera sélectionné aléatoirement.

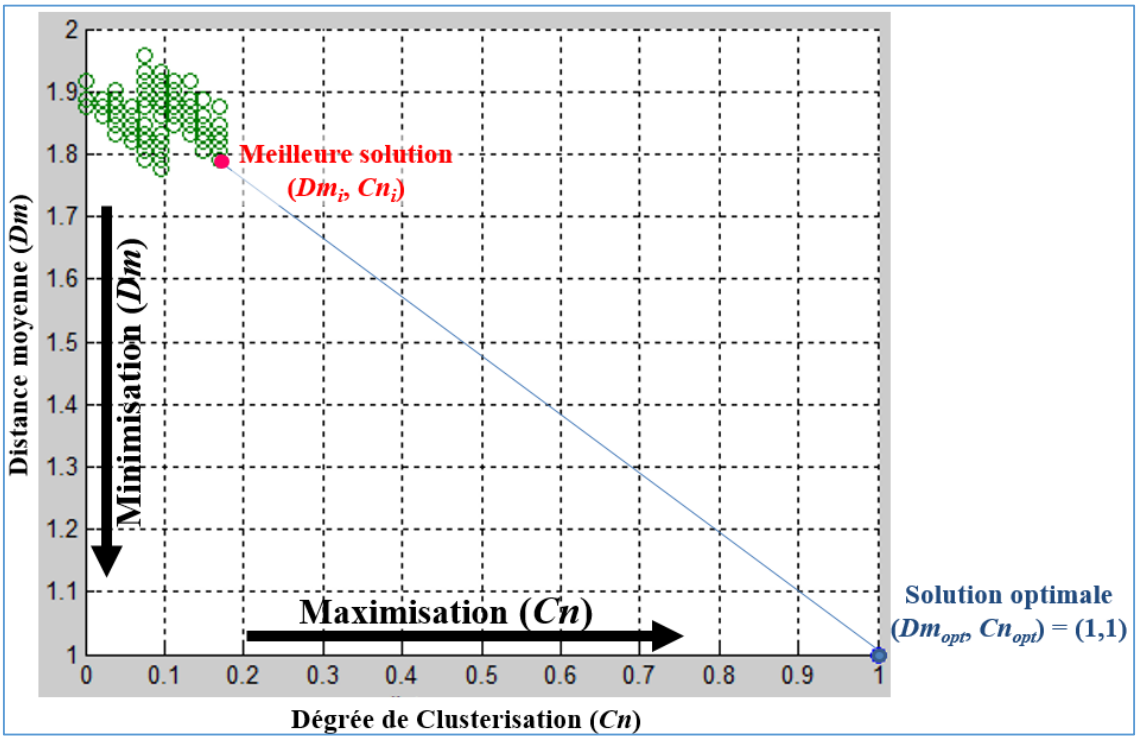


Figure 3.2 : Election de la combinaison gagnante

Etape 4 : lors de cette étape, le premier lien est inséré définitivement sur la configuration actuelle C_k pour obtenir une nouvelle configuration C_{k+1}.

Etape 5 : Tant que le budget initialement prévu n'est consommé, retour à l'***Etape 3***.

VII.1.1.4. L'algorithme d'insertion de liens

L'algorithme présenté ici est composé de deux parties principales (cf. figure 3.3). La première partie concerne la sélection et l'insertion des liens stratégiques sur le NoC de base. Une nouvelle contrainte vienne s'ajouté à la contrainte d'ores et déjà existante, il s'agit du nombre de voisins (dégrée) maximal sur chaque nœud, qui est limité à 8 voisins.

Cet algorithme peut prendre en entrée tout type d'architecture sur puce de toute taille et tout type de modèle de trafic. Pour nos évaluations et nos simulations, nous avons considéré donc cinq types de NoC (2D Mesh, 2D Torus, X-Mesh, WK et Spidergon) et quatre modèles de trafic, (Transpose, Uniforme, Bit-Reversal et Shuffle). Le calcul des route-codes pour le routage est automatique, et les paramètres de simulation sont écrits directement sur les fichiers de configuration du simulateur NIRGAM. Le simulateur NIRGAM dans sa version originale ne permet pas d'ajouter des liens, et il est basé uniquement sur l'une des topologies régulières 2D Mesh ou 2D Torus. Nous avons fait donc beaucoup de modifications sur NIRGAM afin de pouvoir simuler tout type de réseau d'interconnexion sur puce, en définissant simplement sa matrice d'adjacence.

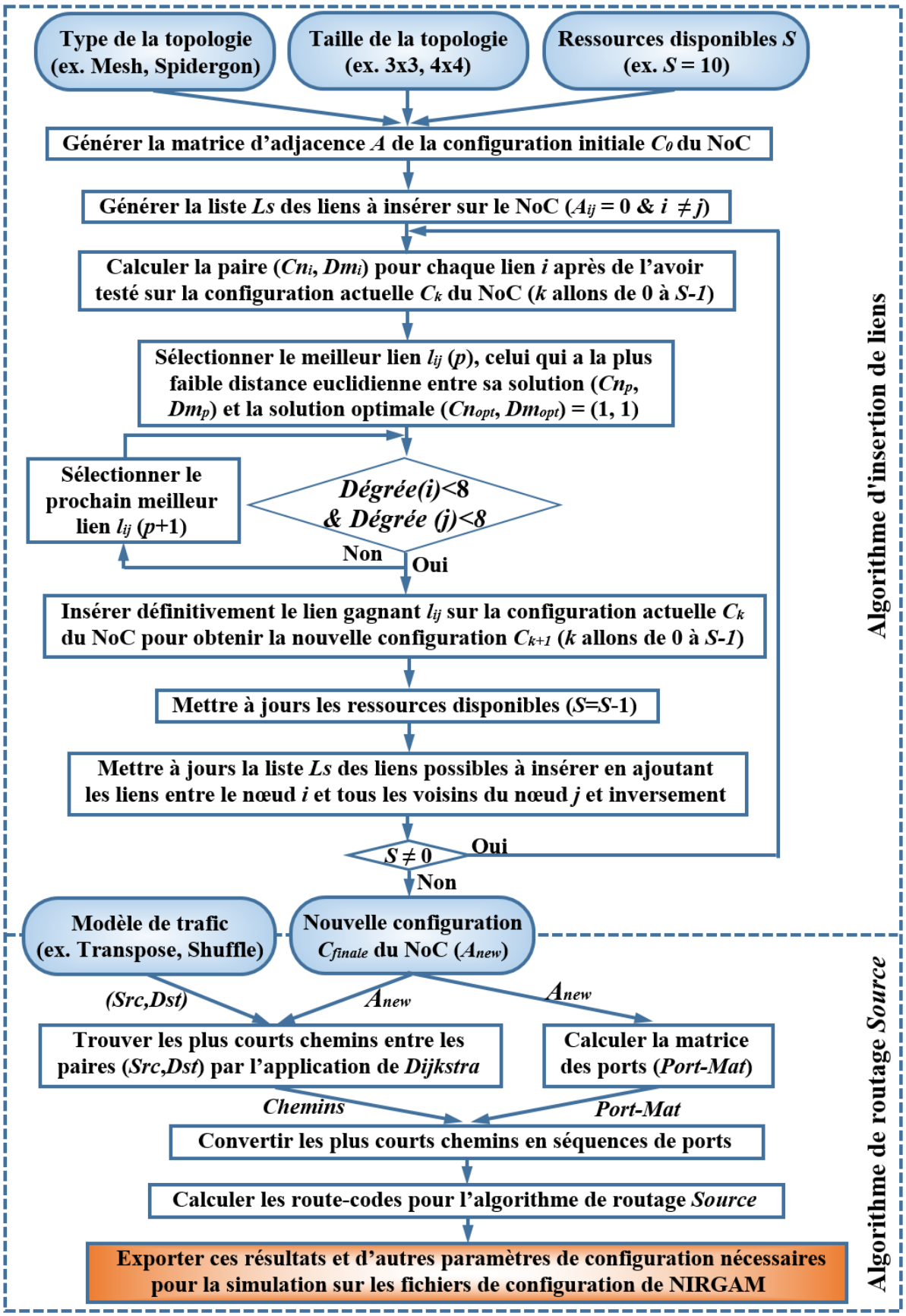


Figure 3.3 : L'algorithme de personnalisation de NoC par insertion de liens

La figure 3.4 illustre l'exécution, étape par étape, de la première partie de l'algorithme de personnalisation sur un NoC de type 2D 3x3 Mesh, avec un budget $S = 4$.

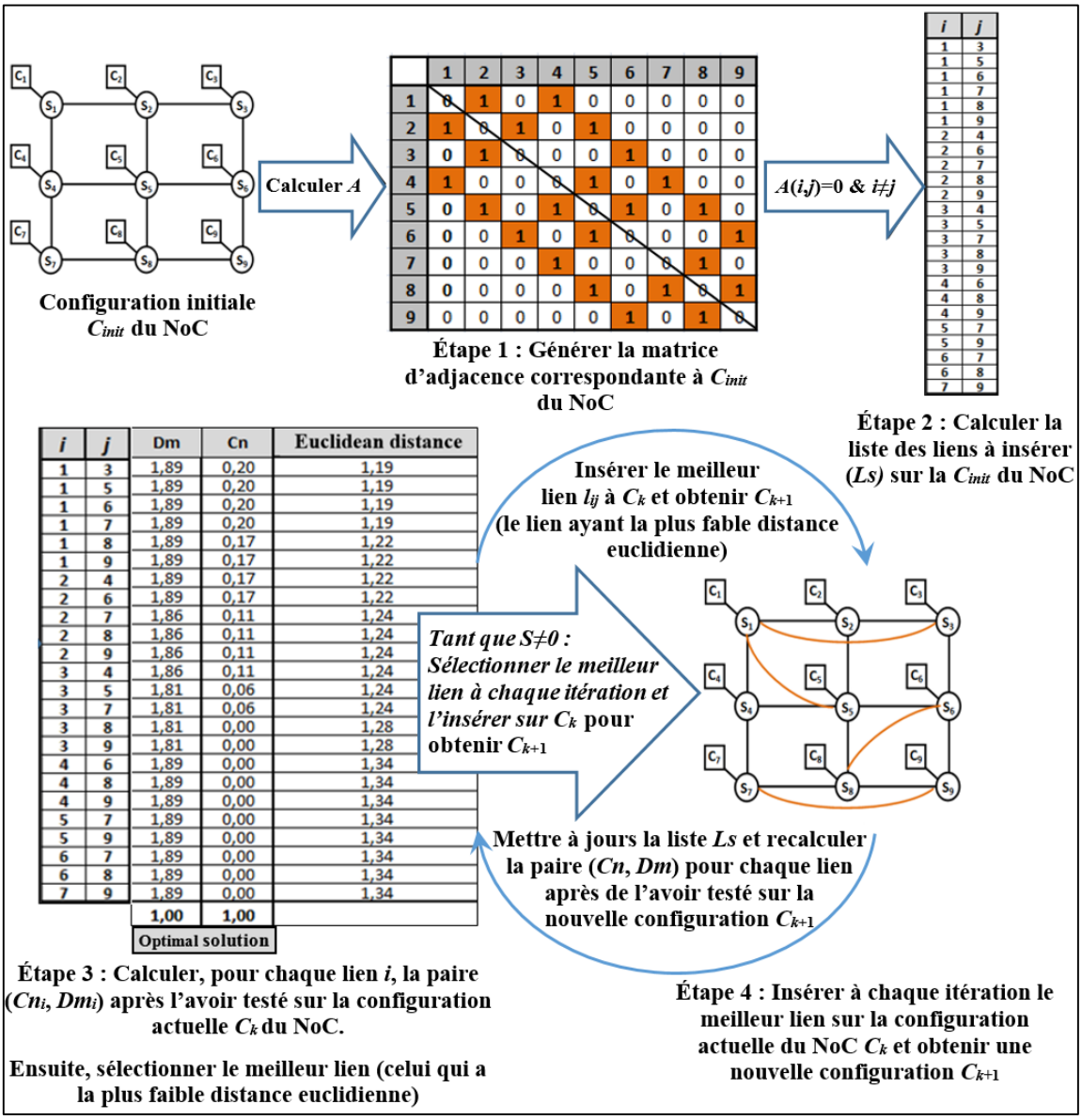


Figure 3.4 : Exemple de déroulement de l'algorithme de sélection de liens à insérer

Cet algorithme commence par une configuration NoC standard $C_{k=0}$ (2D 3x3 Mesh dans notre exemple) et les ressources disponibles ($S = 4$). Ensuite, la matrice de d'adjacence (A) qui modélise la topologie du NoC est calculée, et la liste Ls des liens qui pourraient être ajoutées au NoC est générée (24 liens sont possibles). Cette liste est filtrée afin de ne conserver que les liens qui améliorent simultanément le degré de clusterisation (Cn) et la distance moyenne (Dm). Ensuite, les paires (Dm_i, Cn_i) sont calculées pour chaque lien testé sur la configuration actuelle C_k. La solution optimale est (Dm_{opt}, Cn_{opt}) = (1,1), elle correspond au NoC complètement connecté. La distance euclidienne entre chaque solution (Dm_i, Cn_i) du lien i et (Dm_{opt}, Cn_{opt}) est calculée. La solution ayant la distance euclidienne minimale est ensuite sélectionnée et insérée à la configuration actuelle C_k pour obtenir une nouvelle configuration C_{k+1}. Ce processus est répété sous réserve des ressources disponibles et du nombre maximal de ports égal à huit dans chaque nœud, ce dernier paramètre est reconfigurable, nous pouvons augmenter le nombre de ports à chaque nœud en fonction des besoins, par exemple 16, 24, 32 ports ou plus.

La deuxième partie consiste à calculer les route-codes après l'obtention de la nouvelle configuration du NoC. Le protocole de routage *Source* est utilisé. Nous rappelons qu'il s'agit d'une technique permettant au concepteur de spécifier la route qu'un paquet doit suivre dans le NoC par un code décimal. À n'importe quel routeur, un décalage à droite de 4 bits du route-

code détermine la direction de sortie du paquet. Dans notre cas d'étude, le routage source est mis en œuvre par la lecture des 4 bits les plus à droite du route-code, puis de décaler à droite le route-code de 4 bits. Avec 4 bits il est possible d'encoder au maximum 16 ports à chaque nœud, si nous aurons besoin de plus de 16 ports, on peut encoder le nombre de ports sur 5 bits, ce qui nous permet d'encoder 32 ports à chaque nœud, etc. Dans la version originale du simulateur Nirgam, chaque nœud possède seulement 5 ports qui sont encodés sur 3 bits [77]. Pour le routage du modèle de trafic, le plus court chemin entre chaque paire de nœuds communicants (src, dst) est calculé avec l'algorithme de *Dijkstra*. Dans cette partie de l'algorithme, nous calculons les route-codes nécessaires pour l'algorithme de routage *Source* sur Nirgam, ces route-codes correspondent à la séquence de ports qui définit l'itinéraire entre les nœuds communicants représentés par la paire (src, dst). Tous les route-codes sont exportés directement dans les fichiers de configuration Nirgam.

L'évaluation analytique de notre approche (cf. section VII.1.1.6) et les résultats de simulations (cf. section VII.1.1.9) sont présentés ci-dessous. La simulation est faite par le simulateur NIRGAM qui est dédié aux réseaux NoCs (cf. chapitre 2, section VI.2.1.2.B.j). L'évaluation a été réalisée sur un large choix de topologies de base (2D Mesh, 2D Torus, Spidergon, WK et X-Mesh), un large choix de modèles de trafic (*Uniform*, *Transpose*, *Shuffle* et *Bit-Reversal*), ainsi que d'autres paramètres de simulation qui seront détaillés ci-dessous (cf. section VII.1.1.8). Les résultats de simulation obtenus montrent l'efficacité de notre approche. C'est-à-dire, que les simulations de plusieurs modèles de trafic sur des NoCs personnalisés/optimisés par l'ajout de liens montrent tous une amélioration de la performance de communication du SoC en termes de latence, de débit, de charge des liens et d'énergie consommée, comparé à un NoC de base.

VII.1.1.5. Le routage avec l'insertion de lien

L'algorithme de routage utilisé pour nos simulations est l'algorithme de routage *Source*. L'entrée nécessaire pour l'algorithme de routage *Source* est un code appelé route-code qui représente toute la trajectoire d'un paquet à partir de la source jusqu'à la destination. A chaque routeur il pourrait y avoir cinq directions possibles dans lesquelles un paquet peut être routé, à savoir *N*, *S*, *E*, *W*, et *C*. Chaque direction est représentée par un code.

Dans NIRGAM, le 2D Mesh est modélisée comme une grille de nœuds. Chaque nœud est constituée d'un noyau IP (C_i : Core i) connecté à un routeur/commutateur (S_i : Switch i) par un canal principal bidirectionnel. Ainsi, chaque nœud est connecté à ces nœuds voisins par des canaux bidirectionnels. Chaque nœud est identifié par un identifiant *ID* entier unique, représenté par une paire de coordonnées (x, y), où $ID = y + (x * Nb\_Colonne)$. Ainsi, pour permettre d'interconnecter ces nœuds, les auteurs de NIRGAM ont défini trois types de nœuds (cf. figure 3.5), les nœuds de coin appelés "*Corner-Tile*", les nœuds de bord appelés "*Border-Tile*" et les nœuds du centre appelés "*Generic-Tile*". Ces trois types de nœuds ont des caractéristiques différentes et propres à chaque type, notamment le nombre de voisins (le dégrée du nœud). Par exemple, le nœuds de type "*Corner-Tile*" a que deux voisins, le nœud de type "*Border-Tile*" a trois voisins, et le nœud de type "*Generic-Tile*" a quatre voisins.

Par conséquence, l'insertion de nouveaux liens sur un nœud de type "*Generic-Tile*" n'était pas possible (le cas du lien l_{ij} = (1,5) dans l'exemple de la figure 3.4), vu que le dégrée max de quatre voisins était atteint. C'était donc ça notre premier obstacle rencontré.

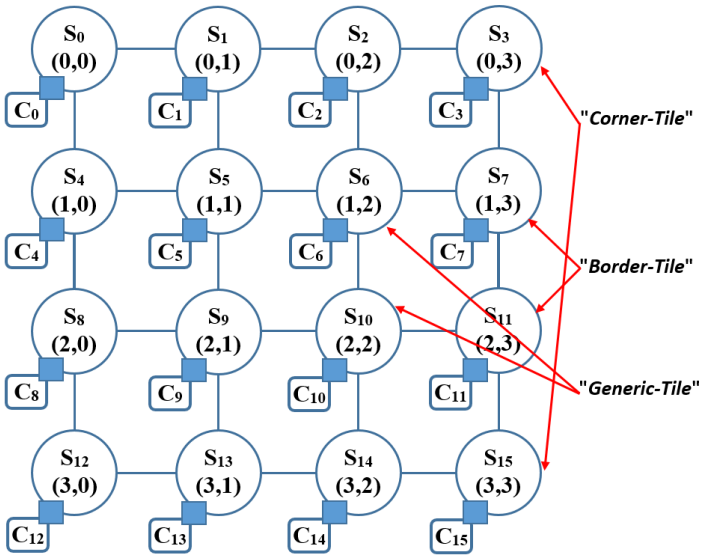
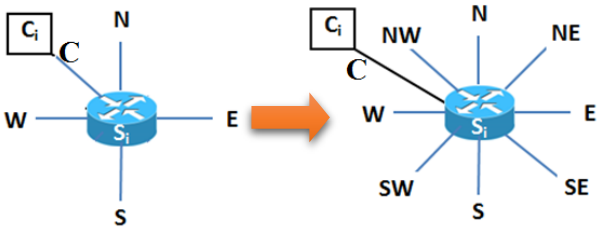


Figure 3.5 : Types de nœuds dans NIRGAM

Dans la version originale de NIRGAM, il n'y a que quatre ports bidirectionnels possibles de sortie (*N*, *S*, *E* et *W*) pour chaque nœud vers les nœuds voisins, et un port (*C*) vers la ressource IP locale. Pour pouvoir ajouter des liens, nous avons défini quatre ports supplémentaires. La figures 3.6.a montre la structure du commutateur/routeur dans la version originale de NIRGAM et la figure 3.6.b montre le commutateur/routeur étendu par l'ajout de quatre ports supplémentaires (*NW*, *NE*, *SW* et *SE*) dans la version de NIRGAM que nous avons amélioré. Ainsi, il est devenu possible de connecté le nœud 1 au nœud 5 dans le cas de notre exemple de la figure 3.4 via le port *SE* du nœud 1 et le port *NW* du nœud 5.



(a) **(b)**

Figure 3.6 : La structure du commutateur, (a) dans la version originale de NIRGAM, (b) dans la version personnalisée de NIRGAM

A chaque routeur il pourrait y avoir donc neuf directions possibles dans lesquelles un paquet peut être routé, à savoir *N*, *S*, *E*, *W*, *NW*, *NE*, *SW*, *SE* et *C*. Chaque direction est représentée par un code comme illustré dans le tableau 3.1. La représentation binaire de chaque port est faite sur 4 bits, puisque si on utilise 3 bits, on ne peut représenter que 8 ports, comme illustré dans le tableau 3.1. La représentation binaire de chaque port est faite sur 4 bits.

Les ports	Le code décimal	Le code binaire
North (N)	0	0000
South (S)	1	0001
East (E)	2	0010
West (O)	3	0011
North-West (NW)	4	0100
North-East (NE)	5	0101
South-West (SW)	6	0110
South-East (SE)	7	0111
Core (C)	8	1000

Tableau 3.1 : Les codes de ports pour l'algorithme de routage "*Source*"

Donc, à chaque routeur traversé par le paquet, un décalage à droite de 4 bits du route-code, contenu dans le paquet lui-même, détermine la direction de sortie de ce paquet. Le routage source est mis en œuvre par la lecture des 4 bits les plus à droite du route-code, puis de décaler à droite le route-code par 4 bits [77].

Par exemple, on considérant une architecture d'interconnexion sur puce de type 2D 4x4 Mesh, comme le montre la figure 3.7. Pour envoyer un paquet du nœud S_3 vers le nœud S_9 (le flux f_2) tout en utilisant le lien inséré entre S_3 et S_1. Le flux f_2 va être acheminé le long du chemin : $S_3 \rightarrow S_1 \rightarrow S_5 \rightarrow S_9$. Ce chemin doit être converti en une séquence de direction : $NW \rightarrow S \rightarrow S \rightarrow C$. Le code de route peut être obtenu en écrivant les codes de directions de ces directions dans le sens inverse, tout en conservant l'ordre de 4 bits dans chaque code de direction. Ainsi, le code de route est : $NW\ (0100) + S\ (0001) + S\ (0001) + C\ (1000) = 1000\ 0001\ 0001\ 0100\ =\ 33044$.

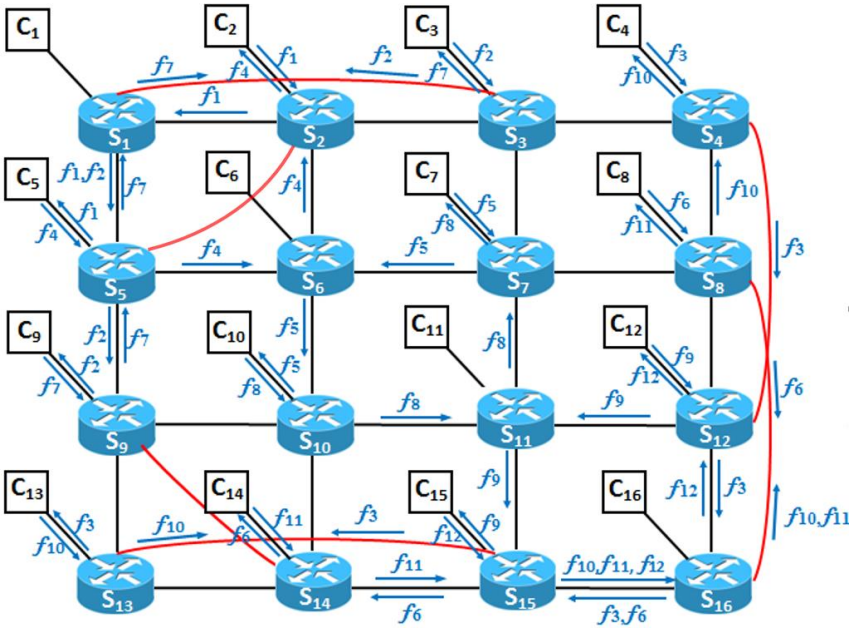


Figure 3.7 : Le NoC de type 2D 4x4 Mesh avec les flux de données entre les nœuds communicants

La deuxième partie de l'algorithme d'insertion de lien, illustré par la figure 3.3, est dédiée au calcul des plus courts chemins entre les nœuds communicants (src, dst), définis par un modèle de trafic, et le calcul des route-codes leur correspondants pour servir au routage par l'algorithme "*Source*". Donc, le protocole de routage *Source* est associé à l'algorithme de sélection du plus court chemin *Dijkstra*. Les résultats de ces calculs sont écrits directement sur les fichiers de configuration du simulateur NIRGAM.

VII.1.1.6. L'évaluation analytique de l'algorithme d'insertion de lien

Pour montrer l'intérêt de notre algorithme, nous l'avons évalué analytiquement. Pour ce faire, nous avons pris, d'une part, des liens aléatoires à insérés, et d'autre part des liens sélectionnés par notre algorithme, avec bien sur le même nombre total de liens ajoutés. Un NoC de type 2D 4x4 Mesh a été utilisé. Les résultats obtenus sont illustrés jusqu'à un budget de $S = 50$ liens supplémentaires (sans oublier les 24 liens de base du 2D 4x4 Mesh) dans le tableau 3.2.

Liens insérés	NoC de type 2D 4x4 Mesh personnalisé par l'ajout de liens sélectionnés par notre algorithme						NoC de type 2D 4x4 Mesh personnalisé par l'ajout de liens sélectionnés aléatoirement					
	Dégrée de Clusterisation (Cn)	Distance moyenne (Dm)	Distance Euclidienne	Lien i	Lien j	Pourcentage d'amélioration	Dégrée de Clusterisation (Cn)	Distance moyenne (Dm)	Distance Euclidienne	Lien i	Lien j	Pourcentage d'amélioration
S = 0	0,00	2,67	1,94	/	/	0,00%	0,00	2,67	1,94	/	/	0,00%
S = 1	0,03	2,48	1,77	1	9	8,56%	0,05	2,53	1,81	6	11	7,08%
S = 2	0,05	2,33	1,64	9	11	15,61%	0,05	2,38	1,68	5	12	13,54%
S = 3	0,14	2,26	1,53	9	12	21,35%	0,07	2,33	1,62	9	11	16,61%
S = 4	0,25	2,22	1,43	2	7	26,37%	0,10	2,26	1,55	1	9	20,40%
S = 5	0,35	2,19	1,36	1	4	30,12%	0,10	2,12	1,43	4	13	26,21%
S = 6	0,38	2,13	1,29	6	14	33,70%	0,10	2,04	1,38	3	16	29,17%
S = 7	0,41	2,07	1,22	9	15	37,09%	0,11	2,01	1,35	10	12	30,75%
S = 8	0,41	1,99	1,15	2	9	40,53%	0,11	1,98	1,32	2	8	32,02%
S = 9	0,48	1,96	1,09	9	14	43,78%	0,16	1,96	1,28	11	14	34,35%
S = 10	0,53	1,91	1,02	4	9	47,24%	0,17	1,93	1,24	1	13	36,20%
S = 11	0,51	1,83	0,97	3	9	50,23%	0,19	1,90	1,21	2	13	37,80%
S = 12	0,52	1,77	0,90	0	9	53,50%	0,23	1,88	1,17	5	10	39,72%
S = 13	0,57	1,72	0,83	9	7	56,99%	0,22	1,84	1,15	1	16	40,89%
S = 14	0,63	1,68	0,78	6	9	59,95%	0,21	1,83	1,14	8	9	41,22%
S = 15	0,67	1,68	0,75	11	14	61,28%	0,22	1,80	1,12	10	16	42,39%
S = 16	0,70	1,67	0,73	6	11	62,31%	0,28	1,78	1,06	2	4	45,30%
S = 17	0,72	1,66	0,71	8	13	63,15%	0,28	1,76	1,04	6	15	46,28%
S = 18	0,74	1,65	0,70	1	6	63,85%	0,29	1,72	1,01	3	10	47,98%
S = 19	0,75	1,64	0,69	4	6	64,51%	0,28	1,69	1,00	1	7	48,50%
S = 20	0,76	1,63	0,68	3	6	65,12%	0,28	1,68	0,99	2	14	49,17%
S = 21	0,77	1,63	0,67	6	15	65,65%	0,29	1,67	0,97	8	10	50,06%
S = 22	0,78	1,62	0,66	10	15	66,23%	0,31	1,66	0,95	6	9	51,03%
S = 23	0,79	1,61	0,64	6	0	66,76%	0,31	1,65	0,95	3	9	51,25%
S = 24	0,80	1,60	0,63	0	5	67,34%	0,29	1,63	0,94	4	11	51,41%
S = 25	0,80	1,59	0,63	8	14	67,74%	0,32	1,62	0,92	3	11	52,76%
S = 26	0,81	1,58	0,61	12	14	68,38%	0,35	1,61	0,89	3	6	54,07%
S = 27	0,81	1,58	0,60	6	8	68,84%	0,36	1,60	0,88	6	12	54,86%
S = 28	0,81	1,57	0,60	6	13	69,24%	0,39	1,59	0,85	3	8	56,40%
S = 29	0,83	1,56	0,59	6	12	69,84%	0,39	1,58	0,84	4	15	56,75%
S = 30	0,82	1,55	0,58	1	7	70,22%	0,39	1,57	0,84	8	15	56,98%
S = 31	0,83	1,54	0,57	1	3	70,77%	0,39	1,55	0,82	6	13	57,66%
S = 32	0,83	1,53	0,56	1	8	71,12%	0,41	1,54	0,80	9	12	58,79%
S = 33	0,83	1,53	0,55	0	8	71,54%	0,43	1,53	0,78	1	10	59,74%
S = 34	0,84	1,52	0,54	5	8	72,11%	0,45	1,53	0,76	9	16	60,73%
S = 35	0,84	1,51	0,53	1	11	72,46%	0,46	1,52	0,75	1	12	61,63%
S = 36	0,84	1,50	0,53	1	14	72,88%	0,48	1,51	0,73	11	13	62,59%
S = 37	0,84	1,49	0,52	1	10	73,28%	0,48	1,50	0,72	7	16	62,90%
S = 38	0,85	1,48	0,51	1	15	73,84%	0,50	1,49	0,70	1	11	63,77%
S = 39	0,84	1,48	0,50	1	13	74,22%	0,51	1,48	0,69	10	13	64,61%
S = 40	0,85	1,47	0,49	1	12	74,76%	0,51	1,47	0,68	5	8	65,16%
S = 41	0,85	1,46	0,48	7	14	75,12%	0,53	1,46	0,66	7	9	66,19%
S = 42	0,85	1,45	0,47	7	15	75,56%	0,54	1,45	0,64	7	15	67,00%
S = 43	0,86	1,44	0,46	7	10	76,10%	0,57	1,44	0,62	1	6	68,22%
S = 44	0,85	1,43	0,46	2	14	76,44%	0,58	1,43	0,61	4	10	68,78%
S = 45	0,86	1,43	0,45	3	14	76,94%	0,58	1,43	0,60	5	16	69,16%
S = 46	0,86	1,42	0,44	2	10	77,30%	0,60	1,42	0,58	2	11	70,28%
S = 47	0,86	1,41	0,43	3	10	77,79%	0,61	1,41	0,56	9	15	70,99%
S = 48	0,86	1,40	0,42	3	11	78,22%	0,61	1,40	0,56	4	12	71,14%
S = 49	0,87	1,39	0,41	2	11	78,75%	0,60	1,39	0,56	7	14	71,19%
S = 50	0,88	1,38	0,40	2	15	79,25%	0,61	1,38	0,55	9	14	71,91%

Tableau 3.2 : Sélection de liens à ajouter, par notre algorithme et aléatoirement

Les figures 3.8 et 3.9 illustrent l'évolution du degré de clusterisation et de la distance moyenne, respectivement, en fonction des liens ajoutés jusqu'à ce que le NoC soit complètement connecté (un 2D 4x4 Mesh ($N = 16$ nœuds) a initialement 24 liens, pour le rendre complétement connecté ($\frac{N(N-1)}{2} = 120$ liens, avec $N = 16$), il faut lui ajouter $120 - 24 = 96$ liens). Nous pouvons constater aussi que le NoC idéal est obtenu lorsque le couple (Dm, Cn) atteint la valeur $(1, 1)$, c'est à dire, tous les nœuds sont entièrement connectés.

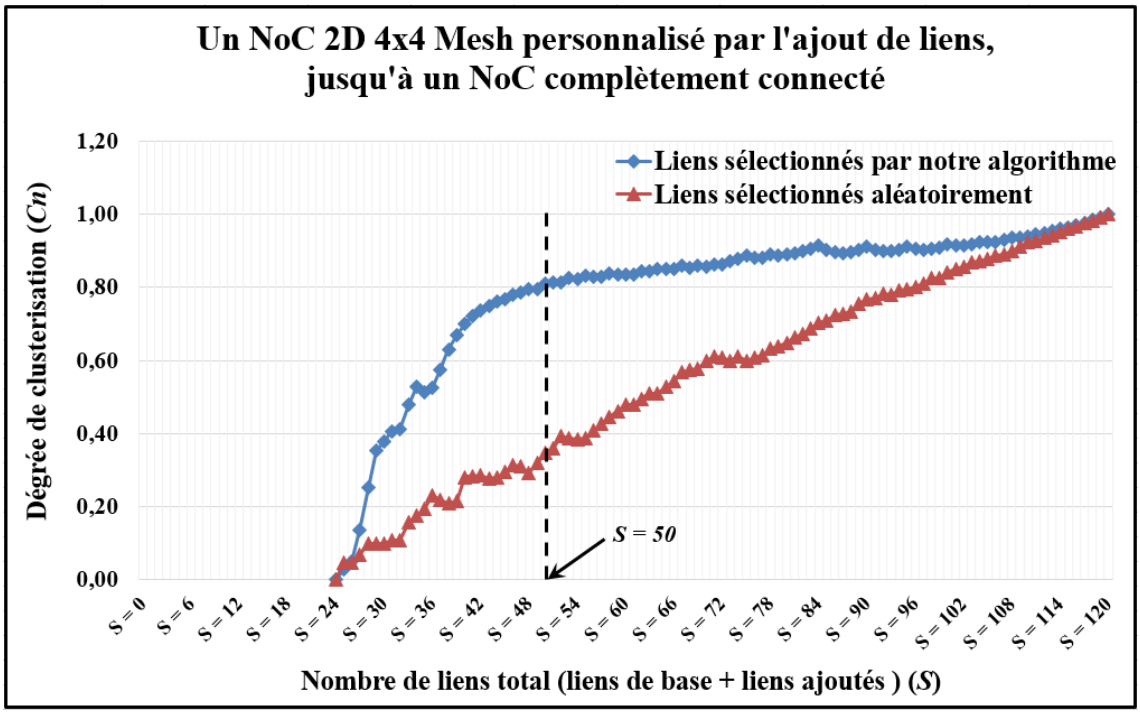


Figure 3.8 : Le degré de clusterisation du NoC de type 2D 4x4 Mesh, lorsque *S* varie de 0 à 120

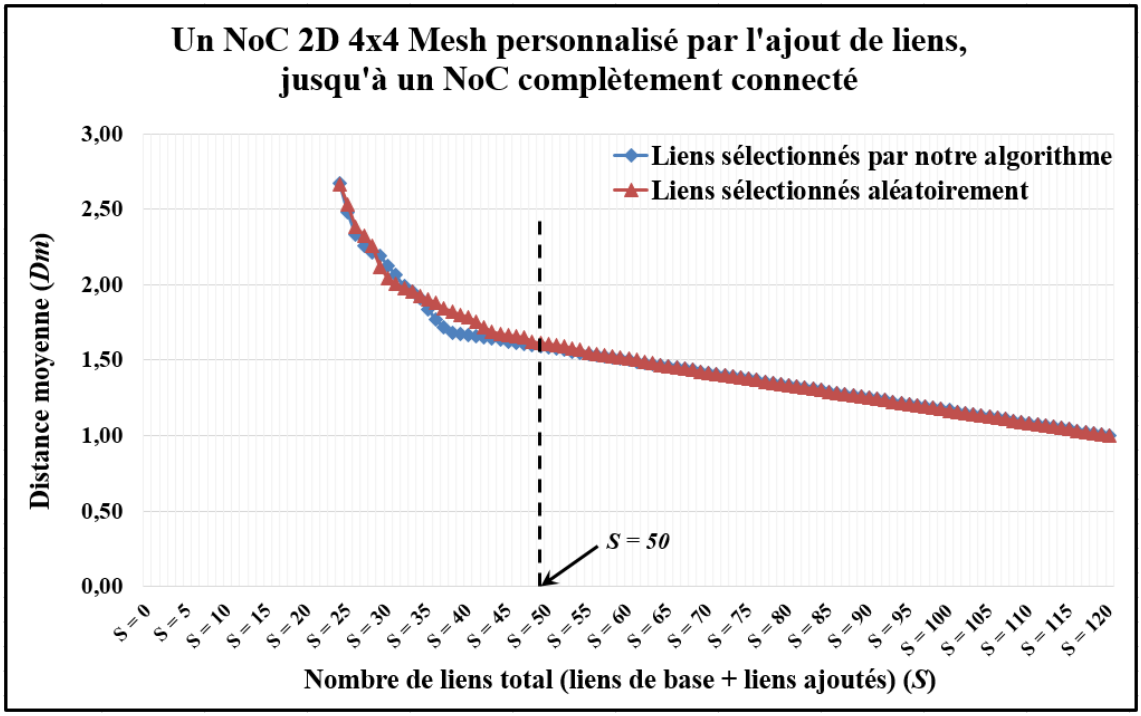


Figure 3.9 : La distance moyenne du NoC de type 2D 4x4 Mesh, lorsque *S* varie de 0 à 120

La figure 3.8 montre clairement que notre algorithme de sélection de liens utilise en premier lieu les meilleurs liens, ce qui lui permet d'être très efficace en améliorant au maximum les performances physiques du NoC avec très peu de liens insérés. Sur cette figure, sachant que le NoC 2D 4x4 Mesh a initialement 24 liens de base, nous pouvons voir que l'amélioration majeure est constatée dans l'intervalle [24,50 liens], c.à.d. avec l'ajout de 26 liens supplémentaires (ce qui représente 27% du budget initial de 96 liens), nous pouvons voir une amélioration des performances physiques de l'architecture (Dm, Cn) d'environ 68% (cf. tableau 3.2, ligne $S = 26$, colonne de pourcentage d'amélioration obtenu par notre algorithme). Après ce cap, l'amélioration de la distance moyenne et du degré de clusterisation stagne et prend une allure linéaire (cf. courbe Blue des figures 3.8 et 3.9). Ces résultats montrent que l'amélioration des performances physiques du NoC de type 2D 4x4 Mesh est très grande dans le cas de notre

algorithme d'insertion de lien que dans le cas où les liens sont insérés aléatoirement, ce qui montre l'efficacité de notre algorithme qui cible les liens les plus bénéfiques à la topologie du NoC.

Nous avons analysé l'évolution des paramètres physiques (Dm, Cn) de six architectures différentes, vis-à-vis le nombre de liens ajoutés, afin de les comparer entre elles et de déterminer laquelle des topologies évolue le mieux avec l'ajout de liens. Nous avons commencé avec les architectures 2D Mesh, 2D Torus, WK, Spidergon et X-Mesh, ayant à la base un nombre de liens équivalent à 24, 32, 30, 24, 42 et 33 liens respectivement (cf. figures 3.10, 3.11, 3.12 et 3.13). Nous avons mesuré leurs paramètres physiques (Dm, Cn) lors de l'ajout de liens supplémentaires sur leur configuration de base.

Sur le figures 3.10 et 3.11 nous avons considéré une contrainte du nombre maximal de ports à chaque nœud égal à "8". En respectant cette contrainte, les nœuds du NoC se saturent rapidement et le nombre maximal de liens devient limité, pour le X-Mesh, le WK, et le 2D Torus par exemple, le nombre de liens maximal peut seulement atteindre 59, 58 et 60 respectivement.

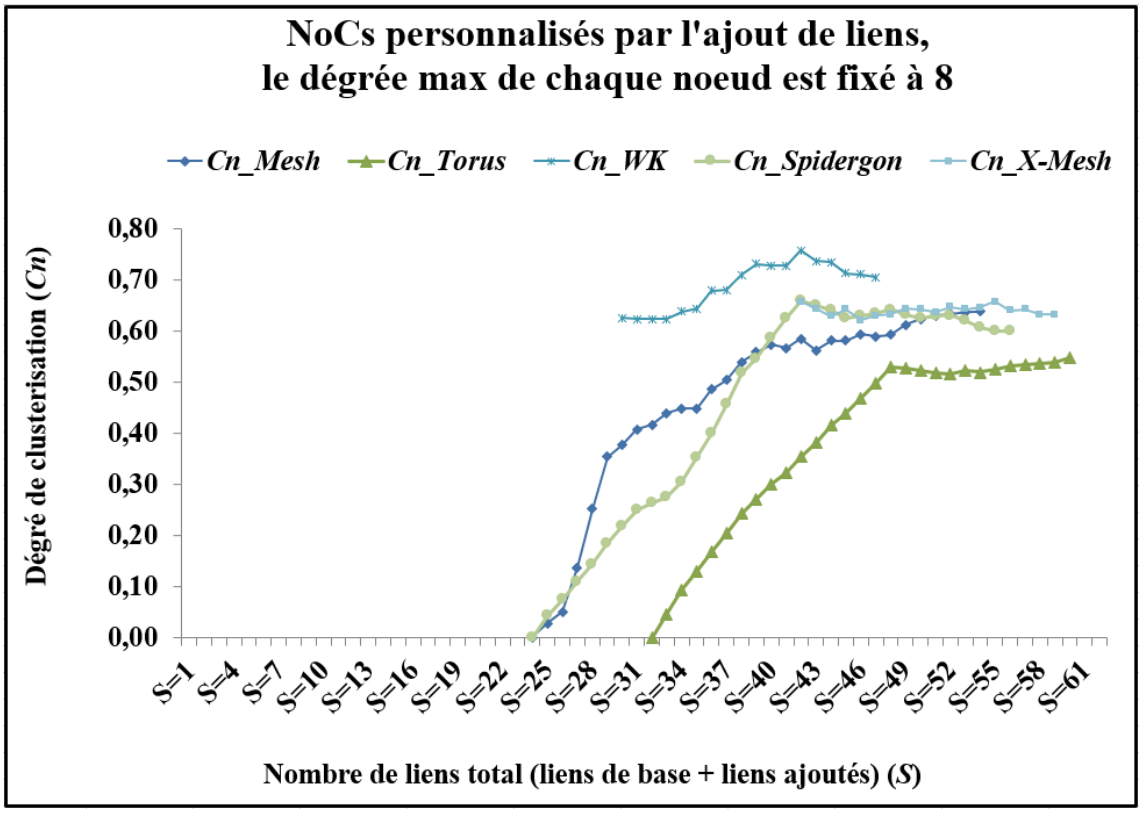


Figure 3.10 : Le degré de clusterisation pour des NoCs personnalisés par l'ajout de liens, avec le dégrée de chaque nœud fixé à 8

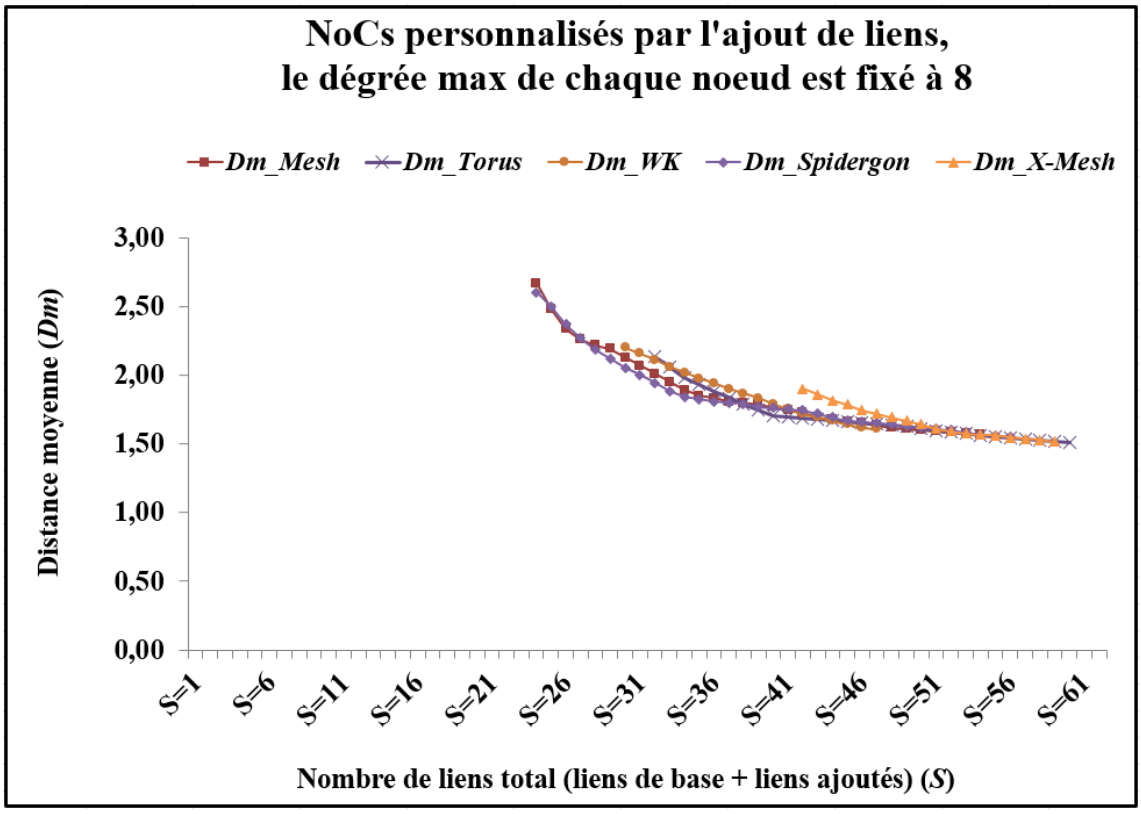


Figure 3.11 : La distance moyenne pour des NoCs personnalisés par l'ajout de liens, avec le dégrée de chaque nœud fixé à 8

Sur les figures 3.12 et 3.13 nous n'avons pas considéré cette contrainte, ce qui permet d'aller jusqu'à une architecture complètement connectée.

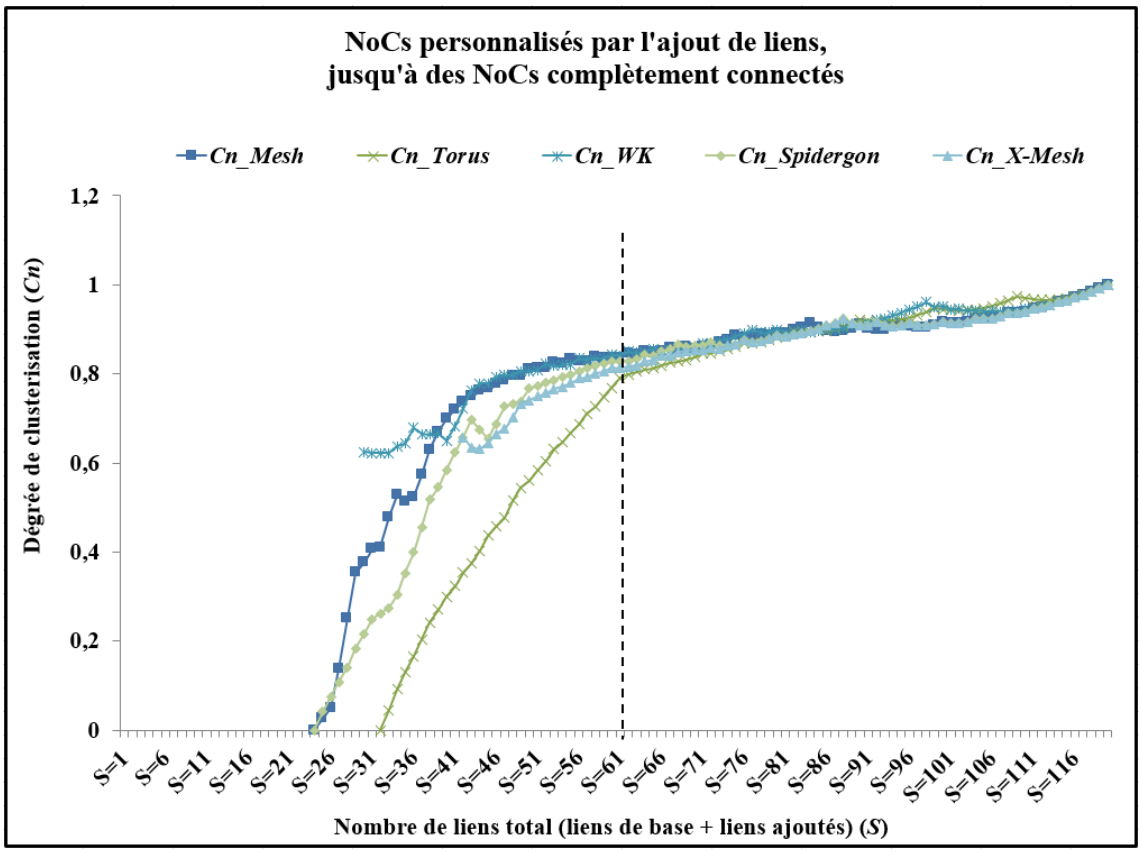


Figure 3.12 : Le degré de clusterisation pour des NoCs personnalisés par l'ajout de liens, jusqu'à des NoCs entièrement connectés

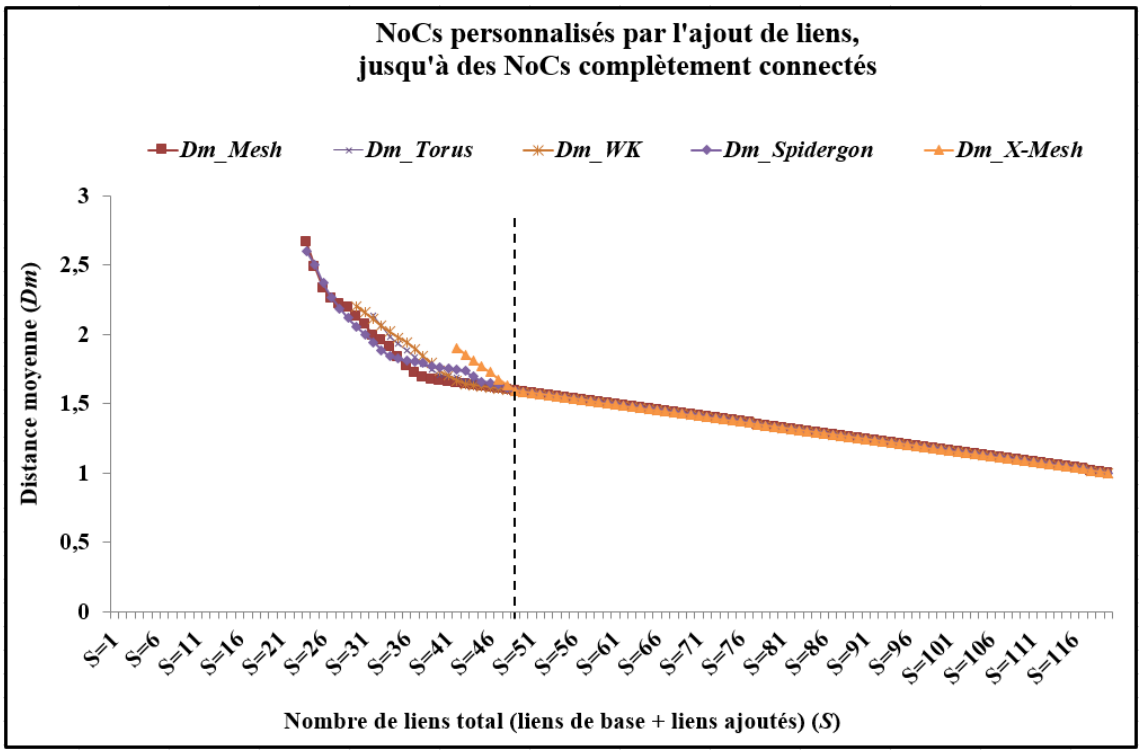


Figure 3.13 : La distance moyenne pour des NoCs personnalisés par l'ajout de liens, jusqu'à des NoCs entièrement connectés

Les figures 3.12 et 3.13 montrent que le NoC idéal est obtenu lorsque la paire (Cn, Dm) atteint la valeur (1, 1), c'est à dire, tous les nœuds sont entièrement connectés. Sur ces figures, nous pouvons voir que la plus grande amélioration est constatée dans l'intervalle [0, 60 liens], par exemple pour S = 60 liens (liens de base plus liens ajoutés, ce qui représente 50% des liens d'un NoC complètement connecté), nous pouvons voir une amélioration d'environ 70% des paramètres physiques (Dm, Cn) sur tous les NoCs. Après ce cap, ces paramètres physiques Dm et Cn diminue et augmente, respectivement, légèrement. Cela veut dire que le nombre de liens global (liens de base plus liens ajoutés) d'une quelconque topologie qui a un impact important sur ces paramètres physiques (Dm, Cn) est compris entre 30 et 60 liens. Au-delà de 60 liens, l'ajout de plus de ressource devient inintéressant, puisqu'il est couteux et n'influence pas suffisamment les paramètres physiques de la topologie (Dm, Cn).

Les figures 3.12 et 3.13 montrent aussi que la distance moyenne et le degré de clusterisation diminue et augmente respectivement davantage que des ressources sont ajoutées. Par exemple, la figure 3.12 montre que lorsque le nombre de liens des NoCs atteint 60 liens, tous les dégrées de clusterisation des différentes topologies convergent vers presque la même valeur. De même, sur la figure 3.13 on peut constater le même comportement de la distance moyenne lorsque le nombre de liens des NoCs atteint 50 liens. Par conséquent, ces résultats d'évaluation analytique montrent qu'il est possible de trouver une topologie commune au point de convergence des topologies évaluées aux alentours de $S = 60$. C'est donc de là qu'elle nous y est venue l'idée de concevoir une architecture commune plus performante qui peut remplacer tous ces NoCs. On faisant quelques recherches dans la littérature, le travail des chercheurs Samik Ghosh et al. [273] nous a particulièrement attiré, et nous avons proposé une nouvelle topologie de type fractale, dénoté FracNoC. Cette topologie sera présentée en détail dans la section suivante.

VII.1.1.7. Une nouvelle architecture fractale d'interconnexion sur puce

Les architectures d'interconnexion sur puce adoptées pour SoCs sont constituées d'un certain nombre de cœurs IPs interconnectés (ex., *CPU*: *Central Processing Unit*, *DSP*: *Digital Signal*

Processor, mémoires) qui communiquent via un réseau interconnexion. Elles sont caractérisées par différents compromis en matière de latence, de débit, de la charge de communication, de la consommation d'énergie et de besoins en surface de silicium. Dans cette section, nous introduisons une nouvelle architecture d'interconnexion sur puce en adaptant une structure de topologie fractale, elle est appelée FracNoC. Elle est conçue pour avoir un meilleur compromis de performances de communication.

La topologie FracNoC est inspiré du travail de [273], où les auteurs présentent une topologie de type 2D Mesh basé sur la géométrie fractale auto-similaire, appelée FraNtiC et destinée à la nouvelle génération de réseau d'accès radio. Cette topologie a été inspirée de la structure d'une ruche d'abeille. C'est ainsi que nous avons formulé la structure de notre topologie destinée quant à elle au réseau d'interconnexion sur puce (cf. figure 3.14).

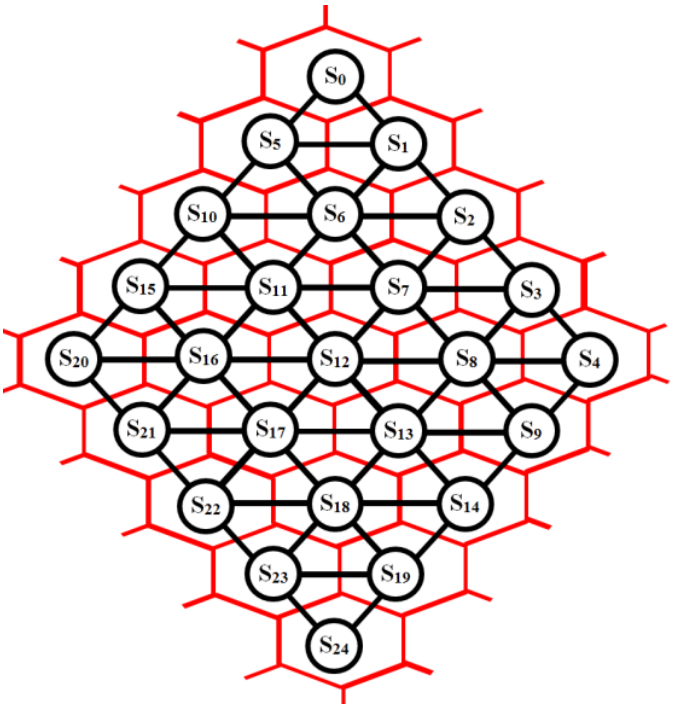


Figure 3.14 : Un NoC de type FracNoC avec *k* = 3

FracNoC est une topologie fractale, notée FracNoC(k), elle peut être décrite par le niveau d'expansion k. Elle est décrite comme suit :

- pour $k = 0$, il y a $N_0 = 0$ nœuds avec un diamètre $D_0 = 0$ et un nombre de liens $P_0 = 0$.
- pour $k = 1$, il y a $N_1 = 4$ nœuds avec un diamètre $D_1 = 2$ et un nombre de liens $P_1 = 5$.
- pour chaque $k > 1$, il y a $N_k = 4N_{k-1}$ nœuds avec un diamètre D_k=$2(D_{k-1} + 1)$ et un nombre de liens P_k=$4P_{k-1} + 13$.

Où N est le nombre total de nœuds, P_k est le nombre total de liens, et D_k est le diamètre maximum de FracNoC(k). Cette famille de topologies, commence à partir d'un FracNoC(0) et évoluer de manière récursive jusqu'à un niveau k (cf. figure 3.15).

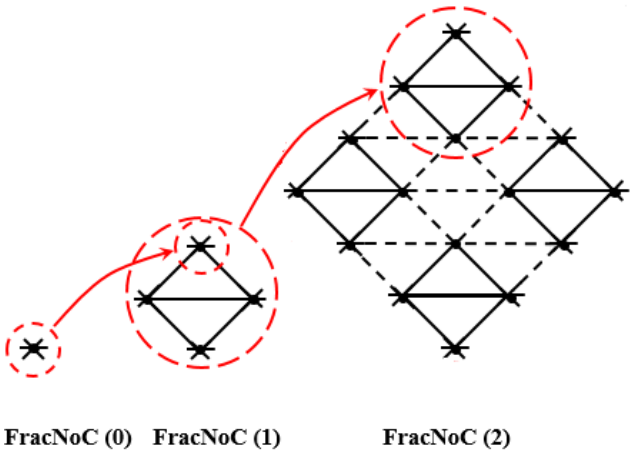


Figure 3.15 : Un NoC de type FracNoC avec *k* = 1, …, 4

Les deux plus importantes propriétés des systèmes multiprocesseurs parallèles sont l'évolutivité et la régularité. La topologie FracNoC possède toutes ces propriétés. Ainsi, le NoC de type FracNoC peut être étendu à n'importe quel niveau d'expansion sans déformer le nombre la topologie reliant les nœuds. Dans un réseau FracNoC le schéma d'indexation est basé sur la règle utilisée pour construire des nœuds de plus haut niveau à partir d'un nœud d'unité de base virtuel unique ayant *N* liaisons bidirectionnelles.

Nous avons évalué cette nouvelle architecture pour démontrer son efficacité et comparer ses performances avec ceux d'autres réseaux sur puce (2D Mesh, 2D Torus, WK, X-Mesh et Spidergon), en utilisant une variété de modèles de trafic (*Transpose*, *Bit-Reversal*, *Shuffle* et *Uniform*), et les résultats de simulation sont inclus dans l'évaluation de la troisième version de notre algorithme de personnalisation de NoC dans la section suivante. Ces résultats montrent que cette architecture fractale permet d'obtenir de meilleures performances.

VII.1.1.8. Paramètres de simulation de l'algorithme d'insertion de liens

Nous avons effectué des simulations en utilisant le simulateur NIRGAM [77]. L'objectif principal est d'évaluer l'impact de l'amélioration des performances physiques du réseau NoC sur ces performances en QoS de communication (débit, latence, charge et consommation d'énergie).

Nous avons conduit ces simulations en utilisant différents modèles de trafic avec une génération de trafic en mode bits à taux constant (*CBR*: *Constant Bit Rate*). Les principaux paramètres configurables de simulation sont détaillés dans le tableau 3.3, les paramètres considérés pour notre simulation sont en gras.

Les résultats de simulation obtenus, en utilisant six différentes architecture d'interconnexion sur puce, à savoir, 2D Mesh, 2D Torus, WK, Spidergon et X-Mesh, montrent que cette approche permet d'obtenir une meilleure performance en terme de QoS de la communication du réseau NoC optimisé, comparé au réseau NoC initial, tout en utilisant peu de budget de ressources (peu de liens insérés).

Topologie	**2D Mesh**	Torus	Ring	Spidergon
Taille de la topologie	NB_Rows & NB_Colls **(ex. 4x4)**	NB_Rows & NB_Colls (ex. 4x4)	NB_Nœuds (ex. 12, 16, 20, etc.)	NB_Rows & NB_Colls (ex.4x4)
Algorithmes de routage	> XY > Odd Even (OE) > **Routage *Source***	Routage *Source*	Routage *Source*	Routage *Source*
Méchanismes de commutation	**> Le mécanisme de commutation *Wormhole* : le paquet est décomposé en 3 types de flits: l'entête (*head*), la charge utile (*payload*) et la queue (*tail*).** > Le mécanisme de commutation de paquet (un seul flit, un spécial flit appelé : hdt)			
Générateurs de trafic	> Source (Sender), > Sink (Reciever), > **Générateurs de traffic synthétique :** > **CBR**, > Bursty, > Input trace based			
Modèles de trafic	**> Transpose**, > Hotspot, > Uniform, > DyAD, > Butterfly, > Shuffle, > Bit-Reversal, > Multimedia, > Linkfail, > Prom, > QoS			
Canaux virtuels	> L'utilisateur peut configurer le nombre de canaux virtuels par canal physique. **(ex. un VC par canal physique)**			
Tampons	> Le nombre de tampons du canal d'entré fifo peut être spécifié. **(par défault, NUM_BUFS = 32)** > Chaque tampon est d'une taille d'un flit. **(la taille du tampon est éstimée à 20, et la taile du bit du tampon est de 4)**			
Taille du flit	> Taille du flit en octets. **(ex. FLITSIZE = 8)**			
Bande passante	> Percentage de charge de la bande passante. Il détermine le pourcentage de la bande passante maximale qui puisse être utilisée. **(ex. LOAD = 100 %)**			
Flit-Intervalle	> L'intervalle entre les flits succesives en cycles d'horloge **(ex. Flit-Intervalle = 10, 15, 20, 25 ou 30, etc.)**			
Fréquence d'horloge	> Fréquence d'horloge (en GHz) peut êtree spécifiée **(par défault, la fréquence d'horloge est 1 GHz)**			
WARMUP	> Il est nécéssaire uniquement pour les générateurs de trafic synthétique > Il définit la période warmup : le nombre de cycle d'horloge pendant lesquels la génération de traffic est intérompue. **(par défault, WARMUP = 5)**			
SIM-NUM	> il définit le nombre cycle d'horloge pendant lesquels la simulation s'éxécute **(par défault, SIM-NUM = 3000)**			
TG-NUM	> Il est nécéssaire uniquement pour les générateurs de trafic synthétique > Il définit le nombre de cycle d'horloge pendant lesquels le traffic est généré **(par défault, TG-NUM = 1000)**			
Paramètres de performances	> La latence moyenne par canal par paquet (en cycles d'horloge) > La latence moyenne par canal par flit (en cycles d'horloge) **> La latence moyenne globale par flit (en cycles d'horloge)** **> Débit moyen (en Gbps)** **> L'éstimation de la consommation d'énergie (en Watt)**			

Tableau 3.3 : Les paramètres de simulation de NoC

Pour montrer l'efficacité de cette approche de personnalisation illustrée par l'algorithme de la figure 3.3, nous avons analysé les performances de communication de six architectures, notamment le débit, la latence moyenne, la charge de communication, et la consommation d'énergie, pour zéro perte de flits. Nous avons commencé avec les architectures initiales, 2D Mesh, 2D Torus, WK, Spidergon, X-Mesh et notre nouvelle architecture FracNoC, puis nous avons mesuré ces paramètres de QoS lors de l'ajout de 10, 20 et 30 liens supplémentaires sur la

configuration de base tout en respectant la contrainte du nombre maximal de ports à chaque nœud fixé à "8".

Nous avons utilisé quatre modèles de trafic, notamment *Bit_Reversal*, *Transpose*, *Shuffle*, et *Uniforme*. Ces modèles définissent la distribution de trafic des applications. Chaque simulation est exécutée pendant 3000 cycles d'horloge et un *warm-up* équivalent à 5 cycles d'horloge, le *warm-up* est défini par le nombre de cycles d'horloge pendant lesquels la génération de trafic est interrompue et elle reprend après. Le simulateur NIRGAM supporte la technique de commutation en mode trou de ver (*wormhole*). Plusieurs applications peuvent être attachées à chaque nœud dans le réseau, un nœud peut être un expéditeur (*source*) ou un récepteur (*Sink*). Dans le cas d'un nœud expéditeur, plusieurs générateurs de trafic synthétiques (*CBR*, *Bursty*, *Trace*, etc.) peuvent être attachés à ce nœud. Dans nos simulations, *le débit binaire constant* (CBR) est utilisé pour la génération du trafic synthétique avec les paramètres suivants :

- la taille des paquets (en octets)
- le pourcentage de charge (le pourcentage de la bande passante du canal à utiliser)
- la destination (nous pouvons spécifier une destination fixe, ou bien de mettre le terme "*Random*" (Aléatoire) pour que la destination soit choisie aléatoirement au moment de l'exécution. Dans le cas du protocole de routage *Source*, la destination est spécifiée par un route-code)
- l'intervalle inter-flit (l'intervalle entre les flits successives en cycles d'horloge)
- d'autres paramètres de simulation sont présentés (en gras) dans le tableau 3.3

VII.1.1.9. Résultats de simulation

Les résultats de simulation correspondants aux paramètres de performance, la latence moyenne, la charge de communication des liens, la consommation d'énergie et le débit, sont illustrés respectivement par les figures 3.16, 3.17, 3.18, et 3.19. Sur chaque figure correspondante à un paramètre de performance, nous montrons des figures correspondantes aux modèles de trafic *Bit-Reversal*, *Transpose*, *Shuffle* et *Uniform* sur la 1ère, la 2ème, la 3ème et la 4ème ligne respectivement, et d'un NoC de base ou augmenté par $S = 0$, $S = 10$, $S = 20$ et $S = 30$ sur la 1ère, la 2ème, la 3ème et la 4ème colonne respectivement.

A. La latence moyenne

La latence moyenne indique à quelle vitesse un NoC peut fournir les flits à leurs destinations, sachant qu'un NoC est considéré comme efficace si la latence est plus petite. Elle est calculée en fonction du temps moyen qui s'écoule entre le début de l'injection des flits dans le NoC depuis les cœurs de source jusqu'à leurs arrivés au cœur de destination. Elle comprend tous les retards encourus pour que les flits atteignent le nœud de destination à travers des chemins constitués d'un ensemble de liens et de commutateurs. La figure 3.16 montre la latence moyenne pour chaque modèle de trafic tout en augmentant les ressources insérées à chaque colonne. Comme le montre cette figure, lors de la diminution du flit-intervalle, le nœud de source injecte plus de flits, le réseau devient plus encombré avec un fort trafic, et donc les files d'attente deviennent saturées mettant plus de flits en attente, cela augmente donc la latence. Comme mentionné ci-dessus, un NoC avec un plus grand nombre de liens (une faible distance-moyenne) pourrait fournir une meilleure performance de communication et réduire la latence moyenne. Par exemple, quand $S = 0$, le WK, le FracNoC et le X-Mesh ont un nombre de liens plus grand comparé aux autres NoC (2D Mesh, 2D Torus et Spidergon), ce qui explique leur faible latence moyenne sans même ajouter des liens supplémentaires.

Cette figure montre que la latence moyenne diminue davantage que de ressources sont ajoutées, indépendamment du modèle de trafic utilisé. Sans l'ajout de ressources ($S = 0$), la latence était élevée en raison de la surcharge de commutation à l'intérieur des commutateurs (ex., la mise en mémoire tampon des flits, le retard dû au routage, etc.). En ajoutant des

ressources, les flits peuvent contourner certains commutateurs congestionnés, ce qui permet donc de réduire la quantité de trafic à l'intérieur de ces commutateurs. Les résultats montrent une amélioration significative lors de l'ajout de plus de ressources, ce qui confirme nos attentes constatés lors de l'évaluation analytique de l'algorithme, comme le montrent les figures 3.11 et 3.13. Nous concluons que, dans ces modèles de trafic, les simulations donnent un résultat analogue de la latence moyenne est proportionnelle à la distance moyenne et le nombre de nœuds communicants du NoC. Par exemple, après l'ajout de 10, 20 et 30 liens, le 2D Mesh montre une amélioration de 34%, 47% et 51% respectivement.

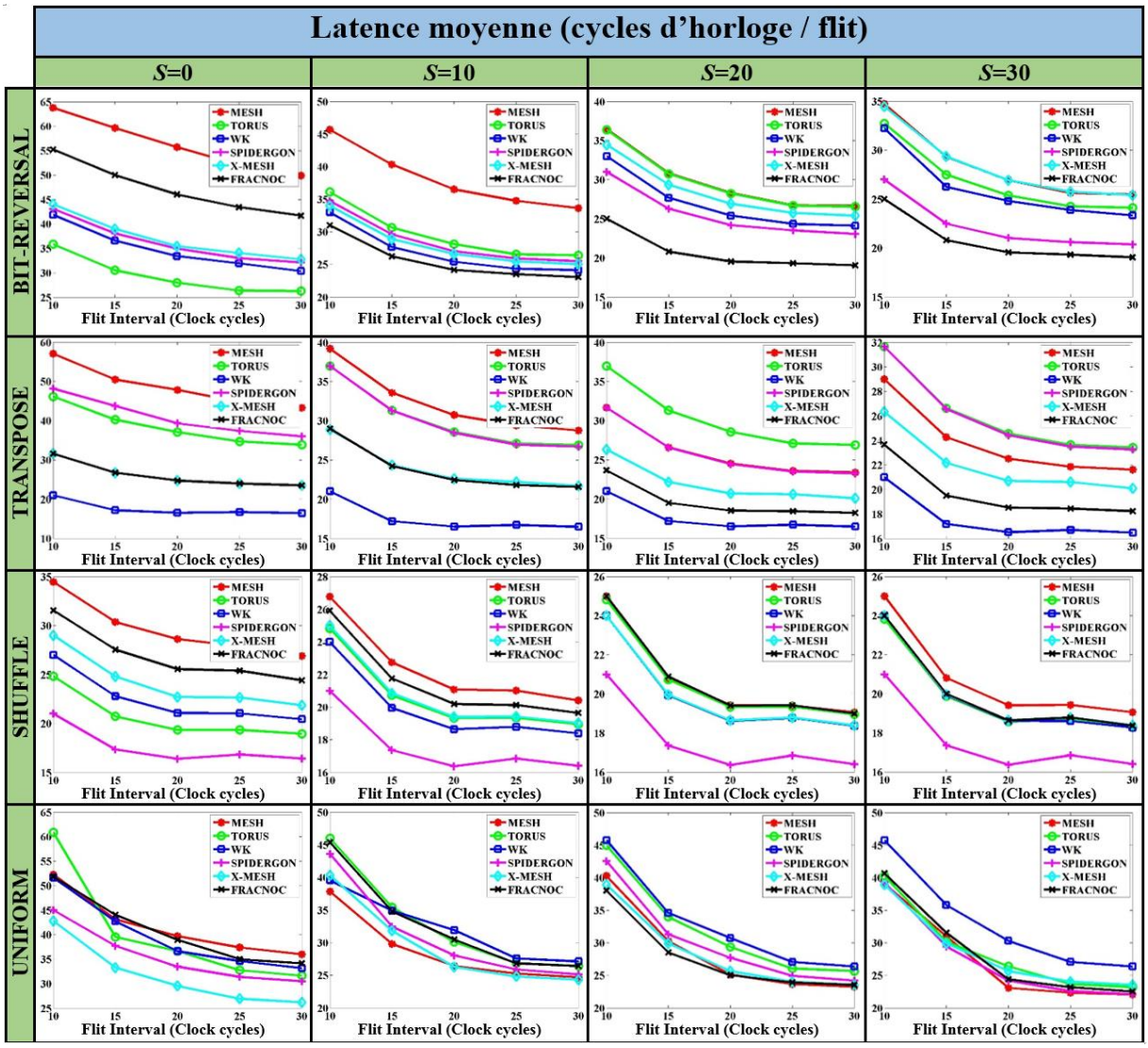


Figure 3.16 : La latence moyenne avec les modèles de trafic Bit-Reversal, Transpose, Shuffle et Uniform (par ligne) lorsque le nombre de liens insérés S égale à 0, 10, 20 et 30 liens (par colonne) et le flit-intervalle varie de 10, 15, 20, 25 à 30 (par cellule)

B. La charge de communication

La charge de communication est une mesure relative du taux d'arrivée par rapport au taux de départ sur l'ensemble des liens. Les résultats illustrés dans la figure 3.17 montrent la variation de la charge de communication sous différentes vitesses d'injection pour les NoCs augmentés avec $S = 0, S = 10, S = 20$ et $S = 30$ liens supplémentaires. Cette figure montre que la charge de communication augmente linéairement lorsque le flit-intervalle diminue (c'est-à-dire, le taux d'injection augmente). Cette augmentation de la charge peut s'expliquer par le grand nombre de flits générés à cause de la hausse du taux d'injection. En outre, les NoCs augmentés avec plus de liens, comme WK et X-Mesh, montrent une charge de communication plus basse par rapport aux autres architectures, principalement quand le trafic est dense. Cependant, comme d'autres

liens sont ajoutés, la charge de communication tend à devenir la même pour tous les NoCs. Par exemple dans le modèle *Uniforme* (cf. figure 3.17, à la 4ème ligne), les NoCs 2D Mesh, 2D Torus, WK, Spidergon, X-Mesh et FracNoC montrent une amélioration moyenne de la charge de communication de 19%, 18%, 10%, 22%, 3%, 16% respectivement. Tous les NoCs convergent vers la même charge de communication lorsque *S* égal à 30 (cf. figure 3.17, à la 4ème ligne - 4ème colonne). Le même comportement est constaté pour d'autres modèles de trafic lorsque *S* est égal à 30 (cf. figure 3.17 à la 4ème colonne). Les résultats présentés sur la figure 3.17 montrent également que les NoCs de type WK, FracNoC et X-Mesh surperforment les autres NoCs.

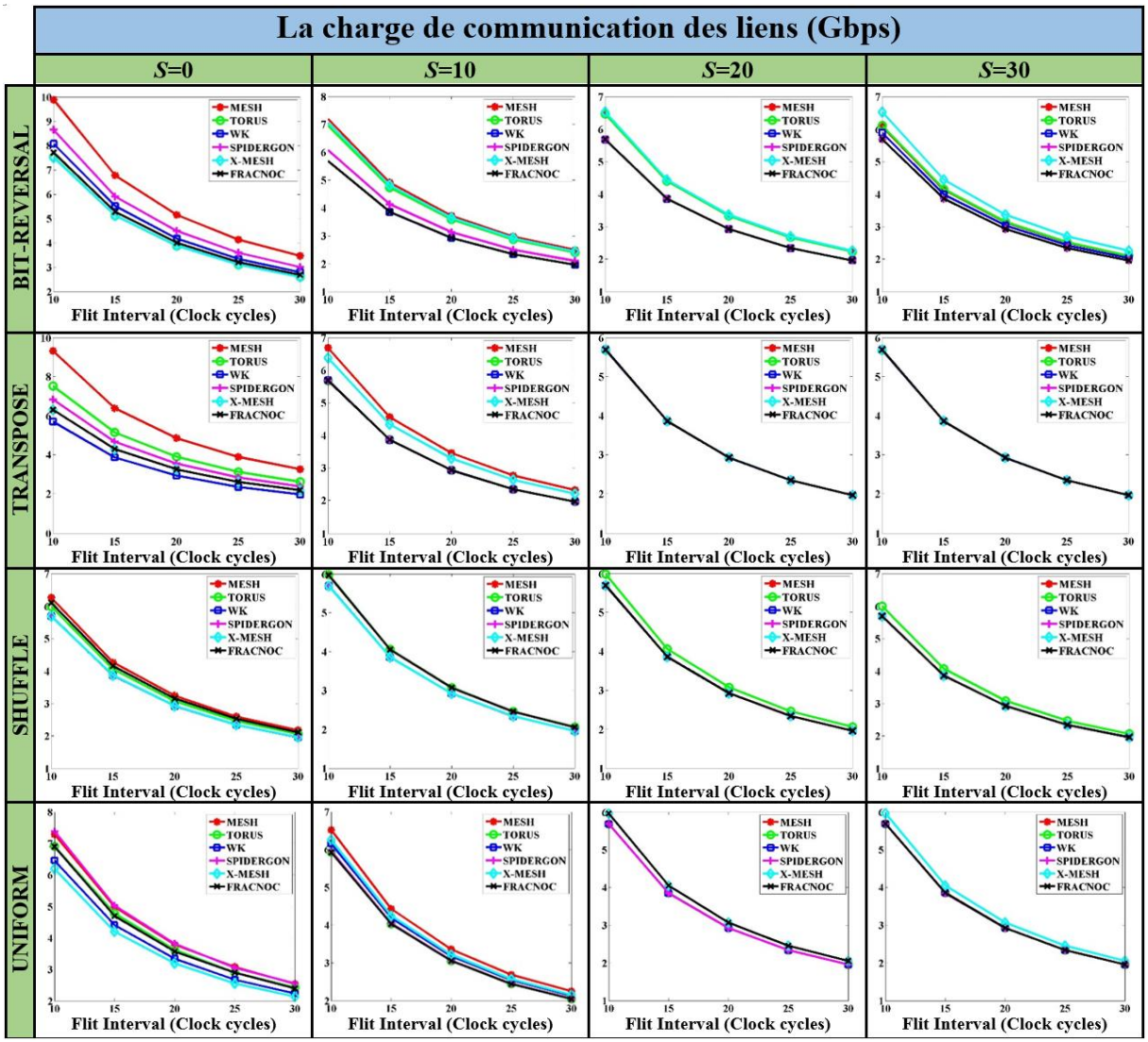


Figure 3.17 : La charge de communication des liens avec les modèles de trafic Bit-Reversal, Transpose, Shuffle et Uniform (par ligne) lorsque le nombre de liens insérés S égale à 0, 10, 20 et 30 liens (par colonne) et le flit-intervalle varie de 10, 15, 20, 25 à 30 (par cellule)

C. La consommation d'énergie

L'énergie totale peut être décomposée en énergie consommée dans les commutateurs (la traversée d'entrée et de sortie des commutateurs) et en énergie consommée par le lien tout au long de la distance parcourue par le flit. La figure 3.18 illustre la consommation d'énergie en fonction des ressources supplémentaires et la charge du trafic (flit-intervalle) pour chacun des modèles *Bit-Reversal*, *Transpose*, *Shuffle* et *Uniform*. Les résultats montrent que la consommation d'énergie décroît linéairement davantage que des ressources sont ajoutées au NoC de base ($S = 0$). Par exemple, pour le modèle *Uniforme*, les NoCs de type 2D Mesh, 2D Torus, WK, Spidergon, X-Mesh, et FracNoC montrent une amélioration moyenne de 29%, 19%, 18%, 21%, 7%, 24%, respectivement (cf. figure 3.18 à la 4ème ligne). Tous les NoCs

convergent vers des valeurs de consommation d'énergie très proches les unes aux autres (des écarts très serrés) lorsque *S* est égal à 30 (cf. figure 3.18 à la 4ème colonne).

Nous pouvons également constater que la consommation d'énergie est proportionnelle au nombre de sauts traversés par les flits et le nombre de nœuds communicants du NoC. En outre, les résultats montrent que la consommation d'énergie augmente davantage que le flit-intervalle diminue (c'est-à-dire, quand le taux d'injection augmente), en raison du grand nombre de flits générés. La figure 3.18 à la 2ème ligne indique la consommation d'énergie du modèle de *Transpose*. Sur cette ligne de la figure, pour $S = 0$, nous pouvons voir que les NoCs ayant plus de liens, comme WK, FracNoC et X-Mesh, consomment peu d'énergie car la flits traversent moins de nombre de sauts, c'est à dire, les flits contournent certains commutateurs surchargés.

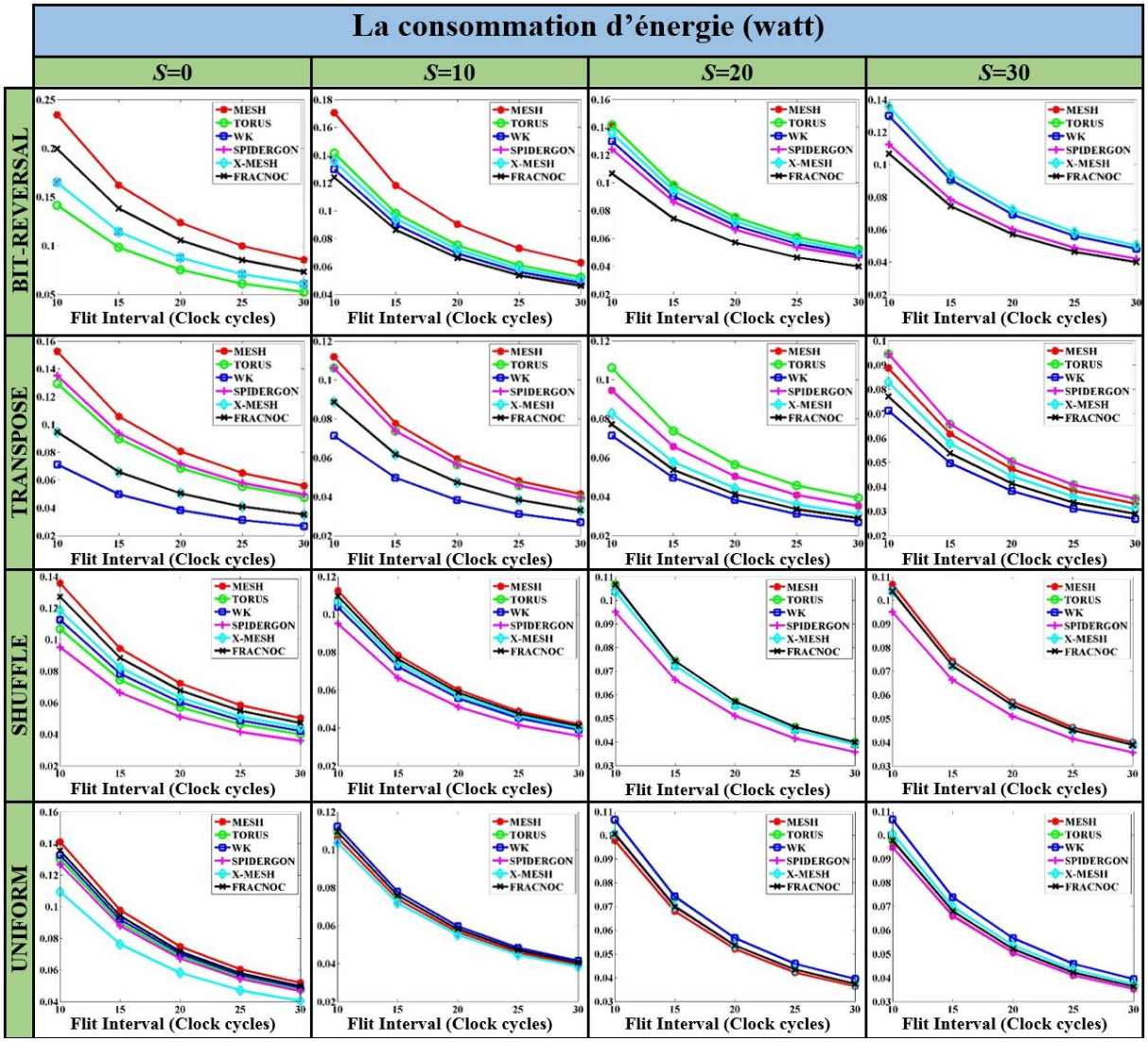


Figure 3.18 : La consommation d'énergie avec les modèles de trafic Bit-Reversal, Transpose, Shuffle et Uniform (par ligne) lorsque le nombre de liens insérés S égale à 0, 10, 20 et 30 liens (par colonne) et le flit-intervalle varie de 10, 15, 20, 25 à 30 (par cellule)

D. Le débit

Le débit pour chaque nœud représente le nombre de bits arrivé à ce nœud (Gbps). Le débit total $T(t)$ est calculé en faisant la moyenne du nombre de flits reçus avec succès par leur destination par seconde pendant le temps de simulation. La figure 3.19 illustre la variation du débit sous différents débits d'injection (flit-intervalle égal à 10, 15, 20, 25, et 30 cycles d'horloge) et pour les différents modèles de trafic. Cette figure montre que les flits injectés par des nœuds de source sont arrivés sur les nœuds de destination sans aucune perte. L'objectif ici est seulement de montrer qu'il n'y a pas eu de perte de flit, afin de faire une comparaison

équitable entre les différents NoCs et d'étudier l'influence de l'ajout de ressources sur les performances des NoCs en termes de QoS de communication avec l'utilisation de plusieurs modèles de trafic.

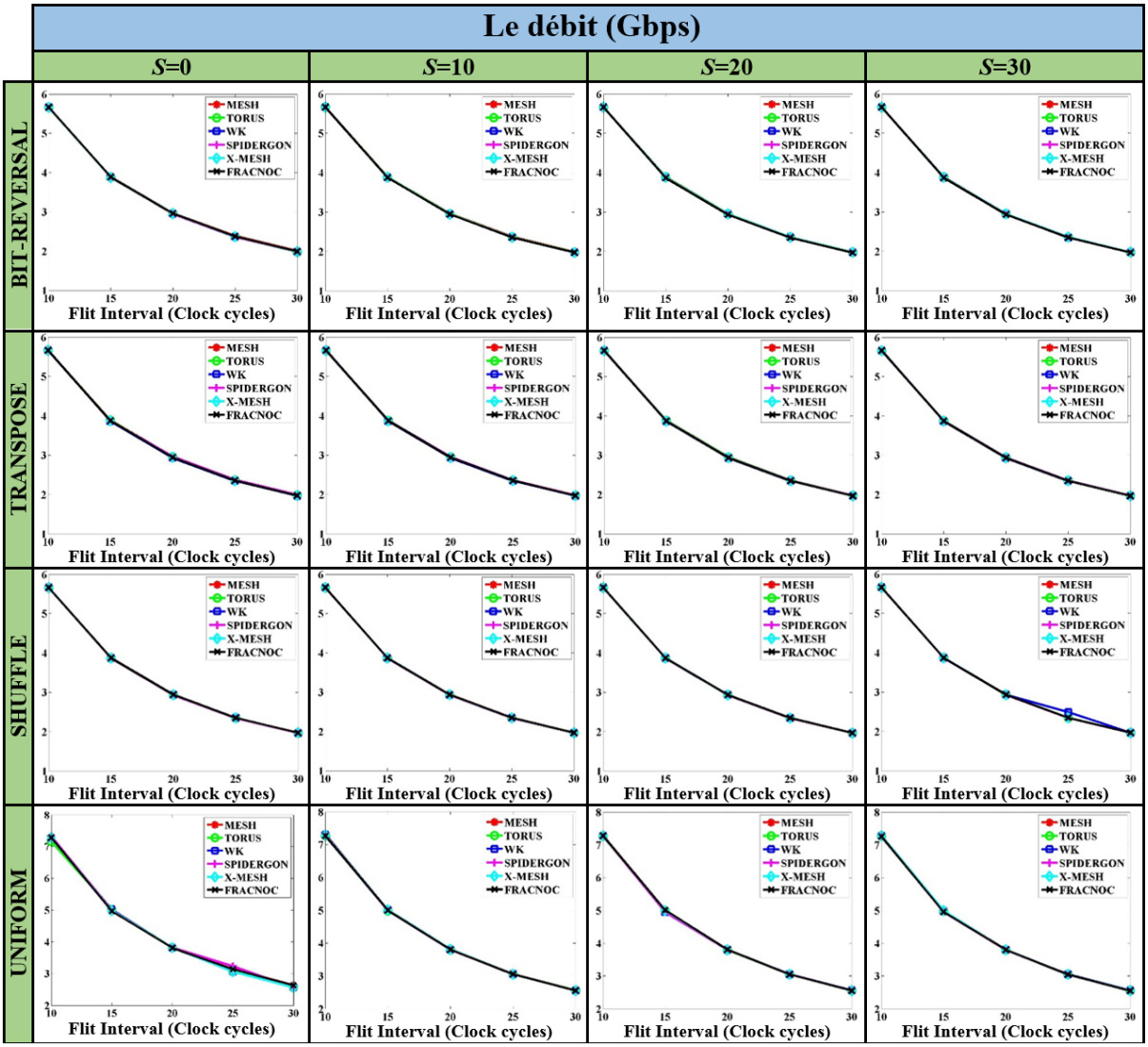


Figure 3.19 : Le débit avec les modèles de trafic Bit-Reversal, Transpose, Shuffle et Uniform (par ligne) lorsque le nombre de liens insérés S égale à 0, 10, 20 et 30 liens (par colonne) et le flit-intervalle varie de 10, 15, 20, 25 à 30 (par cellule)

VII.1.1.10. Synthèse

Dans la littérature, il n'y a pas de NoC universel qui pourrait supporter tous les modèles de trafic des applications SoC. Cependant, il a été montré que le principal problème dans la conception du NoC est la topologie du réseau. Pour améliorer sa flexibilité, il faut qu'elle soit personnalisée avec l'ajout d'un nombre minimum de liens, sans ou avec une légère modification de la structuration des nœuds, avec un petit diamètre et avec un haut degré de clusterisation pour avoir une faible latence.

Nous vous avons présenté ici une approche pour la personnalisation de l'architecture d'interconnexion sur puce par insertion de liens. L'objectif de ce travail est de construire une plateforme combinant l'approche personnalisation et l'outil d'évaluation par simulation, NIRGAM, pour la personnalisation des NoCs. Les résultats des simulations montrent que la personnalisation des NoCs en fonction des ressources disponibles, offre une meilleure performance par rapport aux architectures NoC de base.

L'algorithme de personnalisation de NoC a donné de bonnes performances, et les résultats des simulations montrent que la personnalisation du NoC en fonction des ressources disponibles, offre une meilleure performance par rapport à l'architecture d'interconnexion sur puce de base. Par ailleurs, les résultats de l'évaluation analytique nous ont révélé l'idée de concevoir une nouvelle topologie commune qui pourrait donner un bon compromis de performance de communication. C'est ainsi que nous avons proposé notre nouvelle topologie fractale dénotée FracNoC (k). Cette architecture d'interconnexion a été évaluée par simulation pour différents modèles de trafic de communication sur puce. La comparaison de cette topologie est effectuée avec 2D Mesh, 2D Torus, WK, X-Mesh et Spidergon. Plusieurs modèles de trafic ont été considérés, tel que *Transpose*, *Bit-Reversal*, *Shuffle* et *Uniform*. Les performances de communication analysés sont le débit, la latence, la consommation d'énergie, et de la charge des liens. Les résultats de simulation obtenus montrent que les modèles de trafic et la charge de trafic que les nœuds sources génèrent ont un effet direct sur la performance des NoCs. Ainsi, Ils ont montré que les topologies fractales WK et FracNoC fonctionnent bien dans presque tous les modèles de trafic en raison de leurs propriétés intéressantes, telles que leur degré élevé de régularité, leur symétrie, leur évolutivité et leur facilité d'extension. Une autre propriété intéressante est leur structure fractale, c'est à dire, le réseau peut être construit hiérarchiquement en regroupant les modules de base. En outre, certaines études ont montré que les applications SoC exposent une dépendance à longue portée, c'est à dire, le trafic est auto-similaire ou fractal [15] [273]. Par conséquent, pour des NoCs très grands, les NoCs fractals comme WK et FracNoC pourraient offrir de meilleures performances tout en minimisant la consommation d'énergie.

VII.1.2. Approche de personnalisation de NoC par allocation d'espace tampon

Plusieurs études récentes ont démontré que les NoCs ont des ressources limitées et que les tampons à l'intérieur de commutateurs des NoCs prennent une part importante (voire même dominante) de la surface silicium des systèmes [107] [274] [275] [276] [277], ce qui peut affecter les performances et la consommation d'énergie de ces systèmes. Par conséquent, la taille de ces tampons doit être soigneusement adaptée et personnalisée pour chaque lien/canal d'entrée au commutateur pour mieux correspondre aux modèles de communication d'une application cible spécifique. Les méthodes simples qui allouent uniformément les ressources en mémoire tampon entraînent une utilisation excessive de surface de la puce. Autrement dit, le surdimensionnement des tampons peut affecter la performance, la consommation d'énergie et les besoins en surface silicium du SoC. Il est donc nécessaire d'effectuer une analyse de trafic au moment de la conception pour permettre aux concepteurs de sélectionner les tailles des tampons appropriées pour chaque canal. En outre, la sélection de l'architecture d'interconnexion sur puce, sur la base des modèles de trafic qu'une application spécifique SoC génère, permet aux concepteurs de détecter et de localiser les conflits et les goulots d'étranglement dans le réseau, et donc de le personnaliser pour améliorer sa performance.

Dans la section VII.1.2.1, une approche de modélisation de trafic par réseau à compartiments, inspirée du travail présenté par Guffens et al. [271], est proposée pour l'évaluation analytique des architectures d'interconnexion sur puce. Cette théorie a été utilisée dans différents domaines principalement en biologie et en sciences physiques [278] [279]. Elle peut être utilisée pour modéliser des systèmes qui sont régis par une loi de conservation de la masse et dont les variables d'état sont contraints à rester non-négatives [224] [271]. Cette partie du chapitre se concentre principalement sur l'utilisation pratique de cette approche de modélisation pour la personnalisation de l'espace de mémoire tampon d'une architecture d'interconnexion sur puce donnée et selon une charge de travail d'une application spécifique. En d'autres termes, l'objectif principal de cette approche de modélisation est de permettre aux concepteurs d'adapter les espaces tampons qui conviennent le mieux aux besoins en charge de travail d'une application spécifique. Notre approche de modélisation a été évaluée

analytiquement et par simulation et les résultats préliminaires sont présentés dans la section VII.1.2.3 pour montrer l'efficacité du procédé de modélisation par réseau à compartiments pour une allocation d'espace de mémoire tampon.

Le travail présenté dans cette partie est inspiré de l'approche de modélisation proposée dans [271] [280] [281] pour l'analyse de réseau de macroordinateurs et l'évaluation en prenant en compte également le taux de traitement au niveau de chaque routeur. Ce concept de taux de traitement a également été utilisé dans la méthode du *Network Calculus* dans laquelle le flux maximum de données qui peut être envoyé par chaque commutateur est limité par le flux d'arrivée et de sa vitesse moyenne de traitement. Nous montrons surtout comment l'approche de modélisation par réseau à compartiments peut être utilisée pour l'étude et l'analyse du comportement non-linéaire des applications SoC. En particulier, la modélisation par réseau à compartiments est utilisé comme une méthode d'exploration de l'espace de conception pour la personnalisation ou l'adaptation d'espace de mémoire tampon des architectures d'interconnexion sur puce données selon un modèle de trafic d'une application spécifique.

Notre méthode est comparée à la méthode qui attribue uniformément l'espace de mémoire tampon et les résultats montrent que notre méthode réalise des performances similaires tout en utilisant seulement 70% du budget de ressources.

VII.1.2.1. La modélisation de trafic NoC par réseau à compartiments

Un système NoC peut être considéré comme un système distribué composé de nœuds autonomes qui communiquent en échangeant des messages via un réseau d'interconnexion sur puce. Comme le montre la figure 3.20, il y a trois éléments importants dans le NoC, des cœurs IPs, des routeurs/commutateurs et des liens bidirectionnels. Chaque cœur IP peut être soit une source (expéditeur) soit une destination (récepteur). Chaque commutateur (*switch*) est composé de serveur pour traiter les paquets entrants et les rediriger vers les commutateurs voisins ou vers les cœurs IPs locaux (reliés à lui). En raison de la capacité limitée de traitement du routeur ou du commutateur, les paquets entrants doivent être stockés dans des tampons locaux avant leur retransmission. Chaque port d'entrée au commutateur comporte un tampon de stockage temporaire (cf. le zoom de la figure 3.20). Quand un paquet arrive à un commutateur, il doit être mis dans la mémoire tampon qui correspond à une file d'attente avec un mécanisme de gestion d'ordonnancement en mode PAPS (PAPS: premier arrivé, premier servi). Les tampons sont nécessaires pour absorber les différences de vitesse de traitement des commutateurs et l'injection en rafale du trafic échangé entre les cœurs IPs.

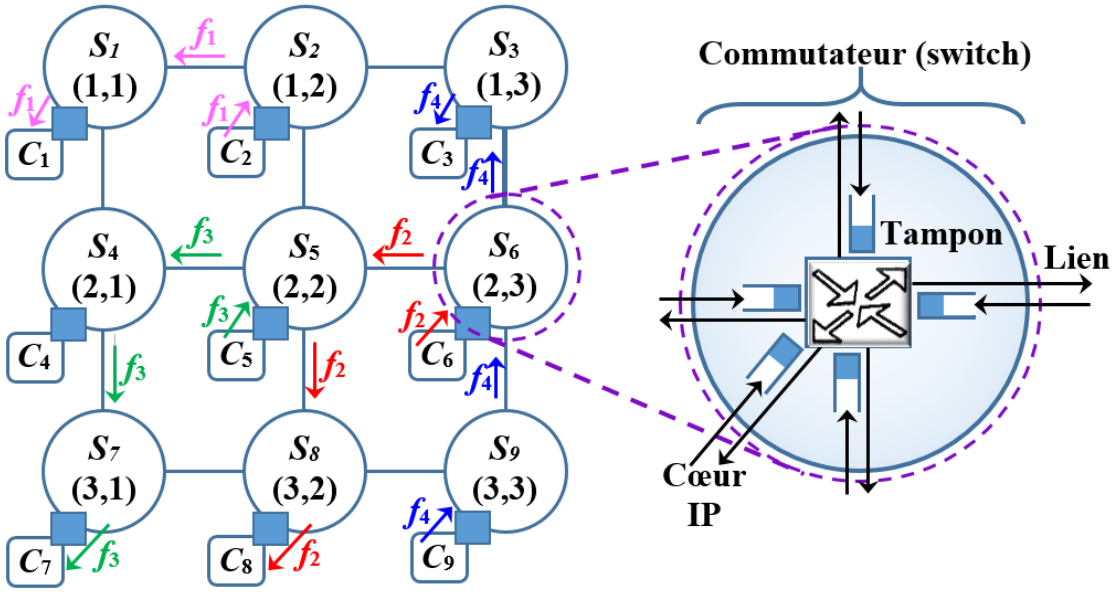


Figure 3.20 : Structure du commutateur et flux de données échangées entre les nœuds sélectionnés

Les communications entre les cœurs IPs sont caractérisées par des flux qui sont représentés par des séquences de sauts. Par exemple, dans la figure 3.20, les flux f_1, f_2, f_3 et f_4 représentent les flux de communication respectivement entre les cœurs IPs de source (C_2, C_6, C_5, C_9) et les cœurs IPs de destination (C_1, C_8, C_7, C_3). Le modèle de trafic du commutateur S_i est représenté sur la figure 3.21.

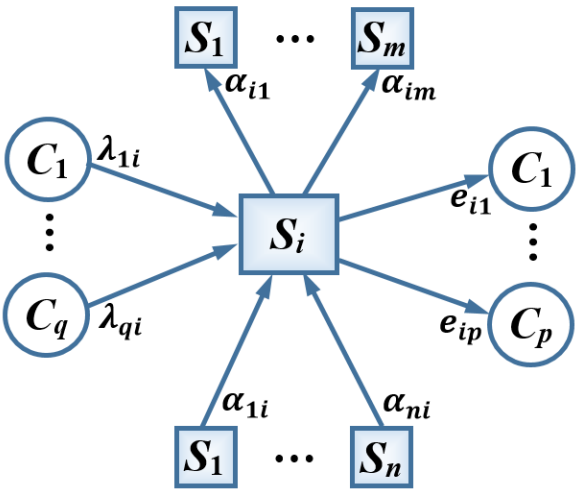


Figure 3.21 : L'entrée/sortie de trafic du nœud Si

Les flux d'entrée d'un commutateur S_i sont : λ_{ki} à partir des q cœurs locaux C_k, $1 \leq k \leq q$ et $k \neq i$; et α_{ji} à partir des commutateurs voisins S_j (c.à.d. la sorties de S_j), $1 \leq j \leq n$, $j \neq i$. Les flux de sortie de S_i sont : e_{ik} vers les p cœurs locaux C_k, $1 \leq k \leq p$ et $k \neq i$; et a_{il} vers les commutateurs voisins S_j, $1 \leq l \leq m$ et $l \neq i$. La modélisation du flux entrant/sortant à un commutateur NoC par réseau à compartiments peut être ensuite exprimée par l'équation différentielle suivante :

$$\dot{x}_i = \left(\sum_{k\neq i}^{q} \lambda_{ki} + \sum_{j\neq i}^{n} \alpha_{ji}(x)\right) - \left(\sum_{k\neq i}^{p} e_{ik}(x) + \sum_{l\neq i}^{m} \alpha_{il}(x)\right) \quad (3.8)$$

Où $x_1(t)$, $x_2(t)$, …, $x_{ns}(t)$ sont appelés des vecteurs d'état du système, ils représentent le nombre total de paquets en attente ou sous traitement dans les commutateurs, de 1 à ns respectivement (ns est le nombre total de commutateurs dans le NoC). Cela signifie que le cumul des paquets dans les tampons d'entrée d'un commutateur ou d'une ressource IPs est la différence entre les flux d'entrées totales (la première parenthèse de l'équation 3.8) et les flux de sortie (la deuxième parenthèse de l'équation 3.8).

Pour exprimer les flux de sortie, le concept de la fonction de taux de traitement, proposé dans [271], peut être utilisé et l'équation 3.8 peut être ensuite réécrite comme suit :

$$\dot{x}_i = \left(\sum_{k\neq i}^{q} \lambda_{ki} + \sum_{j\neq i}^{n} a_{ji} r_j(x)\right) - \left(\sum_{k\neq i}^{p} a_{ik} r_i(x) + \sum_{l\neq i}^{m} a_{il} r_i(x)\right) \quad (3.9)$$

Où les r_i(x) sont des fonctions de taux de traitement, qui peuvent être exprimées comme une factorisation explicite de x_i en :

$$\mu_i = \frac{x_i}{\rho + x_i} \quad (3.10)$$

Avec $\rho > 0$, et μ_i est le taux de service supposé être inférieur à la capacité maximale de transmission des liens de sortie R_i (Bande passante). Ces fonctions devraient être limitées, continues et dérivables, avec $r_i(0) = 0$ et $0 \leq r_i(x) \leq \mu_i, \forall x_i > 0$. Le paramètre μ_i est la vitesse de service du commutateur i. Le paramètres a_{ik}, a_{il} et a_{ji} sont des valeurs positives qui représentent la fraction de paquets qui sont envoyés sur le lien ($i \rightarrow k$), ($i \rightarrow l$), et ($j \rightarrow i$) respectivement, où :

$$\sum_{k\neq i}^{p} a_{ik} + \sum_{l\neq i}^{m} a_{il} = 1 \quad (3.11)$$

En utilisant l'expression matricielle, l'équation 3.9 peut être réécrite sous un format compact formé comme suit :

$$\dot{x} = G(x)\, x \; + \; \theta \tag{3.12}$$

Où *x* est le vecteur d'état avec des éléments x_i tel que $1 \leq i \leq ns$, et θ est le vecteur d'entrée avec les éléments λ_{ki} tel que $1 \leq k \leq q$.

$G(x)$ est une matrice avec les propriétés suivantes [224] :

1. $G(x)$ est une matrice *Metzler* avec des entrées non-diagonales non-négatives, qui sont soit 0, soit :

$$g_{ij}(x) = \frac{a_{ij}\,\mu_i}{\rho + x_i}, \quad \text{(avec i} \neq \text{j)} \tag{3.13}$$

2. les éléments diagonaux sont non-positifs où :

$$g_{ii}(x) = -\left(\sum_{k\neq i}^{p} \frac{a_{ik}\,\mu_i}{\rho + x_i} + \sum_{l\neq i}^{m} \frac{a_{il}\,\mu_i}{\rho + x_i}\right) \tag{3.14}$$

3. $G(x)$ est une matrice à diagonale dominante, c'est-à-dire que :

$$|\, g_{ii}(x)| \geq \sum_{j\neq i}^{m} g_{ji}(x) \tag{3.15}$$

4. la matrice $G(x)$ ne devrait pas être singulière et stable. Cette propriété peut être prouvée en vérifiant si le réseau est entièrement sortie-connectée (*outflow connected*), c'est-à-dire que pour chaque nœud i, il existe un chemin de i à un autre nœud k à partir de laquelle il est une sortie. Par exemple, le réseau illustré à la figure 3.20 est complètement sortie-connectée parce que de toute nœud de source il y a un chemin vers un autre nœud avec une sortie (destination).

VII.1.2.2. Étude d'évaluation

Dans cette section, nous montrons l'utilisation pratique de l'approche de modélisation par réseau à compartiments pour allouer l'espace de mémoire tampon de chaque canal d'entrée. Les résultats sont calculés à la fois par évaluation analytique et par simulation. Nous analysons en particulier la taille maximale du tampon nécessaire pour stocker les paquets en attente ou sous traitement. Pour cette évaluation, nous considérons un NoC de type 2D 4x4 Mesh pour notre cas d'étude. La figure 3.22 illustre les flux de données transmis entre les cœurs participants à la communication.

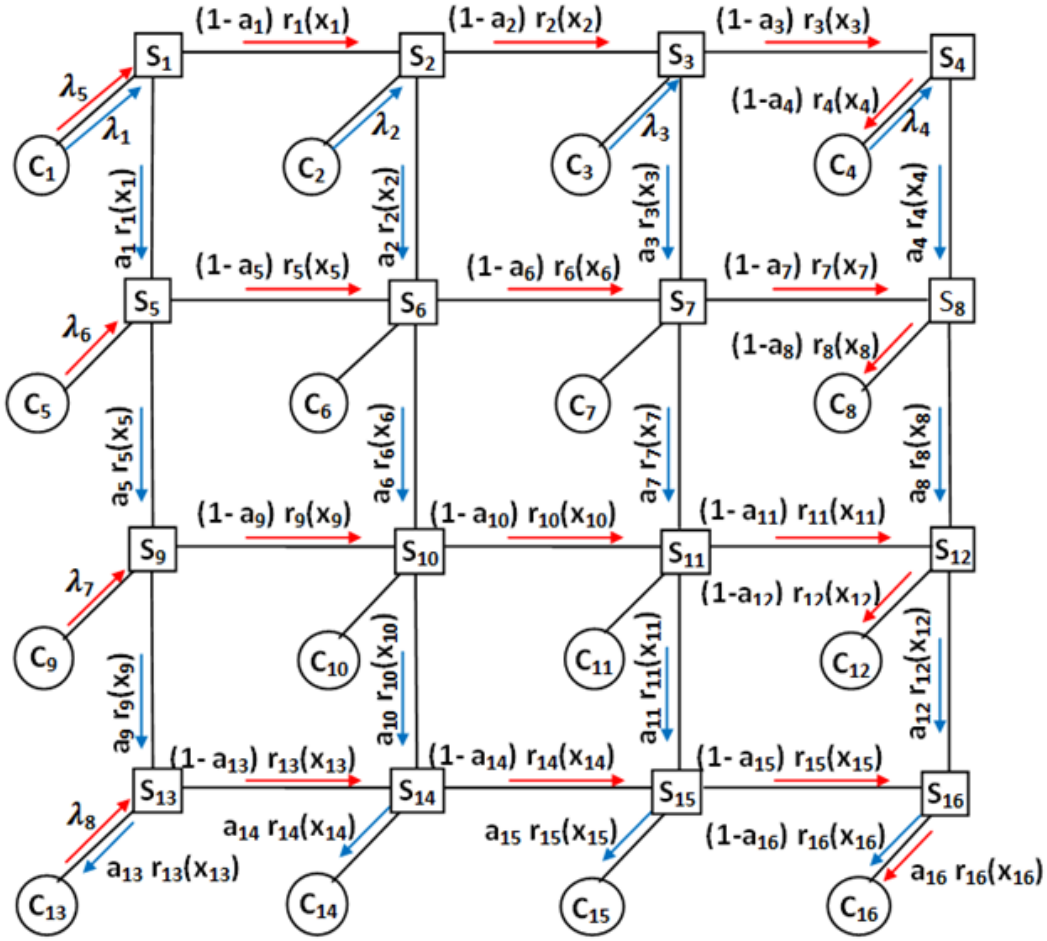


Figure 3.22 : Les flux de données utilisées pour la modélisation par réseau à compartiments, dans le NoC de type 2D 4x4 Mesh

L'application est représentée comme un processus parallèle de communication déjà mis en place entre les cœurs IPs. Les cœurs IPs sélectionnés pour être des sources de trafic sont C_1, C_2, C_3, C_4, C_5, C_9 et C_{13} et les cœurs IPs sélectionnés pour être des destinations sont C_4, C_8, C_{12}, C_{13}, C_{14}, C_{15} et C_{16}. Les flux de données sont calculés en utilisant un protocole de routage déterministe de flits directs entre les sources (*sources*) et les destinations (*sinks*) sélectionnées comme suit, $(C_1 \rightarrow C_4)$, $(C_1 \rightarrow C_{13})$, $(C_2 \rightarrow C_{14})$, $(C_3 \rightarrow C_{15})$, $(C_4 \rightarrow C_{16})$, $(C_5 \rightarrow C_8)$, $(C_9 \rightarrow C_{12})$, $(C_{13} \rightarrow C_{16})$. Comme le montre la figure 3.21, huit flux de données sont calculés sous forme d'une séquence de sauts comme suite :

- $f_1 = (C_1 \rightarrow S_1 \rightarrow S_5 \rightarrow S_9 \rightarrow S_{13} \rightarrow C_{13})$
- $f_2 = (C_2 \rightarrow S_2 \rightarrow S_6 \rightarrow S_{10} \rightarrow S_{14} \rightarrow C_{14})$
- $f_3 = (C_3 \rightarrow S_3 \rightarrow S_7 \rightarrow S_{11} \rightarrow S_{15} \rightarrow C_{15})$
- $f_4 = (C_4 \rightarrow S_4 \rightarrow S_8 \rightarrow S_{12} \rightarrow S_{16} \rightarrow C_{16})$
- $f_5 = (C_1 \rightarrow S_1 \rightarrow S_2 \rightarrow S_3 \rightarrow S_4 \rightarrow C_4)$
- $f_6 = (C_5 \rightarrow S_5 \rightarrow S_6 \rightarrow S_7 \rightarrow S_8 \rightarrow C_8)$
- $f_7 = (C_9 \rightarrow S_9 \rightarrow S_{10} \rightarrow S_{11} \rightarrow S_{12} \rightarrow C_{12})$
- $f_8 = (C_{13} \rightarrow S_{13} \rightarrow S_{14} \rightarrow S_{15} \rightarrow S_{16} \rightarrow C_{16})$

Après avoir défini les flux de données entre les cœurs IPs participants à l'émission/réception de données (cf. figure 3.22), les équations des variables d'état, de chaque commutateurs, issues

de la modélisation par réseau à compartiments peuvent être écrites, après l'interconnexion de tous les flux d'entrée et de sortie, comme suite :

- $\dot{x}_1 = (\lambda_1 + \lambda_5) - \big((1 - a_1)r_1(x_1) + a_1 r_1(x_1)\big)$
- $\dot{x}_2 = \big(\lambda_2 + (1 - a_1)r_1(x_1)\big) - \big((1 - a_2)r_2(x_2) + a_2 r_2(x_2)\big)$
- $\dot{x}_3 = \big(\lambda_3 + (1 - a_2)r_2(x_2)\big) - \big((1 - a_3)r_3(x_3) + a_3 r_3(x_3)\big)$
- $\dot{x}_4 = \big(\lambda_4 + (1 - a_3)r_3(x_3)\big) - \big((1 - a_4)r_4(x_4) + a_4 r_4(x_4)\big)$
- $\dot{x}_5 = \big(\lambda_6 + a_1 r_1(x_1)\big) - \big((1 - a_5)r_5(x_5) + a_5 r_5(x_5)\big)$
- $\dot{x}_6 = \big((1 - a_5)r_5(x_5) + a_2 r_2(x_2)\big) - \big((1 - a_6)r_6(x_6) + a_6 r_6(x_6)\big)$
- $\dot{x}_7 = \big((1 - a_6)r_6(x_6) + a_3 r_3(x_3)\big) - \big((1 - a_7)r_7(x_7) + a_7 r_7(x_7)\big)$
- $\dot{x}_8 = \big((1 - a_7)r_7(x_7) + a_4 r_4(x_4)\big) - \big((1 - a_8)r_8(x_8) + a_8 r_8(x_8)\big)$
- $\dot{x}_9 = \big(\lambda_7 + a_5 r_5(x_5)\big) - \big((1 - a_9)r_9(x_9) + a_9 r_9(x_9)\big)$
- $\dot{x}_{10} = \big((1 - a_9)r_9(x_9) + a_6 r_6(x_6)\big) - \big((1 - a_{10})r_{10}(x_{10}) + a_{10} r_{10}(x_{10})\big)$
- $\dot{x}_{11} = \big((1 - a_{10})r_{10}(x_{10}) + a_7 r_7(x_7)\big) - \big((1 - a_{11})r_{11}(x_{11}) + a_{11} r_{11}(x_{11})\big)$
- $\dot{x}_{12} = \big((1 - a_{11})r_{11}(x_{11}) + a_8 r_8(x_8)\big) - \big((1 - a_{12})r_{12}(x_{12}) + a_{12} r_{12}(x_{12})\big)$
- $\dot{x}_{13} = \big(\lambda_8 + a_9 r_9(x_9)\big) - \big((1 - a_{13})r_{13}(x_{13}) + a_{13} r_{13}(x_{13})\big)$
- $\dot{x}_{14} = \big((1 - a_{13})r_{13}(x_{13}) + a_{10} r_{10}(x_{10})\big) - \big((1 - a_{14})r_{14}(x_{14}) + a_{14} r_{14}(x_{14})\big)$
- $\dot{x}_{15} = \big((1 - a_{14})r_{14}(x_{14}) + a_{11} r_{11}(x_{11})\big) - \big((1 - a_{15})r_{15}(x_{15}) + a_{15} r_{15}(x_{15})\big)$
- $\dot{x}_{16} = \big((1 - a_{15})r_{15}(x_{15}) + a_{12} r_{12}(x_{12})\big) - \big((1 - a_{16})r_{16}(x_{16}) + a_{16} r_{16}(x_{16})\big)$

Où λ_i est le débit d'injection du cœur IP source C_i, avec a_i est la pondération sur le flux sortant d'un commutateur dont leur somme est égal à 1 ($a_i + (1 - a_i) = 1$), dans notre cas $a_i = 0.5$, $\forall\, i$. Il est intéressant de noter que, en NoC de type 2D Mesh de notre cas d'étude, il y a un seule cœur IP lié à chaque commutateur. Dans le cas général, le NoC peut avoir plusieurs cœurs IPs connectés à chaque commutateur donné, comme le cas du NoC de type *Fat-Tree* ou *Butterfly-Fat-Tree* où les commutateurs de la feuille peuvent être connectés à plusieurs cœurs IPs.

VII.1.2.3. Résultats de simulation

Pour montrer l'efficacité de cette approche, nous avons comparé les résultats d'évaluation analytique avec ceux obtenus par simulation du système en utilisant le même modèle de trafic [264] [272]. La bande passante des liens *R* est configuré pour être équivalente à 200 flits/s en raison de la quantité de ressources requise pour chaque instance de simulation. Chaque processus est relié à un générateur de trafic qui injecte les flits en fonction de la distribution sans mémoire (sources de Poisson) avec une moyenne de 12,5 ms, 16,6 ms, 25ms, 50ms, ce qui est équivalent aux respectivement. Les tailles des tampons à l'intérieur des cœurs IPs ne sont pas calculées dans cette évaluation, nous supposons que les cœurs IPs ont assez d'espace et de vitesse à manipuler et à traiter les données d'entrées. Dans les commutateurs, chaque port a une mémoire tampon pour le stockage temporaire des paquets reçus. Le temps de simulation est fixé à 10 secondes.

Les figures 3.23, 3.24, 3.25 et 3.26 montrent la variation de la taille des tampons lorsque les taux d'injection sont fixés à 20, 40, 60 et 80 flits/s respectivement. Les résultats illustrés dans ces figures montrent que lorsque le débit d'injection augmente, plus d'espace de mémoires

tampons est nécessaire pour éviter la perte des flits arrivants à ce commutateur. Comme la vitesse d'injection augmente, le réseau devient de plus en plus encombré avec un trafic dense et donc plus d'espace de mémoire tampons est nécessaire pour absorber les différences de taux de traitement des commutateurs et d'arrivé des rafales de trafic. Nous pouvons également voir que les résultats analytiques sont dans le même ordre de grandeur que ceux obtenus par des simulations, c'est à dire, les résultats de simulation sont conformes au modèle analytique. Ils présentent un écart de moins de 10% en moyenne.

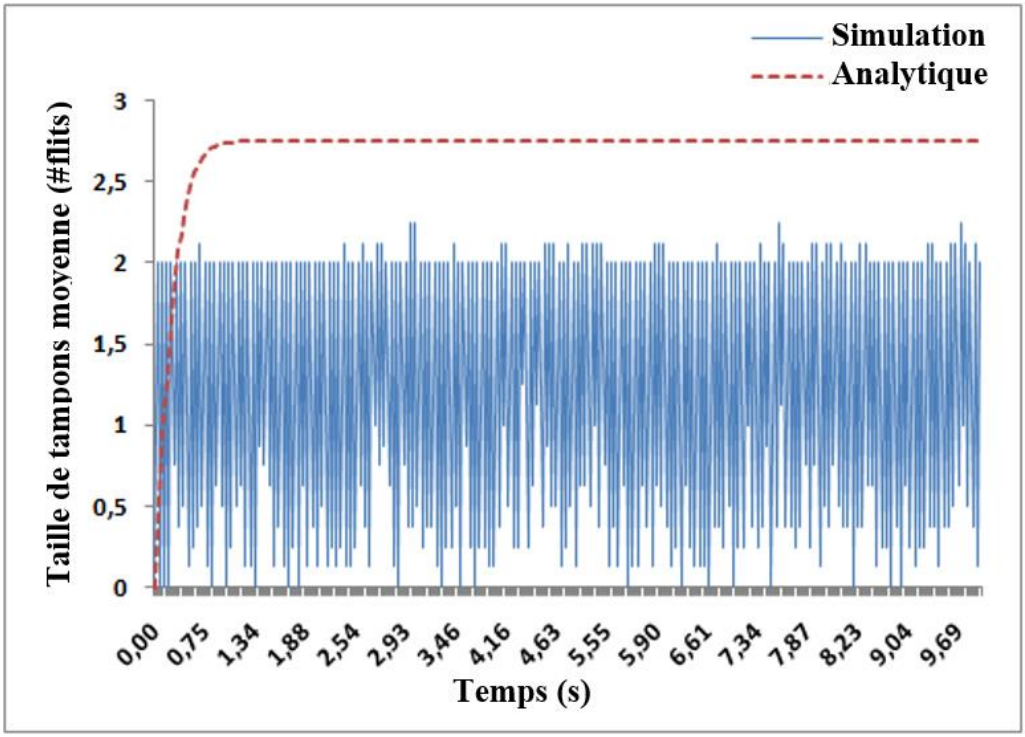


Figure 3.23 : La variation de taille des tampons dans le temps calculée par simulation et analyse lorsque le débit d'injection est de 20 flits/s

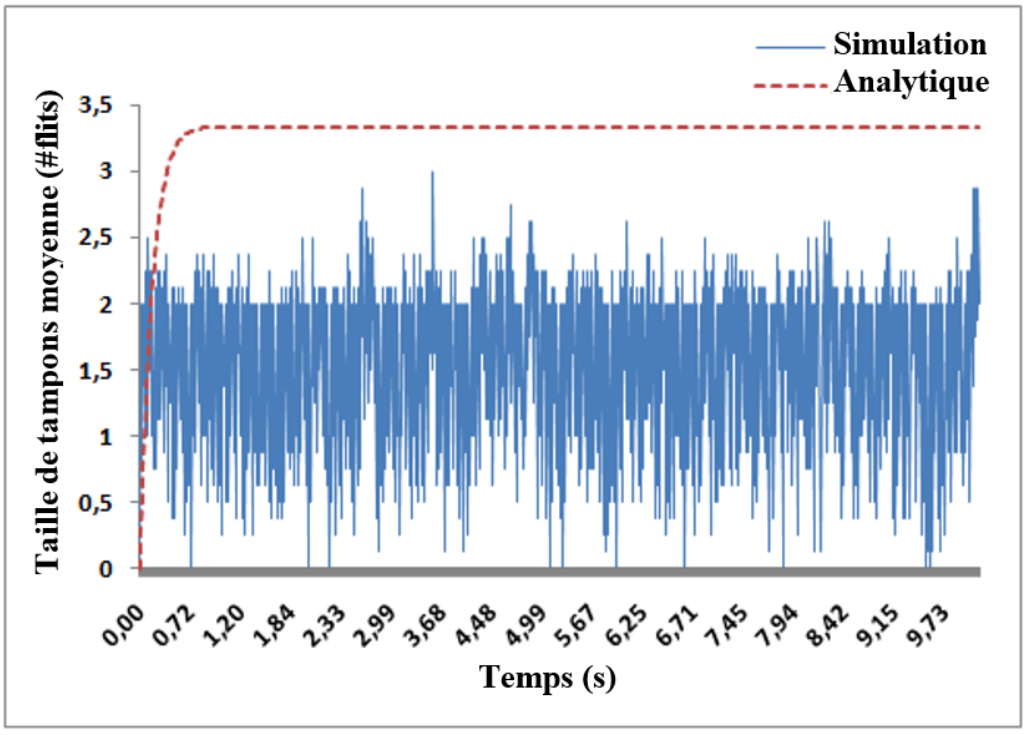


Figure 3.24 : La variation de taille des tampons dans le temps calculée par simulation et analyse lorsque le débit d'injection est de 40 flits/s

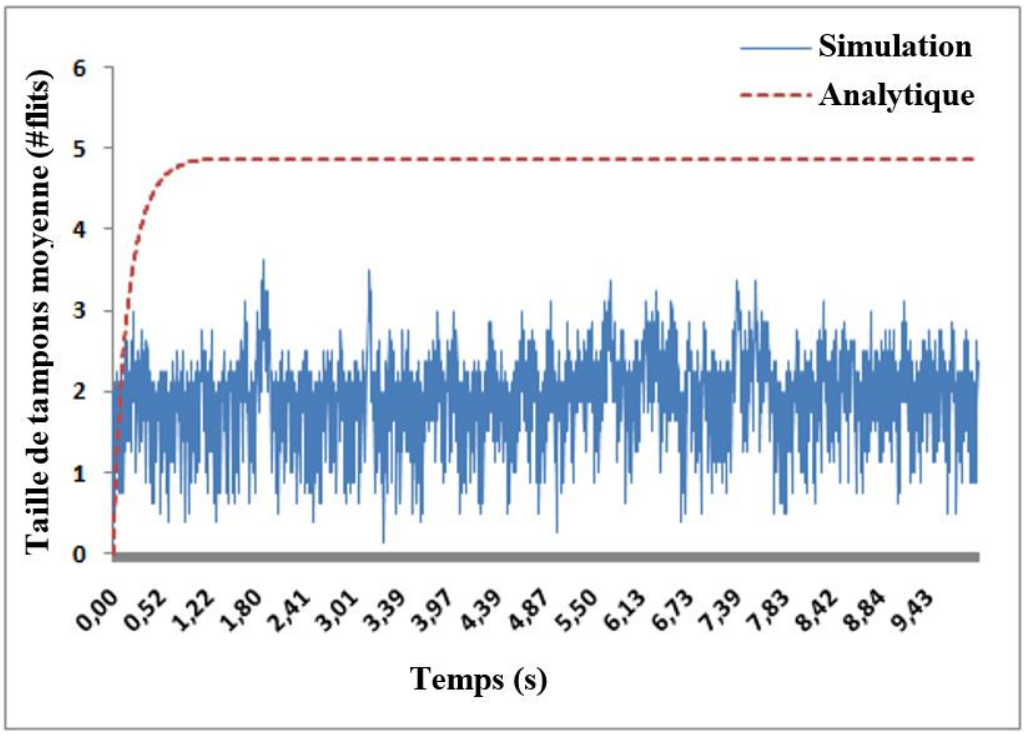


Figure 3.25 : La variation de taille des tampons dans le temps calculée par simulation et analyse lorsque le débit d'injection est de 60 flits/s

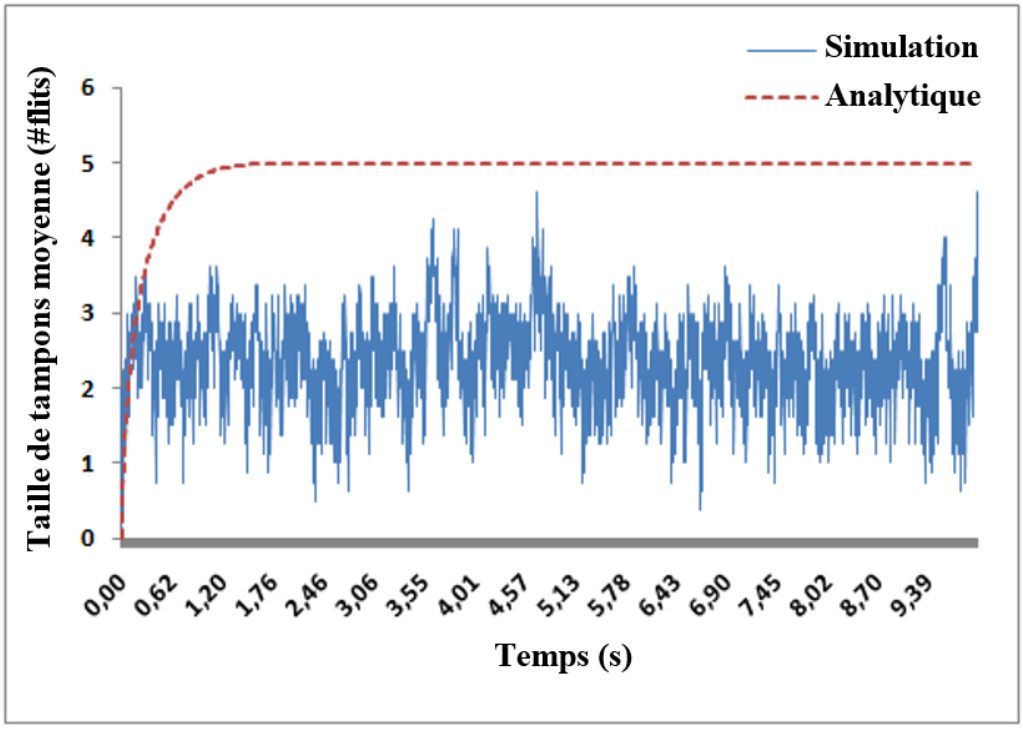


Figure 3.26 : La variation de taille des tampons dans le temps calculée par simulation et analyse lorsque le débit d'injection est de 80 flits/s

Nous avons également comparé le modèle de *Network Calculus* avec notre modèle de réseau à compartiments ainsi que les résultats de simulations sous différent taux d'injection (20, 40, 60, et 80 Mbps). Après avoir défini les flux de données et des cœurs IPs participant à l'émission/réception des données, l'ensemble du réseau est décrit pour obtenir le modèle basé sur le *Network Calculus* par émergement de tous les flux de sortie et d'arrivée. Les résultats présentés à la figure 3.27 montrent que notre modèle de réseau à compartiments fournit une estimation plus précise de la taille des tampons par rapport au *Network Calculus*, qui fournit une analyse du pire-cas (borne supérieure).

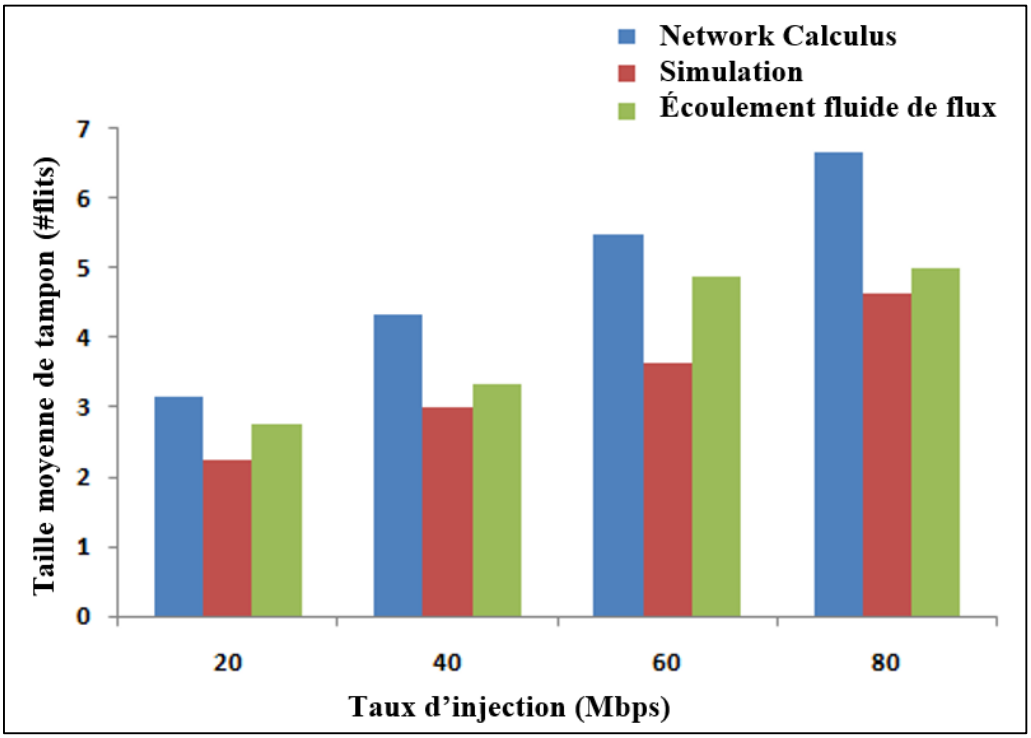


Figure 3.27 : La taille maximale des tampons évaluée en utilisant la modélisation par réseau à compartiments, Network Calculus et la simulation

VII.2. Synthèse

Dans ce chapitre, deux approches de personnalisation de la topologie de NoC sont proposées. La première approche est basée sur l'insertion de liens pour la personnalisation de NoC sans tenir compte du trafic. En outre, une nouvelle topologie fractale et récursive dénotée FracNoC (*k*) a été proposée. Des simulations sont conduites et les résultats montrent l'efficacité de l'approche d'insertion de liens qui permet d'améliorer progressivement les performances tout en ajoutant plus de liens. Les résultats montrent aussi que la topologie FracNoC est bien placée parmi les autres.

La deuxième approche est une approche d'exploration de l'espace de conception basée sur la modélisation du trafic NoC par réseau à compartiments. Le but est de permettre aux concepteurs, à un stade très tôt du processus de conception, d'analyser le trafic rapidement et d'allouer dynamiquement l'espace de mémoire tampon, requis pour chaque canal d'entrée/sortie. Les résultats de simulation montrent l'efficacité de cette approche.

V. Chapitre 4 : Une approche dynamique de contrôle de flux dans les NoCs

Dans ce chapitre, nous présentons une approche utilisant la théorie des systèmes dynamiques pour permettre aux éléments du NoC d'ajuster dynamiquement leur flux d'entrée en utilisant un mécanisme de contrôle de flux basé sur la rétroaction. Il s'agit d'un mécanisme de contrôle de flux basé sur la modélisation par réseau à compartiments présenté par Guffens et al. [271] [272]. Cette théorie a été utilisée dans différents domaines, notamment en biologie et en sciences physiques, principalement pour modéliser des systèmes qui sont régis par une loi de conservation de masse et dont les variables d'état sont contraintes de rester non-négatives. En d'autres termes, cette approche permet aux commutateurs de gérer et de coordonner le trafic à l'aide d'un mécanisme de contrôle de flux par rétroaction, et par conséquent les cœurs IPs sources de trafic vont diminuer leurs taux de transmission.

VIII.1. Approche de systèmes dynamiques

VIII.1.1. La modélisation de la théorie des systèmes dynamiques

La théorie des systèmes dynamiques est une branche des mathématiques qui étudie les propriétés d'un système dynamique. Elle peut être décrite par l'ensemble des méthodes et des outils de modélisations rigoureuses pour comprendre la structure et le comportement des systèmes complexes. Elle permet aux concepteurs de créer un modèle d'un système et ensuite de le simuler pendant un certain temps, afin de comprendre son comportement, et/ou d'évaluer diverses stratégies pour ses opérations [282] [283] [284].

Cette théorie a été initialement créée en 1957 par Jay W. Forrester comme une méthodologie pour construire des modèles de simulation informatique. La modélisation de la théorie des systèmes dynamiques contient cinq types d'équations [212] :

- Le niveau ou l'accumulation,
- Le taux ou la politique de variables,
- Les auxiliaires,
- Le constant,
- Les conditions de la valeur initiale, où

Le niveau ou l'accumulation : est le taux actuel de la variable, résultant de la différence entre les entrées et les sorties sur une certaine période de temps (calculé sur une période distincte). Un exemple de ceci serait le solde d'un compte, le solde de la production végétale, le nombre de personnel.

Le taux ou la politique de variables : il est le débit instantané qui augmente ou diminue les taux des variables (c'est à dire les niveaux). Les taux montrent le mouvement des flux, tandis que les niveaux montrent le résultat comme l'état du système, qui change en raison de ce mouvement. Dans les systèmes naturels, les taux suivent les règles de la nature. Dans les autres systèmes, les taux reflètent les orientations stratégiques qui influencent les choix personnels.

Les auxiliaires : ce sont les paramètres auxiliaires pour le calcul des taux. Les taux et les auxiliaires sont fondés sur certaines constantes, inchangeables dans le temps où les systèmes dynamiques sont étudiés. Dans [285], l'auteur fournit la dynamique du système avec un paramètre supplémentaire, le "retard".

En ce qui concerne le modèle de décision des systèmes dynamiques, les auteurs de [286] et [287] proposent les étapes suivantes :

1. Définition des problèmes à résoudre et des objectifs à atteindre,
2. Description du système avec des diagrammes de réalimentation (boucle causale / diagramme d'influence),
3. Développement des équations,
4. Collection des conditions de la valeur initiale, soit à partir des sources historiques, ou en interrogeant des experts qui sont familiers avec le système qui est en cours d'évaluation,
5. Ratification de modèle pour l'élaboration de sa crédibilité,
6. Simulation du modèle afin de contrôler la politique et l'action qui mènera à la réalisation des objectifs définis.

Les simulations de la théorie des systèmes dynamiques sont basées sur le principe d'un stock (un récipient de données), de flux de données et de rétroaction de flux (transmission en aval), comme illustré dans la figure 4.1. Un stock est un réservoir ou un magasin (réel ou virtuel) de ressources de quelque chose (matérielle ou immatérielle), par exemple l'eau dans un réservoir, le nombre d'arbres dans une forêt, le montant d'argent à la banque, nombre de flits dans une mémoire tampon, etc. Ces réservoirs, dénommés "compartiments", sont interconnectés par des mécanismes de transfert (flux de matière, d'énergie ou d'information). Les entrées du système (injections) sont des énergies, des ressources ou des flits qui coulent dans le stock, alors que les

sorties du système (éjections) sont des informations, des énergies, des ressources ou des flits qui coulent du stock. La rétroaction de flux survient lorsque des changements dans le stock affectent le flux entrant/sortant du même stock, par exemple, mettre de l'argent dans un compte d'épargne susciteront des intérêts, ce qui augmente ensuite le montant d'argent dans le compte.

La dynamique générale des réseaux dynamiques est décrite par des systèmes de "lois de conservation" qui s'écrivent sous la forme d'équations différentielles ordinaires (modèles en dimension finie) ou d'équations aux dérivées partielles hyperboliques (modèles en dimension infinie). Mathématiquement, les équations différentielles de flux entrant/sortant d'un système peuvent être écrites comme suite :

$$\dot{Stock} = injections\ (t)\ -\ ejections\ (t) \qquad (4.1)$$

La résolution de ce modèle dynamique permet de déterminer la quantité de matériel ou d'information accumulée dans chaque stock du système à tout instant *t*.

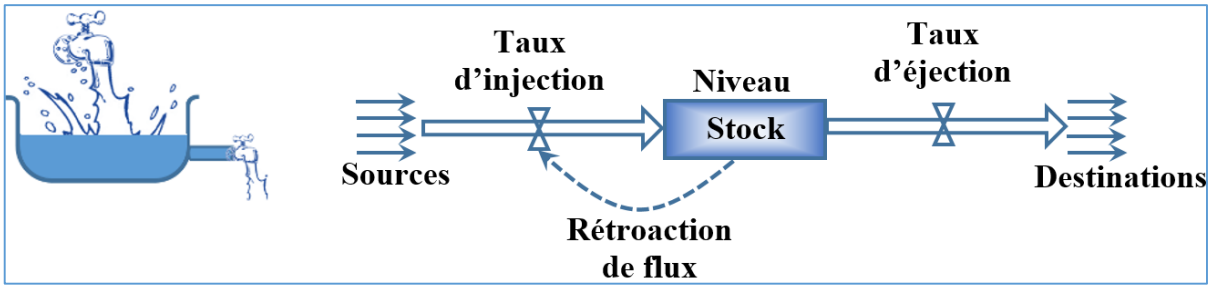


Figure 4.1 : Exemple d'un système et le modèle Stock/flux correspondant

Un système sur puce SoC est un système dynamique, il est basé sur une infrastructure d'interconnexion sur puce (NoC). Un NoC est une combinaison de différents éléments (ex., commutateurs, liens, etc.) et de protocoles (ex., routage, commutation, etc.) qui déterminent l'architecture et les modes de communication. Rappelons qu'il y a trois éléments importants dans un NoC : des cœurs IPs, des routeurs (ou des commutateurs) et des liens bidirectionnels comme illustré dans la figure 4.2. Le commutateur est composé d'une unité de contrôle pour traiter les flits entrants et les diriger aux commutateurs voisins ou bien aux cœurs IPs locaux.

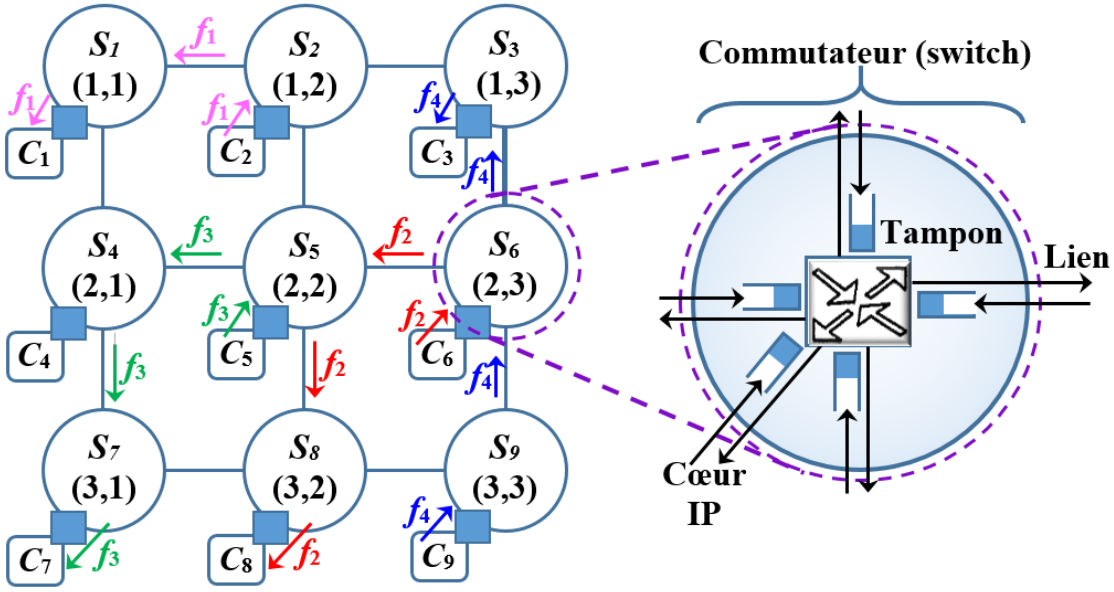


Figure 4.2 : Un NoC de type 2D 3x3 Mesh, avec 9 cœurs IPs de traitement

Dans ces SoCs dynamiques, les communications entre les cœurs sources et les cœurs destinataires se font par des échanges de paquets (flits) de données. Les chemins que les paquets de données peuvent éventuellement prendre dans le réseau dépendent principalement de la faculté d'adaptation permise par l'algorithme de routage (algorithme de routage partiellement

ou totalement adaptative). En outre, l'acheminement efficace de paquets de données au sein d'un réseau dynamique est essentiel afin d'obtenir de bonnes performances des applications en cours d'exécution. En effet, au moment de l'exécution, certaines zones du NoC peuvent devenir indisponibles (commutateurs ou liens congestionnés) et les paquets de données doivent contourner ces zones. Il existe aussi d'autres techniques pour y remédier au problème de congestion dans les réseaux NoC, tels que l'allocation dynamique de la bande passante des liens ou bien l'allocation dynamique de la taille des tampons, selon le trafic de l'application en cours d'exécution. Mais tout d'abord, il va falloir prévoir et détecter dynamiquement les zones du NoC congestionnées. Ces zones indisponibles doivent être détectées en permanence par l'utilisation d'un mécanisme de contrôle de flux.

Les réseaux sur puce sont donc considérés comme des réseaux dynamiques conservatifs. Leur structure est représentée par une topologie quelconque (ex, 2D Mesh), comme cela est illustré dans la figure 4.2. Chaque commutateur du NoC dispose d'un réservoir de stockage (appelé "compartiment") qui contient une quantité variable de jetons.

VIII.1.2. Le modèle de contrôle de flux par rétroaction

Considérons qu'un NoC est composé de trois ensembles : les commutateurs *S*, les cœurs *C*, et les tampons *B*. Chaque cœur peut être une source ou une destination. La communication entre deux cœurs est caractérisée par des flux qui sont représentés par des séquences de sauts. Vu que la capacité de traitement des commutateurs et la bande passante des liens sont limitées ; les flits entrants doivent être stockés dans des tampons locaux des commutateurs avant leur retransmission. Ces tampons sont nécessaires pour absorber les différences de vitesse des commutateurs et les rafales de trafic échangées entre les cœurs. Le comportement des flux de données peut être modélisé en utilisant un mécanisme de système dynamique qui peut être considéré comme le modèle de système dynamique suivant :

$$\dot{x}_i = \left(\lambda_{ji} + \sum_{k\in U_i} \alpha_{ki}\right) - \left(e_{ij} + \sum_{k\in D_i} \alpha_{ik}\right) \quad (4.2)$$

Où $\dot{x}_i$ est l'occupation de la mémoire tampon b_i, qui a les flux d'entrée λ_{ji} à partir d'un cœur source local $j \in C$, et les flux d'entrée α_{ki} à partir des tampons en amonts (*upstream*) $U_i \subset B$. Il a aussi les flux de sortie α_{ik} vers des tampons en avals (*downstream*) $D_i \subset B$ et un flux de sortie e_{ij} vers un cœur récepteur local $j \in C$. Les variables $x_1(t)$, $x_2(t)$, ..., $x_n(t)$, sont appelées des vecteurs d'état du système, elles représentent le nombre total de flits en attente ou en cours de traitement dans les tampons b_1, b_2, ..., b_n (où n est le nombre total de tampons). Cela signifie simplement que les flits accumulés dans le tampon d'entrée b_i est la différence entre le flux total d'entrée (le premier et second terme de l'équation 4.2) et le flux total de sortie (le troisième et quatrième terme de l'équation 4.2). Pour exprimer les flux de sortie, le concept de la fonction du taux de traitement avec un mécanisme de contrôle de flux, présenté en [271], est utilisé et l'équation 4.2 peut être réécrite comme suite :

$$\dot{x}_i = \left(\mu_{ji} r_j(x) + \sum_{k\in U_i} \alpha_{ki}\, \mu_{ki} r_k(x)\right) - \left(\alpha_{ij} r_i(x) + \sum_{k\in D_i} \alpha_{ik} \mu_{ik} r_i(x)\right) \quad (4.3)$$

Où les $r_i(x)$ sont des fonctions du taux de traitement, qui peuvent être exprimées comme une factorisation explicite de x_i par :

$$r_i(x) = \frac{\mu_i x_i}{a + x_i} \quad (4.4)$$

Ces fonctions devraient être limitées, continues et dérivables, avec $r_i(0) = 0$ et $0 \le r_i(x) \le \mu_i, \forall\, x_i > 0$ où le paramètre μ_i est la vitesse de service du tampon b_i, et a est un nombre entier positif. Les paramètres α_{ki} et α_{ik} sont des valeurs positives qui représentent la fraction de flits présentée par le tampon en amont b_k vers b_i et de b_i vers le tampon en aval b_k respectivement. Le paramètre α_{ij} est la fraction de flits soumis au cœur local $j \in C$. Le paramètre μ_{ij} est un paramètre de contrôle de congestion utilisé pour ralentir la transmission du trafic avec l'objectif principal d'empêcher le tampon de déborder en le maintenant à une

occupation inférieure à un seuil donné (ex., sa taille réelle). Le mécanisme de contrôle par rétroaction, qui est utilisé en [201], est utilisé ici pour contrôler la charge x_i comme suite :

$$\mu_{ij} = \frac{\sigma_j - x_i}{\epsilon_j + \sigma_i - x_i} \tag{4.5}$$

Cela signifie que l'occupation du tampon reste bornée par σ_j et puis aucun flit ne seras perdu. La loi de contrôle peut être mise en œuvre par des réservoirs (récipients) de jetons. Un réservoir à jetons permet une régulation de l'enveloppe du trafic à l'aide de deux paramètres, la vitesse de remplissage des jetons, qui dicte la vitesse moyenne du trafic, et la taille du réservoir, ce qui détermine l'état de rafale autorisée [288]. Par conséquent, μ_{ij} de l'équation 4.5 peut-être réécrit en ajoutant une autre variable d'état comme suite :

$$\mu_{ij} = \frac{y_i}{\epsilon_j + y_i} \tag{4.6}$$

Où $y_i = \sigma_i - x_i$, y_i est initialisée par σ_i et représente le nombre de jetons dans chaque réservoir. Dans ce cas-là, un flit sera transféré à partir d'un tampon b_i vers un autre tampon b_j, si au moins un jeton est disponible dans le réservoir de b_i représenté par y_i.

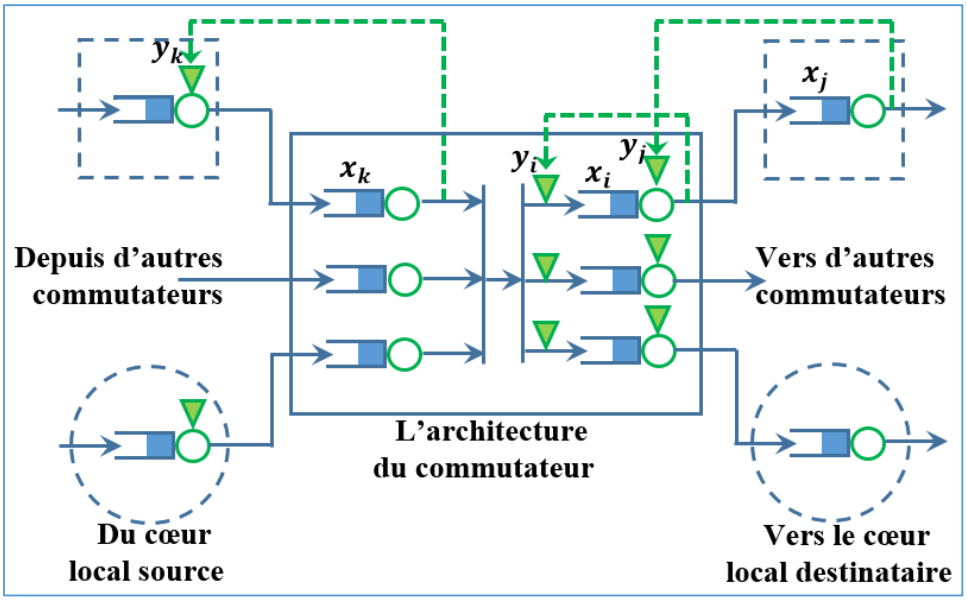


Figure 4.3 : L'architecture des commutateurs

Cela signifie que μ_{ij} permet la modulation de la vitesse de sortie de b_i en fonction de l'état de b_j. Nous avons utilisé la même architecture de commutation considérée dans le simulateur développé, elle est semblable à une architecture de commutateur de paquets traditionnel décrite dans [272]. Grâce à cette architecture (cf. figure 4.3), les flits sont stockés dans le tampon jusqu'à ce qu'ils soient acceptés par la toile de commutation (réseau NoC), et commutés par la suite vers le tampon de sortie approprié avant d'être transmet au commutateur suivant ou bien au cœur IP local. Le mécanisme de contrôle de flux garanti qu'aucun débordement de tampon ne se produit et donc qu'aucun paquet ne soit perdu. Grâce à cette architecture, les commutateurs ont des variables communes partagées et ils ne doivent accéder qu'aux variables des commutateurs d'injection/éjection de flux. Par exemple, le routeur d'injection de flux ne peut pas connaître instantanément la valeur du contenu du tampon x_i.

Pour illustrer l'algorithme de contrôle, prenons comme exemple l'architecture représentée sur la figure 4.4. Le flux de données transmis entre les deux cœurs C_4 et C_6 est indiqué par des flèches rouges. Chaque tampon d'injection alimente un tampon d'entrée à l'intérieur du commutateur. Chaque interface de sortie est également équipée d'un tampon séparé. Chaque tampon b_i a deux variables x_i et y_i qui représentent l'occupation du tampon et le niveau du réservoir (nombre de jetons disponibles) respectivement. Comme l'illustre la figure 4.4, le cœur

injecte des données à un certain débit vers sa mémoire tampon b_n et la variable x_n est incrémentée avec la quantité de flits injectés (dans ce cas d'étude, un seul cœur est relié à chaque commutateur). Le tampon b_n à son tour, alimente le tampon b_m du commutateur, si au moins un jeton est disponible dans son réservoir (représenté par la variable y_m). Par exemple, si T jetons sont disponibles ($y_m = T$), le tampon b_n pourrait envoyer jusqu'à T flits au tampon b_m (x_m est incrémentée et y_m est décrémentée respectivement). Le tampon b_m à son tour, envoie un flit à b_l et une variable temporaire est incrémentée. Lorsque l'intervalle de temps Δt expire, cette variable temporaire est envoyée à b_n. Lors de la réception de cette variable temporaire par b_n, elle est accumulée à y_m, qui représente le nombre de jetons disponibles. Le même processus est répété avec des tampons b_k, b_i, b_j, b_p, et b_q jusqu'à que les flits arrivent à leur destinations.

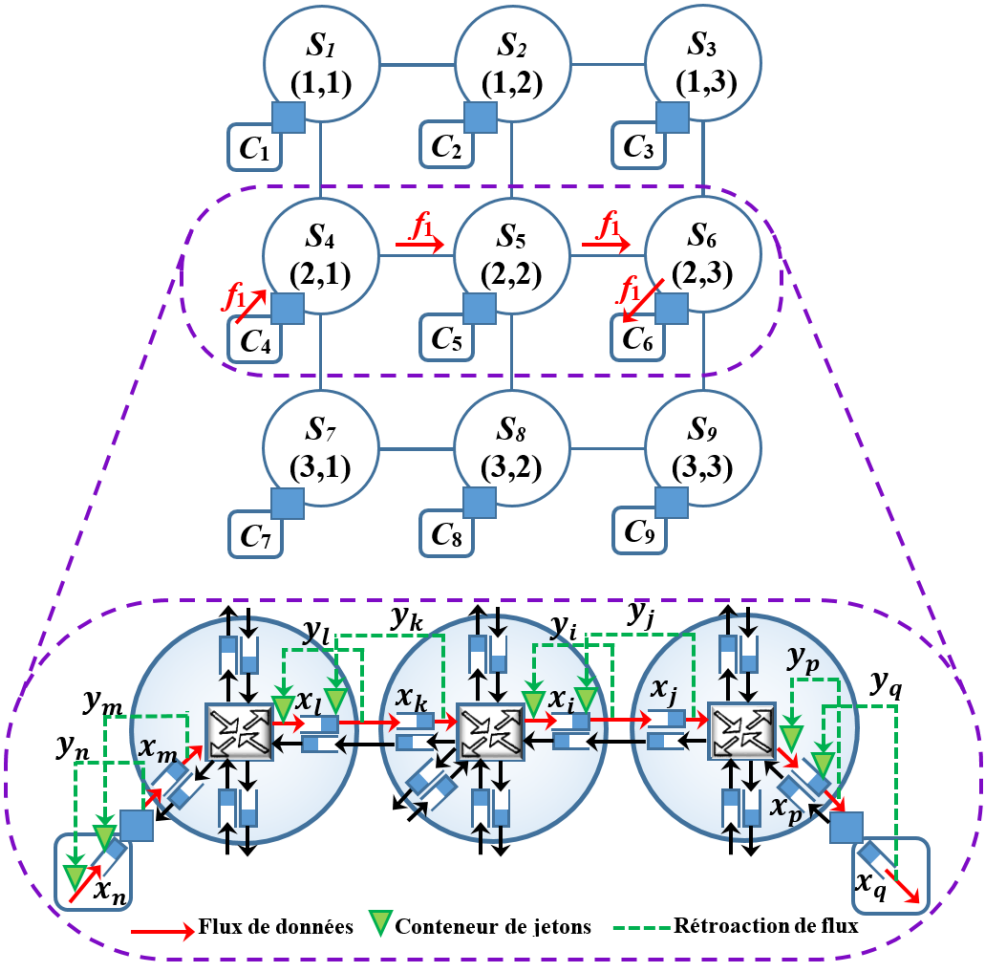


Figure 4.4 : La mise en œuvre du mécanisme d'asservissement

VIII.2. Etude d'évaluation, avec l'utilisation du modèle de trafic *HotSpot*

Dans cette section, nous montrons l'utilisation pratique de l'approche de contrôle de flux. Les résultats sont obtenus à la fois par une évaluation analytique et par simulations. Nous analysant en particulier le taux d'occupation de la mémoire tampon nécessaire pour stocker les flits en attente ou en cours de traitement. La comparaison du débit et de la latence avec et sans contrôle de congestion est également fournie. L'objectif est, tout d'abord, de montrer l'efficacité du mécanisme de contrôle, puis de montrer comment ce dernier peut être utilisé pour élaborer d'autres techniques adaptatives basées sur les informations de la rétroaction.

Dans ce cas d'études, l'architecture d'interconnexion sur puce 2D 4x4 Mesh est considérée (cf. figure 4.5), mais l'approche est applicable à toute autre architecture d'interconnexion sur puce. Nous avons utilisé le modèle de trafic *HotSpot*, dans lequel le cœur C_2 a été sélectionné

pour recevoir l'ensemble du trafic provenant de tous les autres cœurs, comme le montre la figure 4.5. Ce modèle de trafic représente le pire-cas, dans lequel tous les flux de données sont envoyés à un seul cœur. En effet, tous les cœurs sources (sauf C_2) envoient simultanément des données sur le réseau d'interconnexion sur puce, ce qui pourrait conduire à une charge excessive des liens ou à une congestion sur les routeurs, et cela influe la performance globale du réseau. Le mécanisme de routage XY est utilisé pour acheminer les flits entre les cœurs sources vers le cœur *HotSpot*. Dans cette technique de routage, les flits sont tout d'abord acheminés le long de l'axe des abscisses (axe des X), jusqu'à ce qu'ils atteignent la colonne où se trouve la destination, puis ils sont acheminés le long de l'axe des ordonnées (axe des Y), jusqu'à ce qu'ils arrivent à destination.

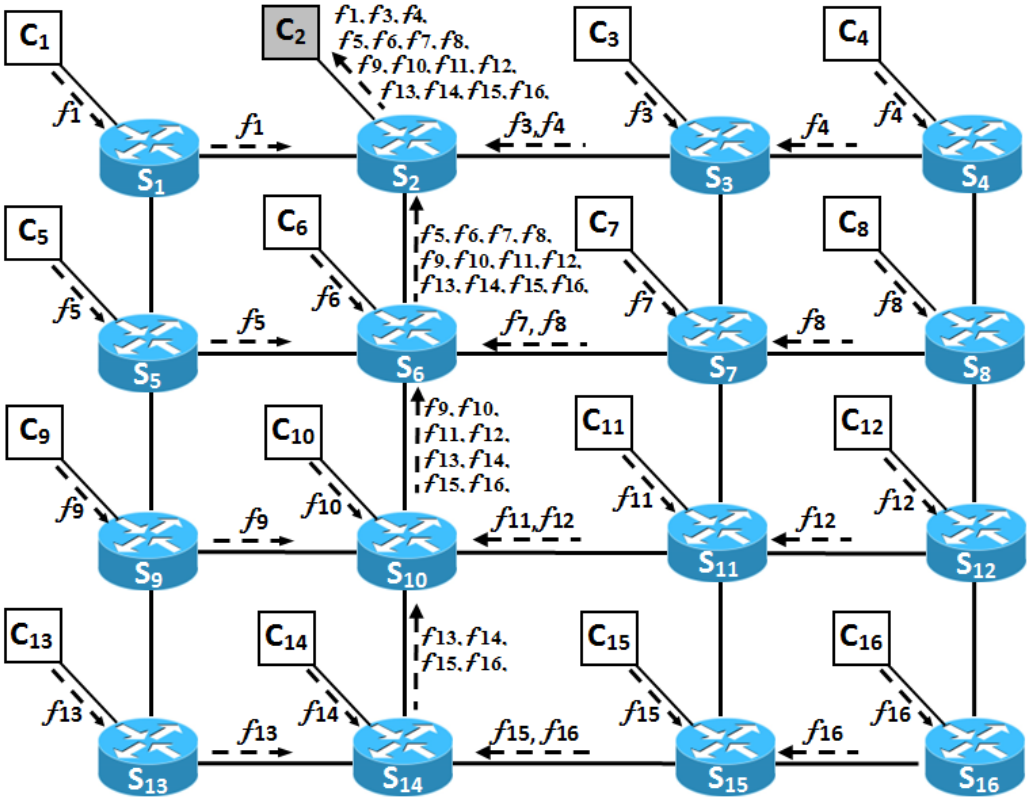


Figure 4.5 : Le modèle de trafic HotSpot sur une architecture d'interconnexion sur puce 2D 4x4 Mesh

L'évaluation analytique est réalisée en utilisant deux outils, le premier outil est Simulink, qui est un environnement de diagramme fonctionnel destiné à la simulation multi-domaine et à l'approche de conception par modélisation Model-Based Design. Il prend en charge la conception et la simulation au niveau système, la génération automatique de code, ainsi que le test et la vérification en continu des systèmes embarqués. Afin de confirmer les résultats obtenus par Simulink, nous avons utilisé un autre outil de système dynamique appelé *Stella* [289], cet outil offre un moyen pratique pour visualiser dynamiquement le fonctionnement réel des systèmes complexes. Les résultats analytiques obtenus par les deux outils, Simulink et Stella, sont similaires à la virgule près.

Les résultats analytiques obtenus sont comparés, par la suite, aux résultats d'une simulation détaillée du système en utilisant le même modèle de trafic. Un simulateur à événement discret, appelé OMNeT++ [264] (cf. chapitre 2, section VI.2.1.2.B.c), est adapté et utilisé pour réaliser cette étude d'évaluation.

Les équations des variables d'état des 16 tampons de l'architecture 2D 4x4 Mesh (cf. figure 4.5) peuvent être écrites, après interaction des flux, comme suite :

$$\dot{x}_1 = -\left(\alpha_{1,2} * \mu_{1,2} * r_1(x)\right) + \left(\mu_{1,1} * r_1(x)\right)$$

$$\dot{x}_2 = -\left(\alpha_{2,2} * r_2(x)\right) + \left(\alpha_{1,2} * \mu_{1,2} * r_1(x) + \alpha_{3,2} * \mu_{3,2} * r_3(x) + \alpha_{6,2} * \mu_{6,2} * r_6(x)\right)$$

$$\dot{x}_3 = -\left(\alpha_{3,2} * \mu_{3,2} * r_3(x)\right) + \left(\alpha_{4,3} * \mu_{4,3} * r_4(x) + \mu_{3,3} * r_3(x)\right)$$

$$\dot{x}_4 = -\left(\alpha_{4,3} * \mu_{4,3} * r_4(x)\right) + \left(\mu_{4,4} * r_4(x)\right)$$

$$\dot{x}_5 = -\left(\alpha_{5,6} * \mu_{5,6} * r_5(x)\right) + \left(\mu_{5,5} * r_5(x)\right)$$

$$\dot{x}_6 = -\left(\alpha_{6,2} * \mu_{6,2} * r_6(x)\right) + \left(\alpha_{5,6} * \mu_{5,6} * r_5(x) + \alpha_{7,6} * \mu_{7,6} * r_7(x) + \alpha_{10,6} * \mu_{10,6} * r_{10}(x) + \mu_{6,6} * r_6(x)\right)$$

$$\dot{x}_7 = -\left(\alpha_{7,6} * \mu_{7,6} * r_7(x)\right) + \left(\alpha_{8,7} * \mu_{8,7} * r_8(x) + \mu_{7,7} * r_7(x)\right)$$

$$\dot{x}_8 = -\left(\alpha_{8,7} * \mu_{8,7} * r_8(x)\right) + \left(\mu_{8,8} * r_8(x)\right)$$

$$\dot{x}_9 = -\left(\alpha_{9,10} * \mu_{9,10} * r_9(x)\right) + \left(\mu_{9,9} * r_9(x)\right)$$

$$\dot{x}_{10} = -\left(\alpha_{10,6} * \mu_{10,6} * r_{10}(x)\right) + \left(\alpha_{9,10} * \mu_{9,10} * r_9(x) + \alpha_{11,10} * \mu_{11,10} * r_{11}(x) + \alpha_{14,10} * \mu_{14,10} * r_{14}(x) + \mu_{10,10} * r_{10}(x)\right)$$

$$\dot{x}_{11} = -\left(\alpha_{11,10} * \mu_{11,10} * r_{11}(x)\right) + \left(\alpha_{12,11} * \mu_{12,11} * r_{12}(x) + \mu_{11,11} * r_{11}(x)\right)$$

$$\dot{x}_{12} = -\left(\alpha_{12,11} * \mu_{12,11} * r_{12}(x)\right) + \left(\mu_{12,12} * r_{12}(x)\right)$$

$$\dot{x}_{13} = -\left(\alpha_{13,14} * \mu_{13,14} * r_{13}(x)\right) + \left(\mu_{13,13} * r_{13}(x)\right)$$

$$\dot{x}_{14} = -\left(\alpha_{14,10} * \mu_{14,10} * r_{14}(x)\right) + \left(\alpha_{13,14} * \mu_{13,14} * r_{13}(x) + \alpha_{15,14} * \mu_{15,14} * r_{15}(x) + \mu_{14,14} * r_{14}(x)\right)$$

$$\dot{x}_{15} = -\left(\alpha_{15,14} * \mu_{15,14} * r_{15}(x)\right) + \left(\alpha_{16,15} * \mu_{16,15} * r_{16}(x) + \mu_{15,15} * r_{15}(x)\right)$$

$$\dot{x}_{16} = -\left(\alpha_{16,15} * \mu_{16,15} * r_{16}(x)\right) + \left(\mu_{16,16} * r_{16}(x)\right)$$

Avec $\alpha_{ij} = 1, \forall\ i,j \in S$, c'est-à-dire que la fraction de flux entrant/sortant à chaque commutateur est de 100%, Ainsi, $r_i(x)$ et μ_{ij} sont données par les équations 4.4 et 4.5 respectivement.

Les résultats obtenus par évaluation analytique sont présentés sur les mêmes graphiques des résultats de simulation, afin de pouvoir les comparer.

VIII.2.1. Paramètres de simulation

La bande passante du lien est configurée à 200 flits/s, ce qui correspond au paramètre μ_i dans le modèle (cf. équation 4.4). L'application est représentée comme un processus parallèle communiquant déjà mappé dans les cœurs. Chaque processus est lié à un générateur de trafic qui injecte des flits selon la loi de distribution de la mémoire (loi de Poisson) avec une moyenne de 10 ms, 12.5 ms, 16.6 ms, 25 ms, 50 ms, qui sont équivalent aux taux d'injection de 100 flits/s, 80 flit/s, 60 flits/s, 40 flits/s, et 20 flits/s, respectivement. La valeur du paramètre a est fixée à "1", la taille du flit est fixée à 8 octets, et le temps de simulation est fixé à 10 secondes. Le temps de service moyen de chaque serveur de file d'attente est fixé à 10 ms, ce qui est équivalent à un taux de soumission de 100 flits/s. Toutefois, lorsque on utilise le contrôle de flux par rétroaction, le nombre de flits qui peuvent être transmis est égal au nombre de jetons disponibles. Le retard δt entre deux messages de rétroaction consécutifs est fixé à 5 ms.

Les résultats d'évaluations analytiques et de simulations sont présentés pour montrer l'efficacité de ce mécanisme pour éviter la congestion dans le NoC. Il est important de rappeler que la technique qui permet d'éviter la congestion dans les NoC a été abordée sous différentes angles, en utilisant par exemple des algorithmes de routage dynamique. Dans les réseaux de communication, deux types de mécanismes de contrôle ont été utilisés, le mécanisme de bout-en-bout et le mécanisme pas-à-pas. Les systèmes de contrôle de flux de bout-en-bout sont implémentés avec succès dans les réseaux à haut débit tels que *ATM* et *Frame Relay* [290]. Par exemple, dans *Frame Relay* le commutateur peut fixer à "1", d'une part le bit de notification d'encombrement *Forward-Explicite* afin d'informer le récepteur que ce paquet a subit une

congestion, et de l'autre part le bit de notification de congestion *Backward-Explicite* pour informer l'expéditeur de la congestion subite sur le lien. Toutefois, dans le mécanisme de contrôle de flux pas-à-pas, chaque nœud envoie des informations de congestion à ses nœuds voisins directs, ceux qui à leur tour, envoient leur état de congestion à leurs nœuds voisins en amont [291]. Les nœuds qui reçoivent l'état de congestion de leurs nœuds voisins peuvent immédiatement prendre des mesures pour réduire la congestion.

VIII.2.2. Résultats de simulation

VIII.2.2.1. Sans l'utilisation du mécanisme de contrôle

Dans cette section, nous illustrons l'influence de la congestion dans le réseau en supposant qu'aucun flit ne sera perdu. Cela signifie que la taille de la mémoire tampon est assez grande pour absorber les différences entre la vitesse du réseau d'interconnexion sur puce, le taux de transmission élevé et l'injection de flux en rafale par les cœurs. Une fois qu'un flit est injecté, soit il atteint sa destination, soit il est mis en attente dans un tampon. La figure 4.6 montre l'occupation moyenne de la mémoire tampon, obtenue par simulation et par évaluation analytique, tout en faisant varier le taux d'injection de la source.

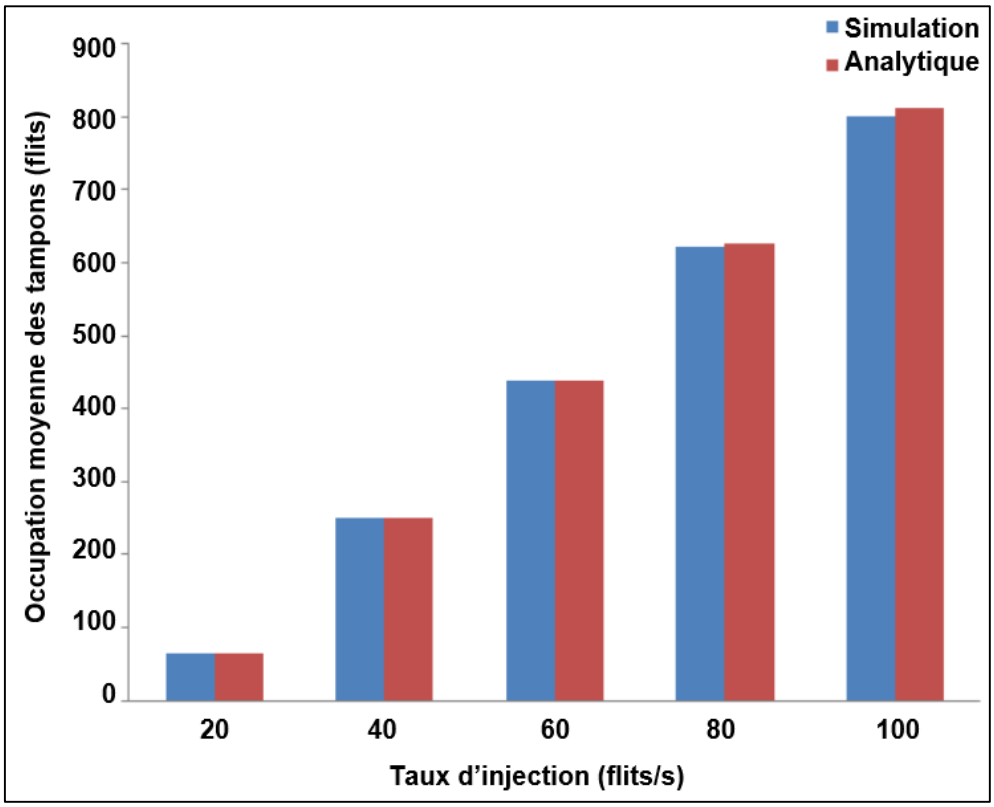


Figure 4.6 : L'occupation moyenne des tampons en fonction du taux d'injection

Cette figure montre que la taille moyenne du tampon augmente en fonction de la hausse du taux d'injection. Le nombre total de flits à l'intérieur des tampons représente la fraction des flits qui n'ont pas atteints leur destination, le cœur *HotSpot* C_2, en raison de la limite de la capacité des liens (la bande passante fixée à 200 flits/s), c'est à dire, le nombre total de flits qui pourrait être soumis par le commutateur S_2 vers son cœur local C_2 est de 2000 flits au cours du temps de simulation (10s). Cette figure montre que l'augmentation du taux d'injection nécessite suffisamment d'espace dans les tampons pour empêcher les pertes des flits. Nous pouvons également constaté que les résultats de simulation sont conformes à ceux obtenus analytiquement.

Le nombre moyen de flits en attente à l'intérieur des tampons peut être utilisé pour calculer la charge de la communication, qui est la valeur relative à la différence entre le taux d'arrivée

et le taux de départ sur l'ensemble des liens. Plus le nombre de flits à l'intérieur du tampon augmente, plus la charge de communication augmente.

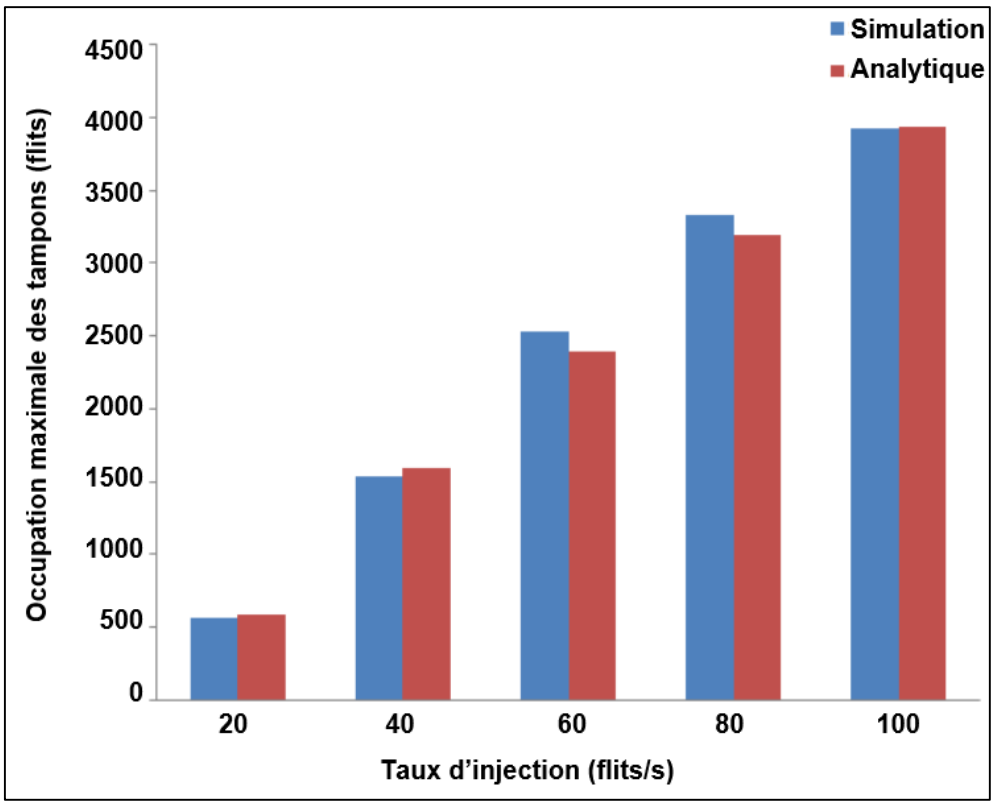


Figure 4.7 : L'occupation maximale des tampons en fonction du taux d'injection

La figure 4.7 montre l'occupation maximale des tampons pour différents taux d'injection. Plus le taux d'injection augmente, plus d'espace tampon est nécessaire pour éviter la perte des flits. La figure 4.8 montre l'évolution de l'occupation des tampons des commutateurs S_2, S_6, S_{10} et S_{14} lorsque le débit d'injection est fixé à 60 flits/s. Seuls ces commutateurs, qui se trouvent sur la même colonne que le cœur *HotSpot* C_2, sont surchargés. En effet, tous les flux à destination du cœur *HotSpot* C_2 en provenance des autres cœurs les traverses (cf. figure 4.5). En outre, l'occupation de la mémoire tampon continue d'augmenter au cours du temps de la simulation tant que les cœurs continuent d'injecter plus de trafic. Si la taille de la mémoire tampon est fixée au moment de la conception, plusieurs flits seront abandonnés lorsque l'espace tampon n'est plus disponible pour absorber le trafic injecté. Plusieurs simulations ont été réalisées avec des taux d'injection fixés à 20 flits/s, 40 flits/s, 80 flits/s, et 100 flits/s. Les résultats obtenus ont montré des comportements similaires à celui où le taux d'injection est de 60 flits/s, comme représenté dans la figure 4.8.

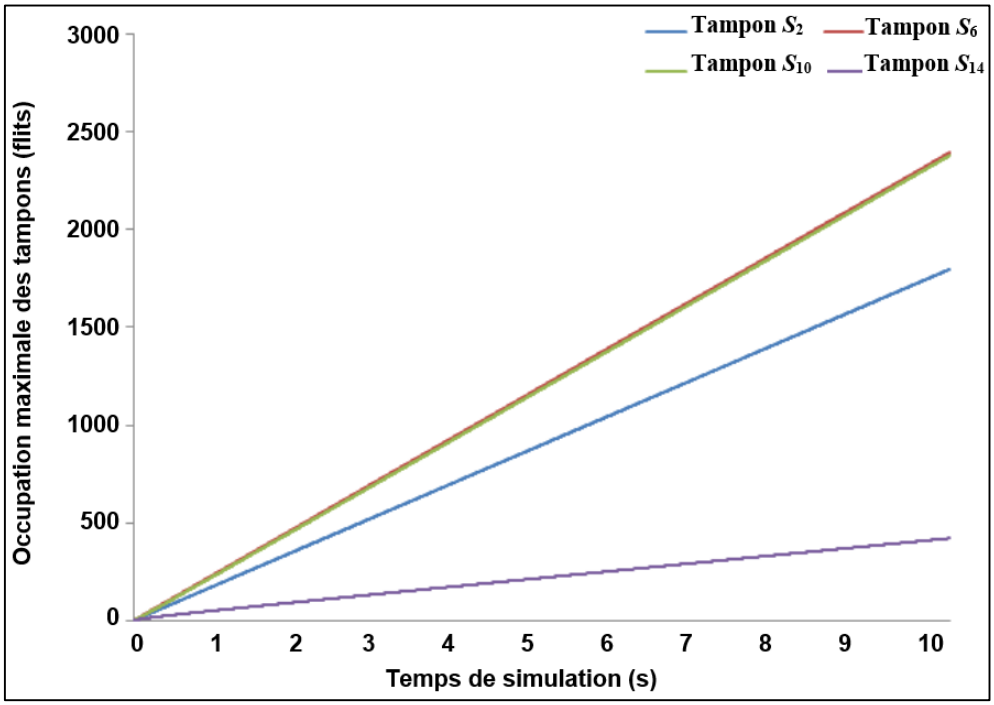


Figure 4.8 : L'occupation maximale des tampons des commutateurs S_2, S_6, S_{10} et S_{14}, résultats obtenus par simulation, le taux d'injection est de 60 flits/s

La figure 4.9 représente le nombre de flit soumis et ceux reçus par le cœur *HotSpot* C_2. Le nombre de flits atteignant C_2 correspond au débit total au cours de l'intervalle $[0, t]$, t est fixé à 10 secondes dans cette évaluation.

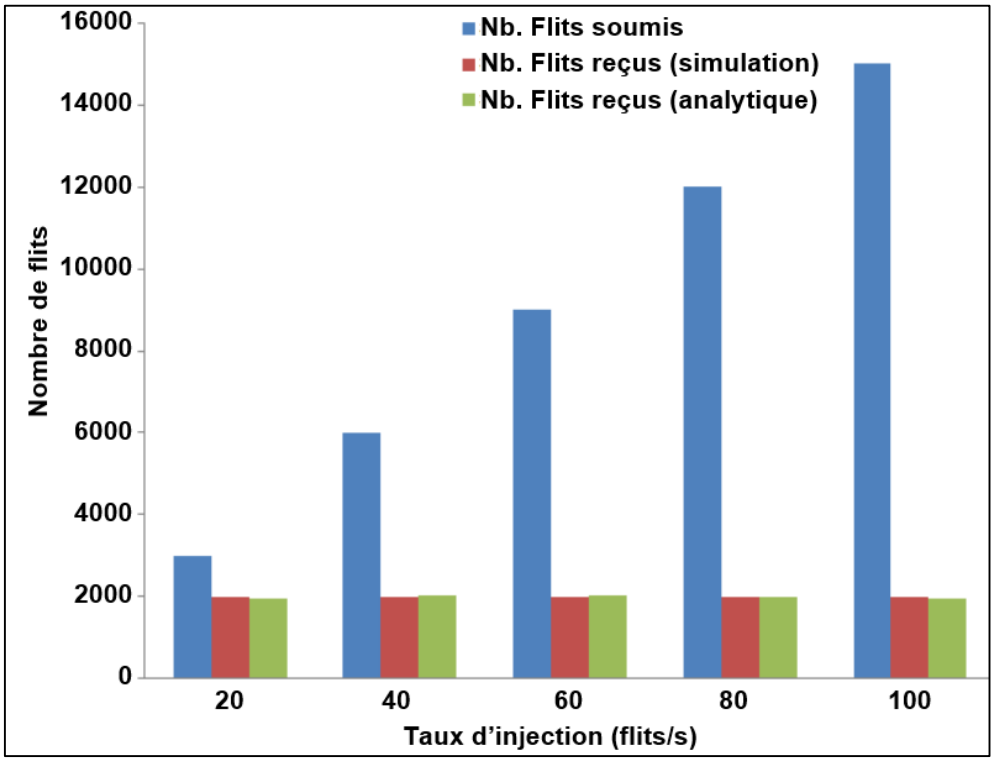


Figure 4.9 : Le nombre de flits soumis et reçus, résultats obtenus par simulation et par évaluation analytique

Comme le montre cette figure, le nombre maximal de flits reçus est inférieure à 2000, ce qui correspond au nombre maximum de flits que le commutateur S_2 pourrait soumettre à C_2, puisque la bande passante du lien de sortie est fixé à 200 flits/s. Par conséquent, sans contrôle, les cœurs sources ne sont pas au courant de la congestion produite sur ce commutateur et ils continuent d'envoyer des flits au même rythme. Les flits non-soumis seront accumulés à

l'intérieur des tampons augmentant ensuite l'occupation de celui-ci. Le nombre total de ces flits est la différence entre le nombre de flits soumis par tous les cœurs et le nombre de flits arrivés au cœur *HotSpot* C_2. En outre, les valeurs obtenues à l'aide de notre modèle analytique sont dans le même ordre de grandeur que les valeurs obtenues par simulations avec un écart de moins de 2%.

VIII.2.2.2. Avec l'utilisation du mécanisme de contrôle

Comme décrit ci-dessus, sans l'utilisation d'un mécanisme de contrôle, les sources ne sont pas au courant du transit du trafic dans le réseau d'interconnexion sur puce (les liens et les commutateurs), ce qui pourrait conduire à une mauvaise performance dans le cas du trafic dense. Par l'intégration de notre mécanisme de contrôle de flux décrit ci-dessus, l'ensemble du système peut être vu comme un système contrôlé par des boucles de rétroaction entre ses cœurs communicants. Plusieurs cas de simulation ont été effectués en faisant varier la taille de la mémoire tampon et du taux d'injection. Du fait de leur comportement similaire, nous montrons que les résultats lorsque la taille de la mémoire tampon est fixée à 20 flits et en faisant varier le taux d'injection. La figure 4.10 montre l'occupation moyenne de chaque tampon. Comme le montre cette figure, plus le taux d'injection augmente, plus d'espace tampon est nécessaire pour stocker les flits en attente. Les valeurs obtenues par simulation correspondent bien à celles obtenues par notre modèle, avec un écart de moins de 5%. Le taux moyen d'occupation du tampon est inférieur à la taille du tampon fixée à 20 flits pour cette évaluation.

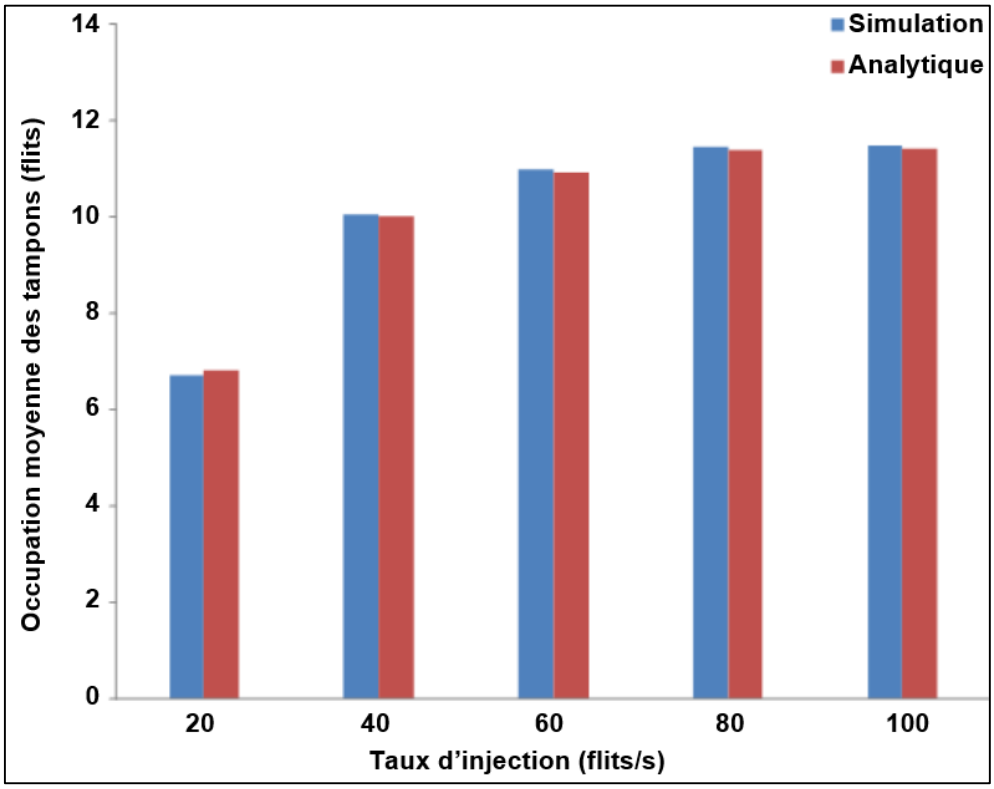


Figure 4.10 : L'occupation moyenne des tampons, avec contrôle, la taille des tampons est de 20 flits

La figure 4.11 représente la taille maximale des tampons (ou de l'occupation de la mémoire tampon) en utilisant différents taux d'injection. Indépendamment des taux d'injection, le mécanisme de contrôle évite l'occupation des tampons d'accroitre au-dessus de la limite déterminée, c'est à dire, l'occupation du tampon est maintenue juste au-dessous de la valeur fixée (ex., 20 flits). Comme décrit dans la figure 4.8, sans utiliser le mécanisme de contrôle de flux par rétroaction, l'occupation des tampons à l'intérieur des commutateurs, S_2, S_6, S_{10} et S_{14}, croît linéairement au fur et à mesure que le taux d'injection augmente.

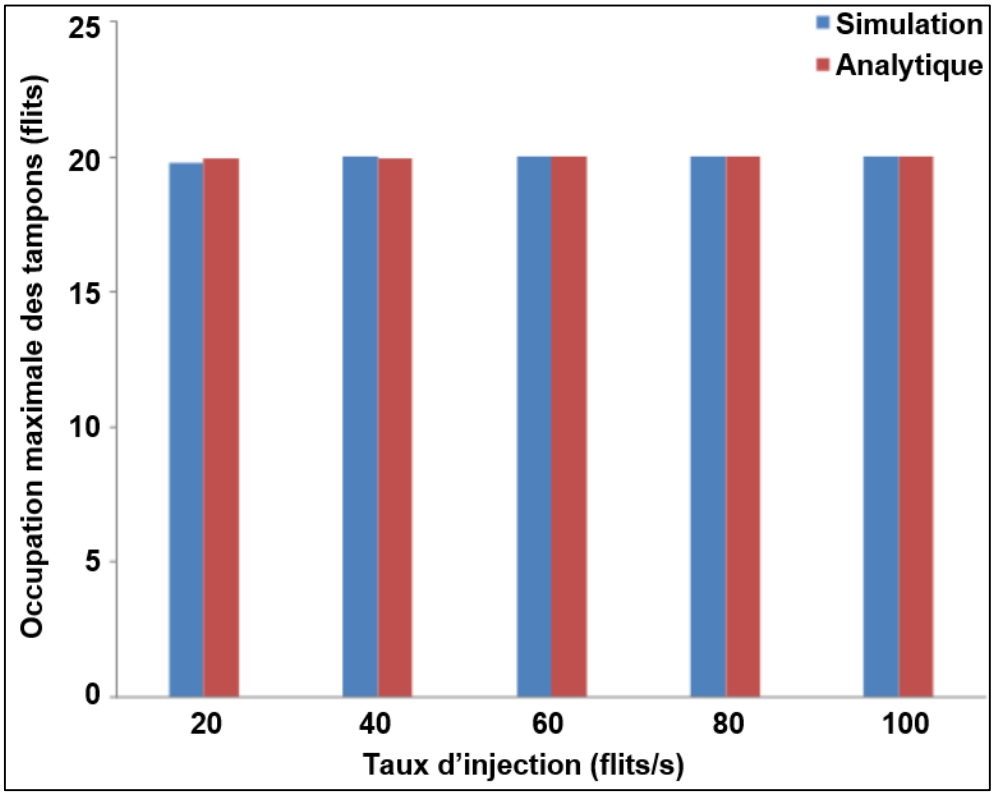


Figure 4.11 : L'occupation maximale des tampons, avec contrôle, la taille des tampons est de 20 flits

Par l'utilisation du mécanisme de contrôle de flux par rétroaction, l'information sur la congestion est transmise en aval pas-à-pas jusqu'aux cœurs sources. L'information de rétroaction aide les commutateurs et les cœurs à adapter leur vitesse de transmission pour éviter le débordement de la mémoire des tampons. Par exemple, la figure 4.12 montre l'occupation maximale des tampons des commutateurs S_2, S_6, S_{10} et S_{14} en utilisant le modèle analytique (avec un taux d'injection de 60 flits/s).

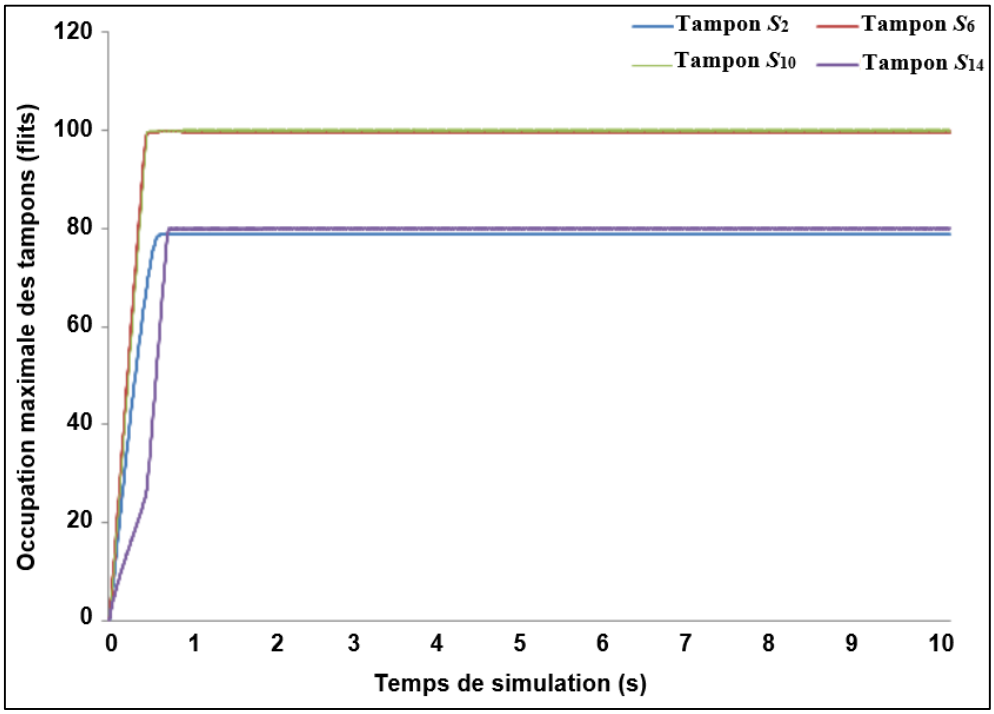


Figure 4.12 : L'occupation moyenne des tampons des commutateurs S_2, S_6, S_{10} et S_{14}, résultats obtenus par évaluation analytique, le taux d'injection est de 60 flits/s

L'occupation maximale des tampons des commutateurs S_6 et S_{10} atteint son maximum de 100 flits, parce que ces commutateurs disposent de quatre tampons d'entrée et un tampon de sortie impliqué dans la transmission de données, chacun a une taille de mémoire tampon fixée à 20 flits. Pour les commutateurs S_2 et S_{14}, l'occupation de tampon atteint son maximum de 80 flits, parce que les deux commutateurs ont trois tampons d'entrée et un tampon de sortie qui reçoivent et transmettent les flits. En d'autres termes, les taux de transmissions des cœurs sources sont adaptés pour que l'accumulation de flits dans les tampons soit inférieure à la taille du réservoir de l'expéditeur (cœur ou commutateur). Les valeurs obtenues par notre modèle correspondent bien à ceux obtenues par simulation, comme le montre la figure 4.13.

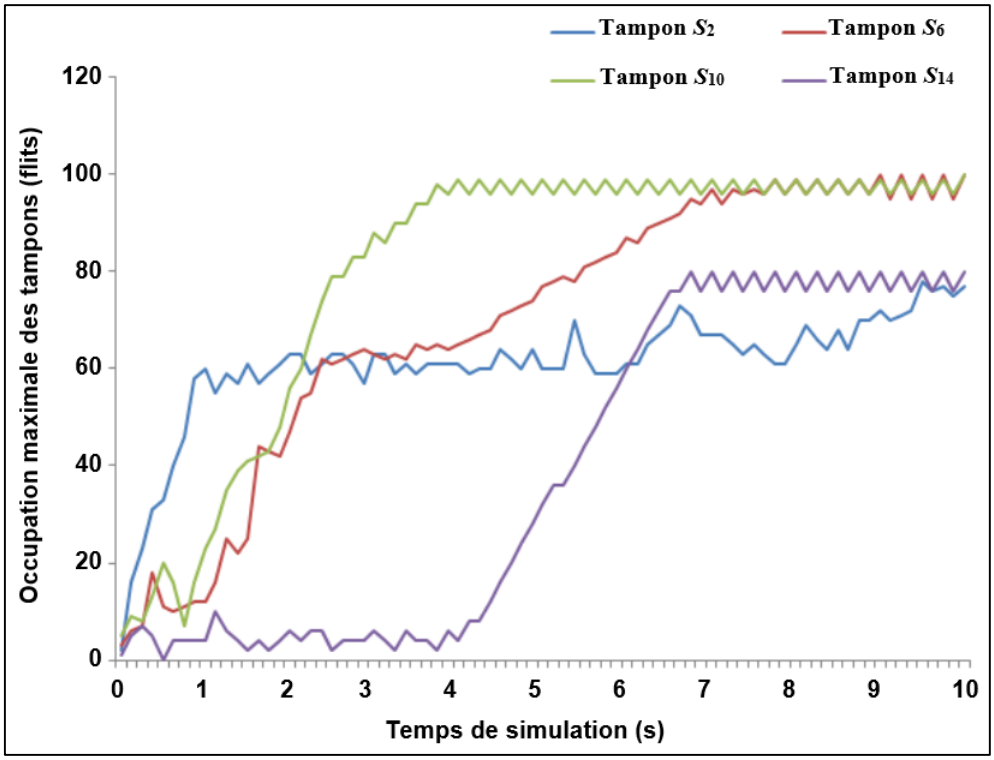


Figure 4.13 : L'occupation moyenne des tampons des commutateurs S_2, S_6, S_{10} et S_{14}, résultats obtenus par simulation, le taux d'injection est de 60 flits/s

Le débit, qui représente le nombre de flits arrivés au cœur *HotSpot* C_2, obtenu par évaluation analytique et par simulation est représenté par la figure 4.14. Comme illustré sur cette figure, le nombre total de flits arrivant à destination atteint approximativement son maximum, qui est de 2000 flits. Du fait que la bande passante de chaque lien est fixée à 200 flits/s, le nombre maximum de flits que le commutateur S_2 peut transmettre au cœur *HotSpot* C_2 durant le temps de la simulation (10s) est de 2000 flits. Dans l'équation 4.4, plus l'occupation de la mémoire tampon augmente, plus le taux de sortie augment sans toutefois dépasser la limite μ_i, correspondant à la bande passante du lien. Nous pouvons aussi constater que les cœurs sources ont adapté leurs taux de transmission pour éviter le débordement des tampons. En outre, les valeurs obtenues par simulations confirment ceux obtenus à l'aide de notre modèle analytique, soit le même ordre de grandeur, avec un écart de moins de 2%. Nous avons également constaté lors de la simulation qu'aucun flit n'était perdu, cela est confirmé par les résultats représentés par la figure 4.14 et la figure 4.10. Le nombre total de flits en attente à l'intérieur des tampons est exactement la différence entre le nombre total de flits soumis (injectés) et le nombre total de flits reçus à destination C_2. Par exemple, prenant le cas du taux d'injection de 20 flits/s, le nombre total de flits en attente à l'intérieur des tampons est de 546 flits, qui est exactement la différence entre le nombre total de flits soumis (2509 flits) et le nombre total de flits reçus (1963 flits). Le même comportement est obtenu en utilisant différentes tailles de tampons.

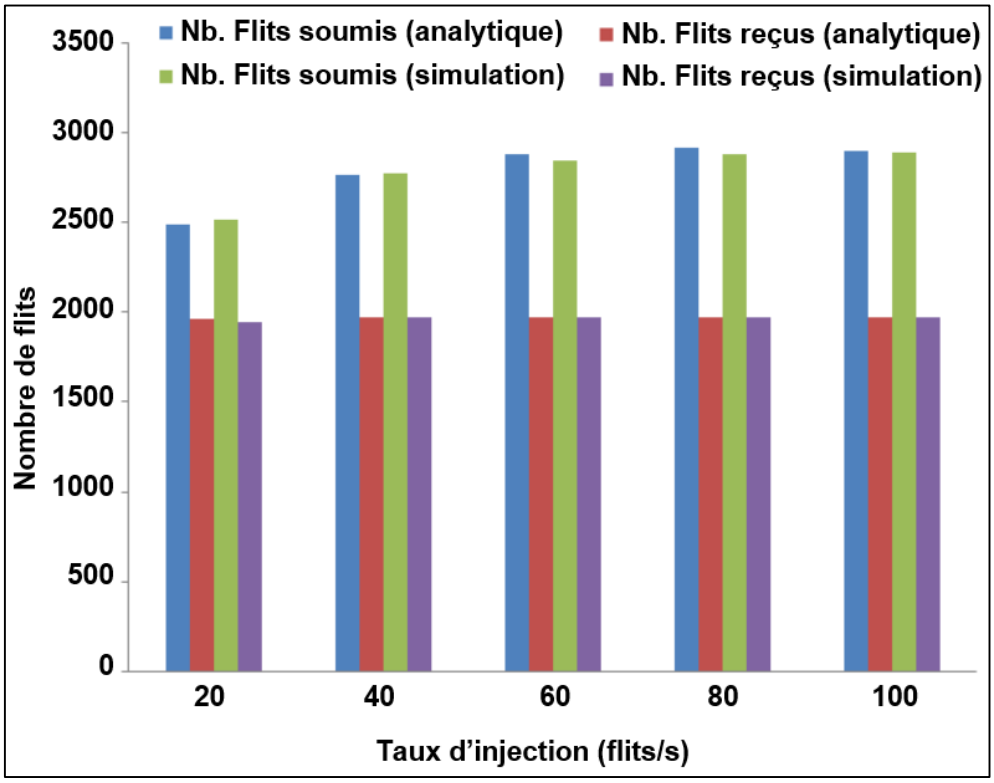


Figure 4.14 : Le nombre de flits soumis et reçus, la taille des tampons est fixée à 20 flits

VIII.2.2.3. Evaluation de la latence moyenne, avec et sans l'utilisation du mécanisme de contrôle

A partir de ces résultats, on peut en déduire que le mécanisme de contrôle de flux par rétroaction empêche les tampons de déborder, c'est-à-dire il maintient l'occupation du tampon toujours sous la limite de sa taille réelle. Pour montrer que cette méthode ne sacrifie pas les performances du réseau sur puce, nous avons comparé la latence sous différents débits d'injection, évaluée par simulations, avec et sans le mécanisme de contrôle, comme le montre la figure 4.15. La latence est le temps qui s'écoule entre le début de l'injection, par le cœur source, des flits dans le réseau d'interconnexion sur puce et leur arrivés, au cœur destinataire. Les résultats montrent que, plus le taux d'injection augmente, plus la latence moyenne augmente, à cause des flits qui sont mis en attente dans les tampons. En d'autres termes, lorsque l'on augmente le débit d'injection, le réseau d'interconnexion sur puce devient plus encombré par un trafic dense, et donc les tampons deviennent pleins forçant les flits à attendre ou à rester en transit, ce qui donc augmente la latence.

En outre, quel que soit le débit d'injection utilisé, dans les résultats de simulation et dans les résultats analytiques, la latence moyenne lors de l'utilisation du mécanisme de contrôle de flux par rétroaction est plus faible, par approximativement 50%, par rapport à la latence lorsque le mécanisme contrôle n'est pas utilisé. La latence lors de l'utilisation du mécanisme de contrôle est également moins sensible à l'augmentation des taux d'injection. Notez que des résultats similaires ont été obtenus pour différentes tailles de mémoire tampon à différentes taux d'injection de trafic. En générale, ces résultats montrent un comportement similaire. Seulement ceux avec la taille de la mémoire tampon fixée à 20 flits sont présentés ici. Du point de vu de la demande, plus le réseau est plein de flits, plus la probabilité que les nouveaux paquets vont être bloqués augmente, donc la latence augmentera de façon exponentielle. Les mesures que nous avons menées ne concernent que les flits reçus au cours de la simulation, puis les sources s'arrêtent d'envoyer des flits lorsque la congestion est détectée.

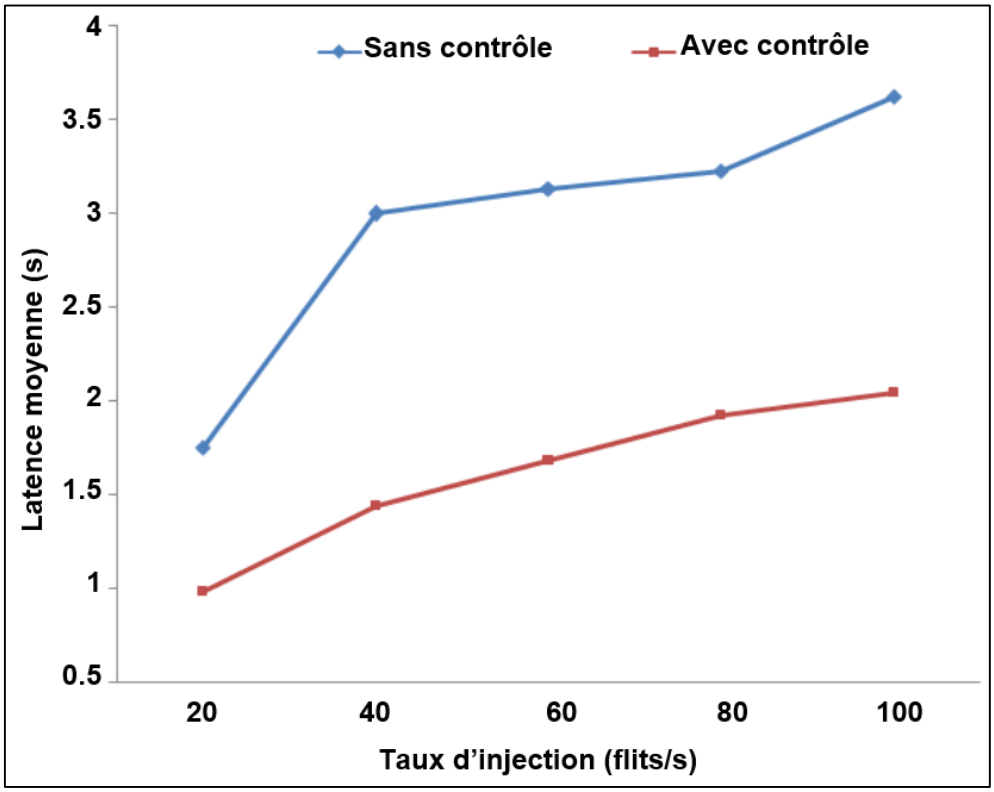


Figure 4.15 : La latence moyenne avec et sans contrôle

Dans les résultats décrits ci-dessus, la taille de la mémoire tampon est fixée à 20 flits, c'est à dire que les ressources tampons sont réparties uniformément entre les commutateurs au moment de la conception. Afin d'étudier le comportement du système lorsque différentes tailles de mémoire tampon et différents taux d'injection sont utilisés (10 flits pour chaque tampon pour les commutateurs S_1, S_4, S_5, S_8, S_9, S_{12}, S_{13} et S_{16} et 20 flits pour les autres commutateurs, S_2, S_3, S_6, S_7, S_{10}, S_{11}, S_{14} et S_{15}), nous avons mené de simulations et des évaluations analytiques. Les résultats sont illustrés dans la figure 4.16 et montrent un comportement similaire, à savoir l'occupation des tampons converge à la taille limité.

Comme décrit ci-dessus, le mécanisme de contrôle permet d'éviter la congestion. Dans le modèle de trafic *HotSpot*, les principaux goulots d'étranglement ont été produits dans les commutateurs S_{14}, S_{10}, S_6 et S_2 qui se trouvent dans la même colonne du cœur *HotSpot* C_2, comme illustré sur la figure 4.6. Le goulot d'étranglement dans ces commutateurs peut être évité en intégrant des techniques supplémentaires pour laisser, autant que possible, les cœurs travailler à pleine capacité.

Nous avons étudié l'utilisation du mécanisme de contrôle pour développer des techniques qui permettent aux commutateurs, soit de changer, au moment de l'exécution, le routage des flits pour éviter les commutateurs congestionnés, soit de changer, au moment de l'exécution, la bande passante des liens de sortie en utilisant les informations collectées sur les boucles de rétroaction. Par exemple, le réacheminement (re-routage) des flits peut conduire à une distribution équilibrée de la charge de trafic dans les applications avec des modèles de trafic *Uniform* et la variation de la bande passante des liens peut résoudre le problème du *HotSpot* lorsque le trafic est dense. Dans le cas du modèle de trafic *HotSpot*, des résultats préliminaires indiquent que l'augmentation de la bande passante des liens de sortie des commutateurs permet d'augmenter le débit de la communication. Comme il nous y est impossible d'augmenter la largeur des liens au moment de l'exécution, nous avons adopté la technique nommée *2X-Link*, qui est conçu pour changer la largeur de la bande passante supportée par l'utilisation de deux liens semi-duplex au lieu d'un lien simplex [89].

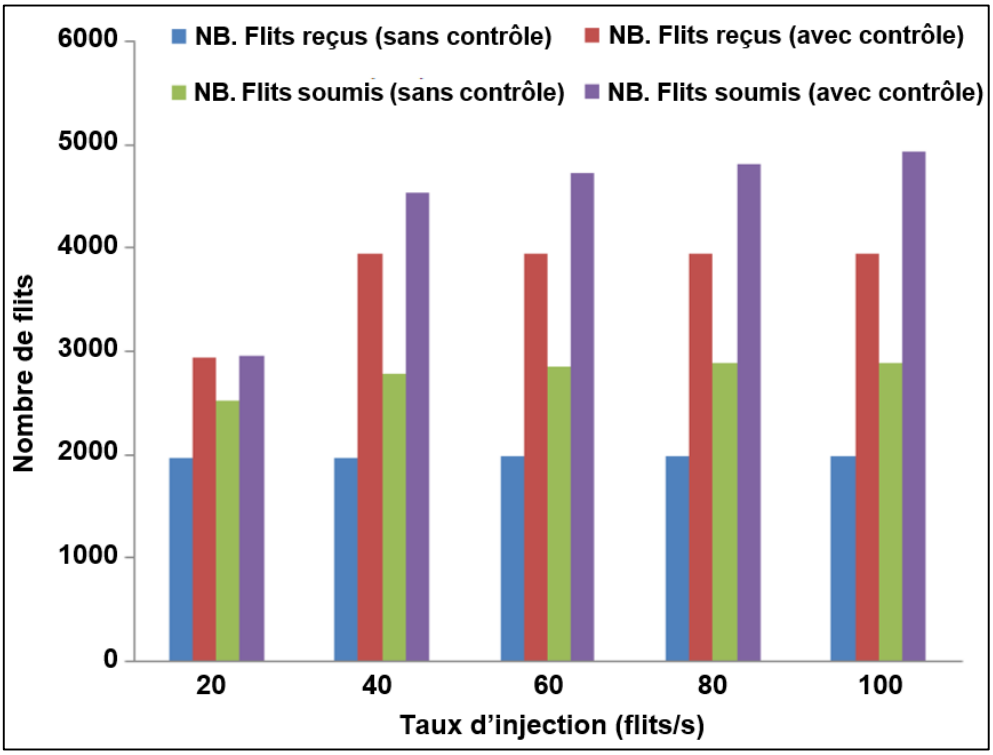


Figure 4.16 : Le nombre de flits soumis et reçus, avec et sans modifier la bande passante, la taille des tampons est fixée à 20 flits

Les résultats préliminaires, obtenus en utilisant ce modèle, représentés dans la figure 4.17, montrent une amélioration de 50% en termes de flits reçus par le cœur *HotSpot* C_2 par rapport à la largeur fixe du lien, comme prévu. Nous pouvons également constaté que les cœurs sources ont augmenté leurs taux de transmission par environs 30%. Le routage adaptatif pourrait également être implémenté, pour permettre aux commutateurs d'utiliser les informations de rétroaction pour réacheminer les flits, et donc éviter que les commutateurs soient congestionnés.

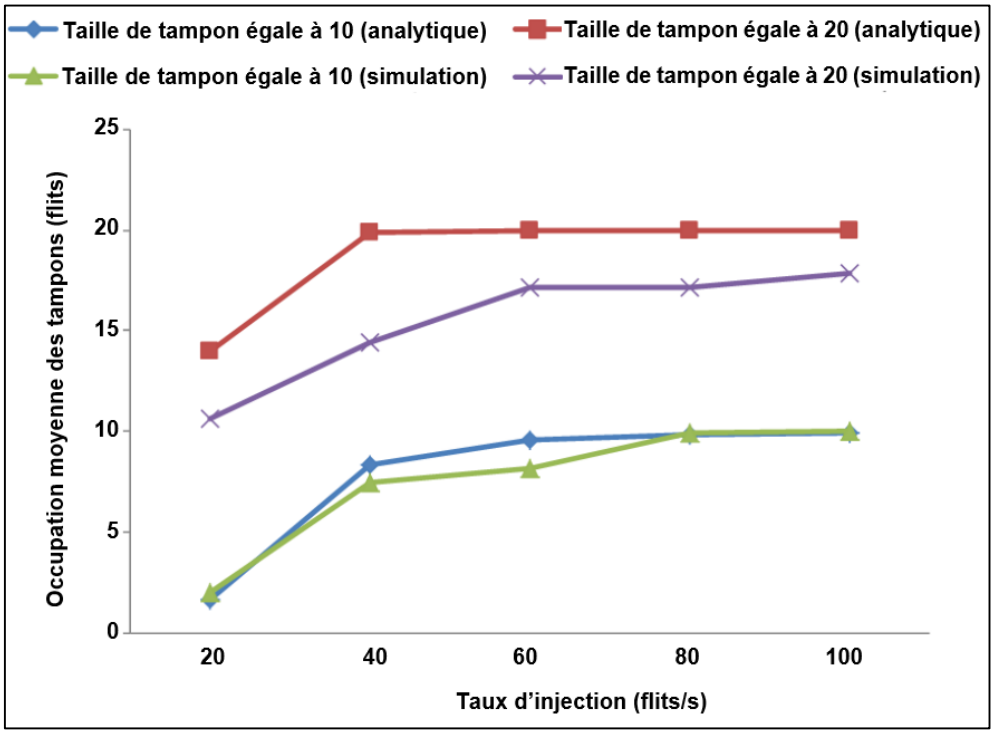


Figure 4.17 : L'occupation moyenne des tampons avec différentes tailles de tampons, 10 et 20 flits

VIII.3. Etude d'évaluation, avec l'utilisation du modèle de trafic *Uniform*

Nous avons également utilisé le trafic *Uniform*, dans lequel des cœurs sélectionnés pour être des sources de trafic sont C_1, C_2, C_3, C_4, C_5, C_9 et C_{13} et les cœurs sélectionnés pour être des récepteurs sont C_4, C_8, C_{12}, C_{13}, C_{14}, C_{15} et C_{16} (cf. figure 4.18).

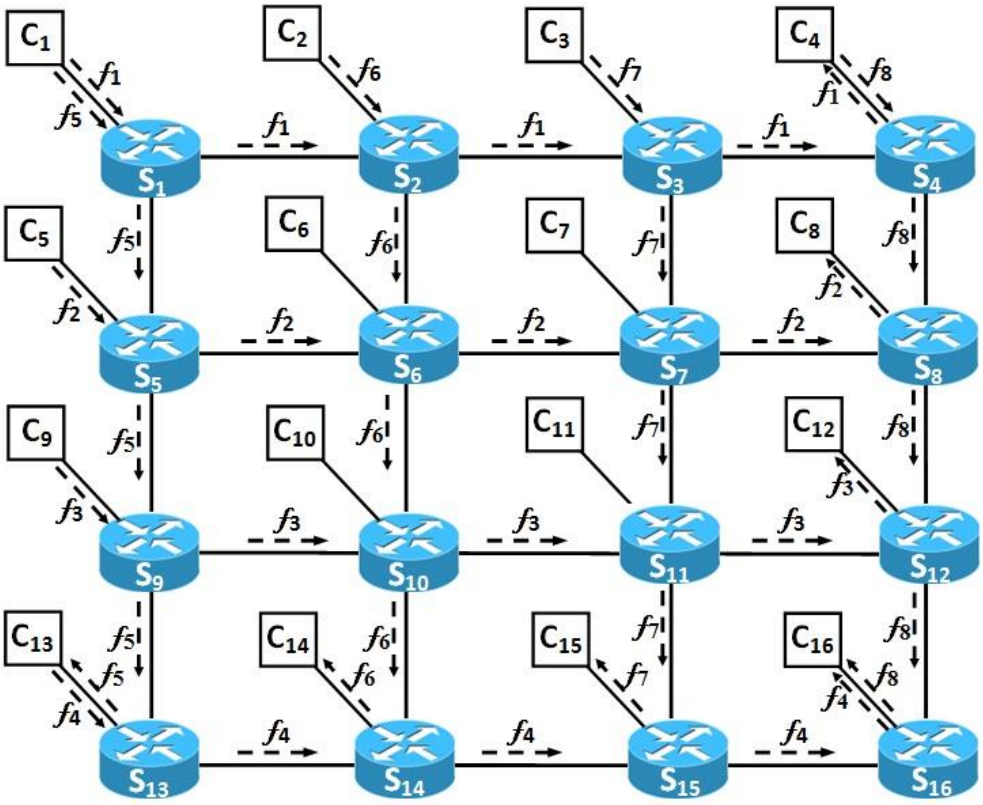


Figure 4.18 : L'architecture d'interconnexion sur puce 2D 4x4 Mesh et les flux de données échangés entre les cœurs sources et destinataires sélectionnés

VIII.3.1. Résultats de simulation

VIII.3.1.1. Sans l'utilisation du mécanisme de contrôle

La figure 4.19 montre la taille moyenne des tampons requise lorsque le taux d'injection est augmenté, nous pouvons voir que plus d'espace est nécessaire pour éviter que les flits soient perdus. Cependant, la taille des mémoires tampon continue d'augmenter au cours du temps de la simulation tant que les cœurs continuent d'injecter plus de trafic. Si la taille des tampons est fixée au moment de la conception, beaucoup de flits seront perdus lorsque l'espace tampons est indisponible pour absorber le trafic injecté.

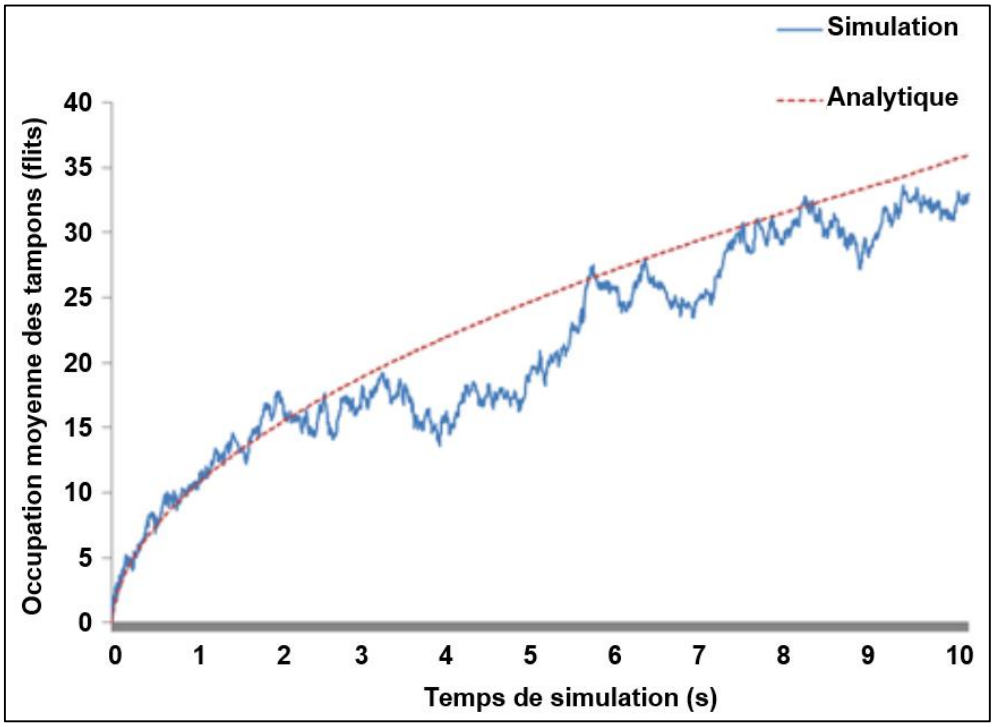


Figure 4.19 : L'occupation moyenne des tampons, sans contrôle, le débit d'injection est de 100 flits/s

VIII.3.1.2. Avec l'utilisation du mécanisme de contrôle

La figure 4.20 illustre la variation de la taille moyenne des tampons obtenue par simulation et par évaluation analytique, lorsque le débit d'injection est fixé à 80 flits/s. Cette figure montre que la taille moyenne des tampons augmente jusqu'à atteindre une valeur fixe (environ 4 flits). Nous pouvons également constaté que les résultats de simulation sont conformes avec le modèle analytique. Plusieurs simulations ont été menées, mais leurs résultats ne sont pas tous présentés, parce que leurs comportement est similaire que celui présenté dans la figure 4.20, c'est à dire, les tailles des tampons augmente rapidement pour atteindre des valeurs fixes, puis elle se stabilise sur cette valeur.

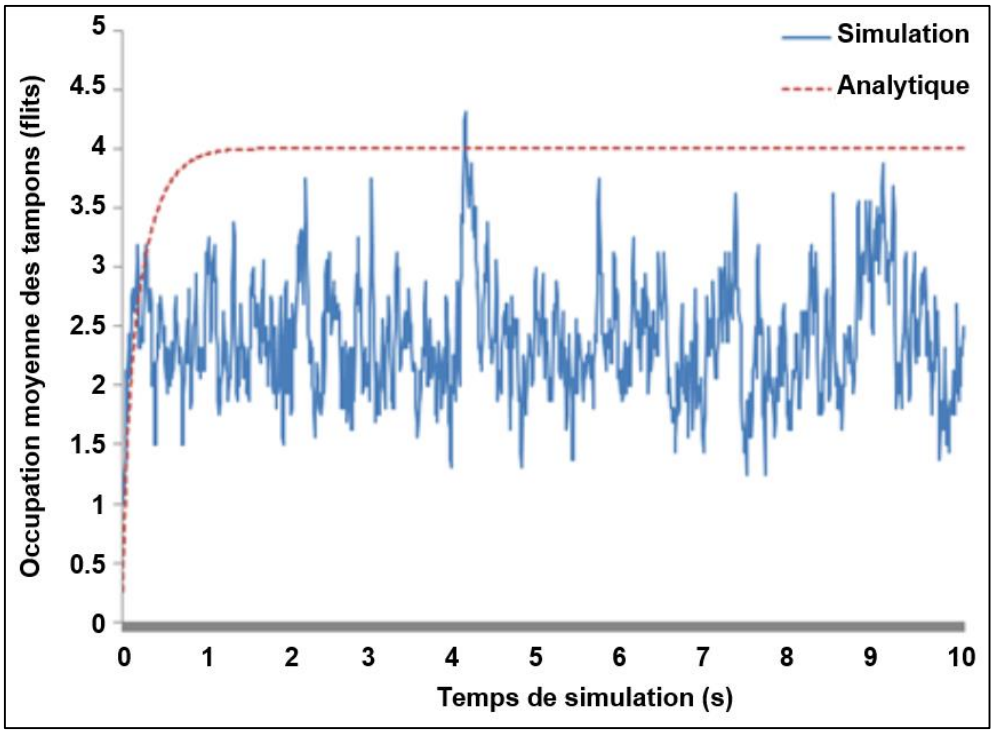


Figure 4.20 : L'occupation moyenne des tampons, avec contrôle, le débit d'injection est de 80 flits/s

Plusieurs autres cas de simulation ont été réalisés en faisant varier la taille des tampons et les taux d'injection, du fait de leur comportement similaire, nous ne présentons que les résultats lorsque les tailles des tampons sont fixées à 4 et à 10 flits et avec un taux d'injection de 100 flits/s.

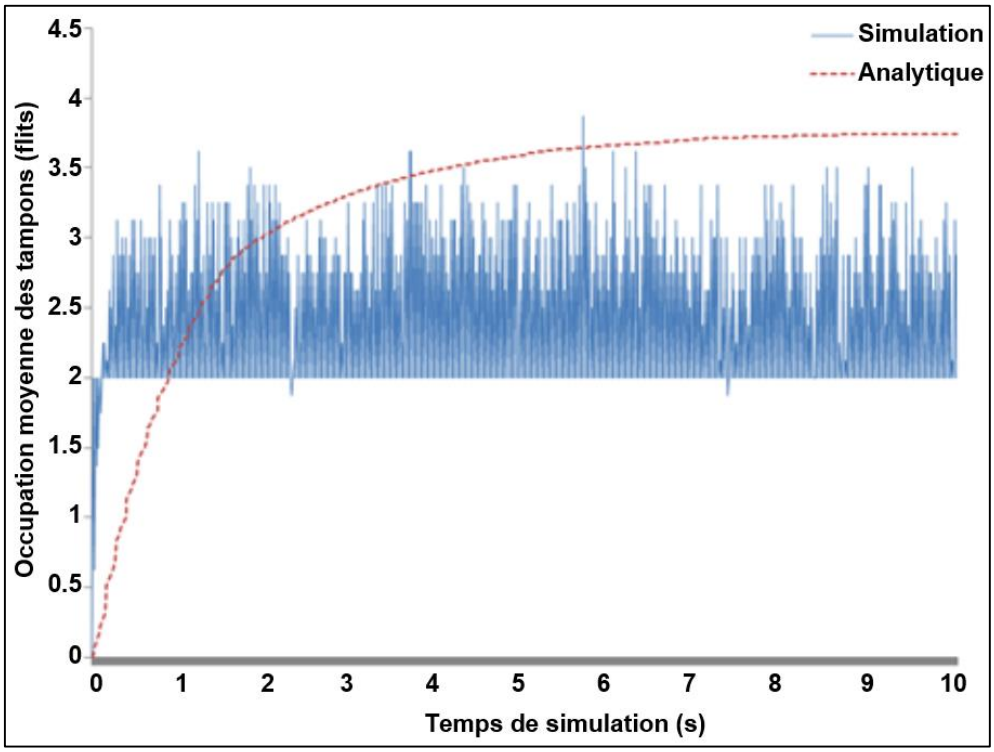


Figure 4.21 : L'occupation moyenne des tampons, avec contrôle, la taille des tampons est fixée à 4 flits et le taux d'injection est de 100 flits/s

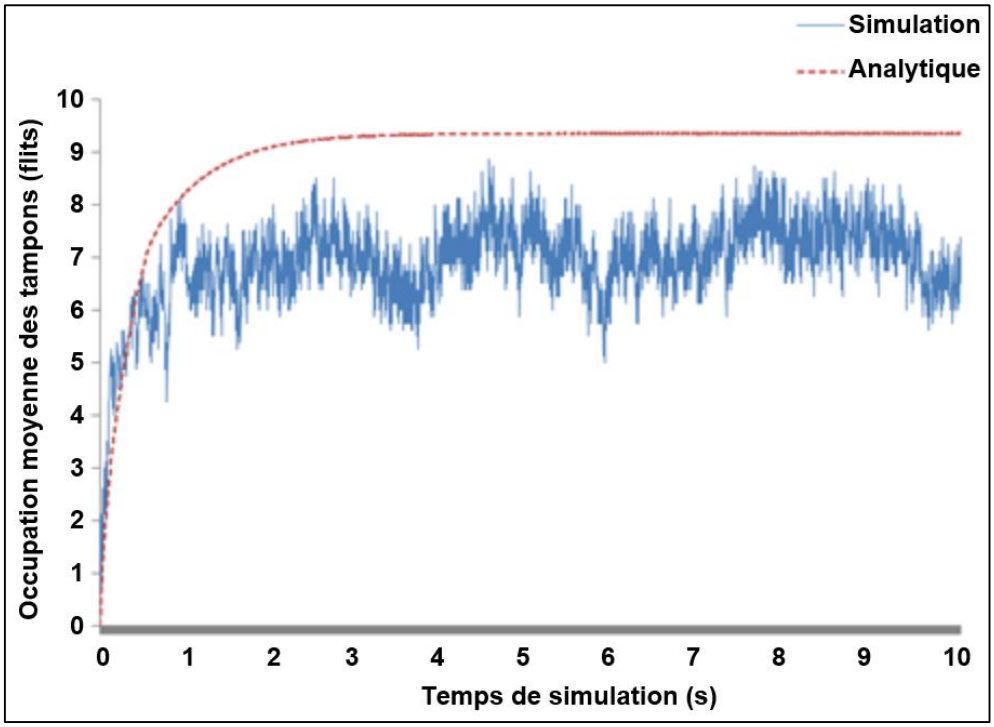


Figure 4.22 : L'occupation moyenne des tampons, avec contrôle, la taille des tampons est fixée à 10 flits et le taux d'injection est de 100 flits/s

Les figures 4.21 et 4.22 montrent que le mécanisme de contrôle empêche que le nombre de flits en attente n'augmente au-dessus de la limite fixée, sachant que les tailles des tampons sont maintenues seulement au-dessous des valeurs 4 et 10 flits, respectivement. Comme décrit dans

la figure 4.19, sans l'utilisation du mécanisme de contrôle de flux par rétroaction, la taille des tampons ne cesse d'accroître linéairement dans le temps. Avec l'utilisation du mécanisme de contrôle, les informations sur la congestion des tampons sont transmises, pas-à-pas, jusqu'à atteindre les sources de trafic, qui adaptent leurs taux d'injection pour éviter le débordement des tampons.

VIII.4. Synthèse

Dans ce chapitre, l'outil de modélisation par réseau à compartiments est utilisé pour analyser et éviter la congestion, en incluant un mécanisme de contrôle de flux par rétroaction pour garantir le non-débordement des tampons, définis dans la phase de conception. Des études analytiques et des simulations ont été menées et les résultats présentés montrent l'efficacité de cette approche pour éviter la congestion.

VI. Conclusions & Perspectives

IX.1. Conclusions

Aujourd'hui, les systèmes embarqués sur puce SoCs sont devenus de plus en plus complexes grâce à l'évolution de la technologie des circuits intégrés. Les performances de ces SoCs dépendent fortement de l'architecture d'interconnexion sur puce et du protocole de communication entre les cœurs IPs. Avec la technologie d'intégration croissante, la conception d'un SoC efficace est critique pour exploiter pleinement le nombre et la puissance de traitement des cœurs IPs dans un même circuit. La quantité de données échangées dans les systèmes est ainsi augmentée. Dans ce contexte, le paradigme NoC est devenu la solution privilégiée pour la communication dans les SoCs complexes. Cependant, faute de méthodologies et d'outils d'expérimentation et d'aide à la conception adaptés, l'évaluation de performance de ces réseaux sur puce constitue un défi majeur pour l'industrialisation de ces systèmes. Des études récentes ont montré que pour améliorer les performances du NoC, l'architecture de celui-ci pouvait être personnalisée, soit au moment de la conception, soit au moment de l'exécution.

L'objectif principal de cette thèse était donc d'implémenter de nouvelles approches pour améliorer les performances des NoCs, notamment la latence, le débit, la consommation d'énergie, la surface silicium et la simplicité de mise en œuvre. Donc, nous avons précisé plusieurs problématiques liées à la conception et la communication des NoCs. Pour pallier à ces problématiques, nous avons proposé trois approches d'optimisation de performances de l'architecture NoC.

Dans le troisième chapitre, deux approches de personnalisation de la topologie de NoC sont présentées. La première approche est basée sur l'insertion de liens pour la personnalisation du NoC sans tenir compte du trafic des applications qui devraient s'exécuter dessus. L'évaluation analytique de cette approche a été réalisé en utilisant cinq architectures NoC (2D Mesh, 2D Torus, X-Mesh, Spidergon et WK), les résultats de l'évaluation analytique ont montré que ces architectures convergent vers les mêmes performances physiques après l'ajout d'environ 30 liens supplémentaires. Avec un NoC ayant 50% de liens (liens de base plus liens ajoutés) par rapport à un NoC complètement connecté (100%), nous avons constaté analytiquement une amélioration d'environ 70% des paramètres physiques (distance moyenne et dégrée de clusterisation) sur tous les NoCs. Ces résultats nous ont permis de proposer une nouvelle topologie commune dénotée FracNoC (k), elle a la particularité d'être fractale et récursive. Des évaluations par simulation sont conduites, d'une part pour comparer les cinq architectures considérées précédemment pour l'évaluation analytique avec la FracNoC nouvellement proposée, et d'autre part pour évaluer l'impact de l'approche d'ajout de lien sur les performance de communication (débit, latence, charge des liens, et énergie) des six architectures simulées. Les résultats de simulation prouvent d'une part l'efficacité de l'approche d'insertion de liens qui permet d'améliorer progressivement les performances tout en ajoutant plus de liens, et d'autre part le bon placement de la nouvelle topologie parmi les autres. En moyenne sur l'ensemble des NoC simulés, avec l'ajout de 30 liens (un NoC à 50% par rapport à un NoC complètement connecté), nous avons constaté une amélioration de 30% sur la latence moyenne, 15% sur la charge des communications et 24% sur la consommation d'énergie. Le débit étant constant, il montre que les flits injectés par les IPs sources sont arrivés aux IPs de destination sans aucune perte.

La deuxième approche concerne l'exploration de l'espace de conception, elle est basée sur la modélisation du trafic NoC par réseau à compartiments. Le but est de permettre aux concepteurs, à un stade très tôt du processus de conception, d'analyser le trafic rapidement et d'allouer judicieusement l'espace de mémoire tampon, requis pour chaque canal d'entrée/sortie, pour la conception de routeurs/commutateurs NoC performants. C'est ainsi que le concepteur peut concevoir des routeurs NoC optimisés et adaptés pour une application spécifique donnée. Le but de cette approche est de fournir un outil d'analyse basé sur la

modélisation par réseau à compartiments pour personnaliser le NoC selon une application cible. Cette approche a été évaluée analytiquement et par simulation en considérant un NoC de type 2D 4x4 Mesh et en utilisant le même modèle de trafic (*Uniform*) avec une variation du taux d'injection de 80, 60, 40, et 20 flits/s. Les résultats de simulation sont conformes au modèle analytique. Ils présentent un écart de moins de 10% en moyenne. Cependant, ils montrent que lorsque le débit d'injection augmente, plus d'espace de mémoires tampons est nécessaire pour éviter la perte des flits arrivants à ce commutateur. Nous avons également comparé le modèle de *Network Calculus* avec notre modèle de réseau à compartiments, et les résultats montrent que notre modèle de réseau à compartiments fournit une estimation plus précise de la taille des tampons par rapport au *Network Calculus*, qui fournit une analyse du pire-cas (avec un écart en moyenne de 15%). Par ailleurs, notre méthode est comparée à la méthode qui attribue uniformément l'espace de mémoire tampon et les résultats montrent que notre méthode réalise des performances similaires tout en utilisant seulement 70% du budget de ressources.

Dans le quatrième chapitre, un mécanisme de contrôle de flux par rétroaction a été proposé pour garantir le non-débordement des tampons dans le NoC. Ce mécanisme est basé sur la modélisation de flux par réseau à compartiments pour analyser et éviter la congestion au moment de l'exécution. Des évaluations analytiques et des simulations ont été menées en considérant, d'une part un NoC de type 2D 4x4 Mesh avec un modèle de trafic *HotSpot* (le cœur C_2 a été désigné récepteur), et d'autre part un NoC de type 2D 4x4 Spidergon avec un modèle de trafic *Uniform*. Les résultats présentés montrent l'efficacité de cette approche pour éviter la congestion. Les résultats préliminaires obtenus en utilisant ce mécanisme (cf. figure 4.17), avec le modèle de trafic *HotSpot*, montrent une amélioration de 50% en termes de flits reçus par le cœur *HotSpot* C_2 par rapport à la largeur fixe du lien. Nous avons également constaté que les cœurs sources ont augmenté leurs taux de transmission par environs 30%. En outre, le mécanisme de contrôle réalise de meilleures performances en ce qui concerne la latence moyenne de paquets avec un gain moyen de 46% sur l'ensemble des taux d'injection, 20, 40, 60, 80 et 100 flits/s. Il est également démontré que les valeurs réelles obtenues par simulation correspondent bien à ceux prédits en utilisant l'évaluation analytique avec un écart de moins de 2%. Les résultats obtenus de l'autre cas d'étude (un NoC en Spidergon avec un trafic *Uniform*) montrent que sans l'utilisation du mécanisme de contrôle de flux par rétroaction, la taille des tampons ne cesse d'accroître linéairement dans le temps. Ainsi, l'utilisation du mécanisme de contrôle de flux empêche que le nombre de flits en attente n'augmente au-dessus de la limite de la taille de tampons fixée au préalable, sachant que les tailles des tampons sont maintenues seulement au-dessous des valeurs 4 flits (cf. figure 4.21) et 10 flits (cf. figure 4.22). Le mécanisme de contrôle de flux par rétroaction pourrait être utilisé pour développer des techniques dynamiques qui intègrent des informations sur les congestions pour aider à décider, par exemple de re-router les flits, au moment de l'exécution, dans les zones du NoC congestionnées, ou bien de changer dynamiquement la bande passante des liens des commutateurs encombrés.

IX.2. Perspectives

De manière générale, nous visant dans nos futurs travaux à court termes de faire des simulations et des évaluations analytiques supplémentaires afin de perfectionner et de comparer nos approches proposées dans cette thèse avec d'autres approches existantes dans la littérature.

Pour la première approche, nous voudrions utiliser un algorithme de routage dynamique afin d'améliorer davantage les performances des architectures personnalisées par l'ajout de liens. Un algorithme de routage dynamique est actuellement en cours de développement sous NIRGAM. Cet algorithme de routage est basé sur l'algorithme de *Dijkstra* pour le calcul du plus court chemin, tout en utilisant les informations de congestion dans les tampons du NoC, afin de modifier les itinéraires des flits dynamiquement, au moment de l'exécution, pour surpasser les zones du NoC congestionnées.

Dans la deuxième approche, d'autres applications avec des flux de données complexes (ex., les modèles de trafic *Uniforme* et *Hotspot*) seront utilisées. Parce que ces modèles représente le pire-cas, ils provoquent des congestions et des goulots d'étranglement dans certains commutateurs en raison de l'insuffisance des ressources, et conduisant donc à des performances médiocres du NoC (vu que les ressources disponibles doivent être partagés entre des connexions multiples).

En ce qui concerne la troisième approche, d'autres architectures NoC seront considérées. En outre, la superficie et la consommation d'énergie seront mesurées pour montrer l'impact de ce mécanisme de contrôle de flux de bout-en-bout dans les grandes NoCs, ainsi quel serait le compromis entre les métriques de performance et de coût. D'autres mécanismes de contrôle de flux basés sur les crédits seront aussi étudiés et comparés à notre mécanisme de contrôle de flux par rétroaction. Nous étudions également l'utilisation du mécanisme de rétroaction pour développer des techniques dynamiques qui intègrent l'information de congestion pour l'aide à la décision, comme re-router les flits congestionnés ou bien modifier la bande passante des liens des commutateurs congestionnés en utilisant les informations recueillies à partir des boucles de rétroaction. Par exemple, le réacheminement de flits congestionnés peut conduire à une distribution équilibrée de la charge de trafic dans les applications avec le modèle de trafic *Uniform*, et en faisant varier la bande passante des liens, on peut résoudre le problème du modèle de trafic *HotSpot*, lorsque le trafic est dense. Cette approche est actuellement en développement et intégration dans le simulateur NIRGAM pour montrer comment l'information de rétroaction (de congestion) pourrait être utilisée pour permettre aux commutateurs de sélectionner automatiquement le meilleur mode d'évitement de la congestion. Comme il s'agit d'un simulateur à événement discret basé sur SystemC, le simulateur à cycle précis NIRGAM fournit un soutien substantiel à expérimenter avec des options de conception en termes d'algorithme de routage, de générateurs de trafic synthétiques et d'applications sur diverses topologies.

IX.2.1. Routage dynamique dans NIRGAM

La plupart des algorithmes de routage existant dans NIRGAM sont statiques (ex., XY, Source, etc.), mais si quelques-uns sont dynamiques ou semi-dynamiques (ex., DayD, Odd-Even), le problème reste entier : ces algorithmes sont adaptés uniquement à des topologies régulières spécifiques (ex., 2D Mesh, 2D Torus, …).

Le routage statique présente un certain nombre d'inconvénients :

- Toute modification de la topologie requiert une intervention manuelle pour adapter les tables de routage,
- Non adaptés aux réseaux de grandes tailles,
- Temps d'indisponibilité en fonction du délai de prise en compte par l'administrateur de la panne d'un équipement ou d'une modification accidentelle de la topologie,
- Consommation d'énergie importante,
- Perte de paquets en cas de congestion.

Pour pallier à ces inconvénients, il est nécessaire de mettre en place dans NIRGAM un algorithme de routage entièrement adaptatif pour optimiser la consommation d'énergie et réduire la congestion. Plus précisément, le routage doit être adaptatif à la fois à la consommation d'énergie disponible et aux contraintes du trafic du réseau. L'algorithme mis en œuvre doit être générique et indépendant de la topologie du réseau. L'algorithme de routage est à implémenter en utilisant C++, le compilateur g++ et la plate-forme LINUX (Ubuntu 10.10)

Nous avons tout d'abord pensé à utiliser un algorithme basé sur la théorie des colonies de fourmis, pour pouvoir trouver le chemin le plus court entre deux nœuds. Mais en réalité, cet algorithme est plus adapté aux réseaux parallèles, dont on ne connaît pas l'architecture à l'avance. Or, dans NIRGAM, nous avons déjà connaissance de la topologie du réseau!

Puisqu'on connait le réseau et ses différentes caractéristiques, nous pouvons d'avance créer des tables de routage pour chaque nœud.

Dans une stratégie de routage adaptatif, l'itinéraire des flux est décidé sur une base par-saut. C'est-à-dire, à chaque nœud on fait appel à l'algorithme de routage afin de déterminer le port de sortie à prendre. Les systèmes adaptatifs impliquent un arbitrage dynamique et des mécanismes de sélection du saut suivant, c'est à dire sur la base de la congestion locale du lien. L'idée est que notre algorithme de routage choisi toujours le chemin le moins encombré vers la destination, en planifiant la trajectoire optimale. Il faut prendre en compte non seulement le plus court chemin mais aussi celui qui permet d'éviter les zones congestionnées. Nous notons que la congestion dépend de l'état de fonctionnement du réseau NoC, intensité et conditions de trafic, qui changent au moment de l'exécution.

Le développement de cet algorithme de routage dynamique s'inscrit dans les perspectives de nos travaux, plus particulièrement pour l'approche d'ajout de liens (topologies irrégulières). Cet algorithme est actuellement sous-développement dans le simulateur NIRGAM. La partie de calcul du plus court chemin est déjà terminée (basée sur l'algorithme de *Dijkstra*) et intégrée dans NIRGAM. La partie la plus intéressante reste à faire, il s'agit d'intégrer les informations de congestion des tampons dans le calcul du prochain saut, afin de donner au final un itinéraire optimal, qui surpasse les tampons congestionnés. Notre algorithme de routage dynamique doit être en mesure de découvrir les congestions et d'effectuer le calcul du plus court chemin en même temps.

IX.2.2. Les systèmes sur puce autonomes

Dans nos futurs travaux, nous allons aussi introduire, dans le processus de la conception matériel et logiciel du NoC, un mécanisme d'automatisation, appelé mécanisme *Self-CHOP* pour l'auto-configuration, l'auto-réparation, l'auto-optimisation et l'auto-protection (en anglais, *Self-Configure*, *Self-Heal*, *Self-Optimize*, and *Self-Protect*, d'où le nom *Setf-CHOP*) [82]. Ce mécanisme d'automatisation sera introduit à chaque niveau de l'infrastructure du NoC, le niveau application, le niveau communication et le niveau physique (cf. figure 5.1). Plus précisément, l'infrastructure NoC peut être considérée comme une hiérarchie logique du système à niveaux. A chaque niveau, des mécanismes de suivi et de supervision sont nécessaires pour alimenter le mécanisme d'automatisation correspondant. Cela permet aux composantes du niveau supérieur d'allouer et de superviser les tâches des composants du niveau inférieur. Ces composants de niveau inférieur surveillent de leur part les cœurs IPs locaux pour s'adapter aux comportements de fonctionnement des tâches. Ils remontent également aux composants du niveau supérieur les modifications apportées à leur niveau. Cela nécessite des composants supplémentaires ayant des capacités pour détecter et analyser le comportement du système et à réagir face à un comportement indésirable. La figure 5.1 montre les trois couches de l'architecture et des mécanismes d'automatisation.

Au niveau application, des composants analysent les besoins de l'application, les décomposent en tâches et les mappent en les présentant aux cœurs IPs correspondants. En outre, les applications doivent s'adapter dynamiquement aux changements de l'environnement, tels que des changements indésirables dans les caractéristiques du système ou dans le comportement des applications, par le déploiement de nouvelles tâches ou la suppression de celles qui existent déjà. Cette adaptation dynamique permet d'assurer un fonctionnement continu des applications même dans des circonstances inconnues.

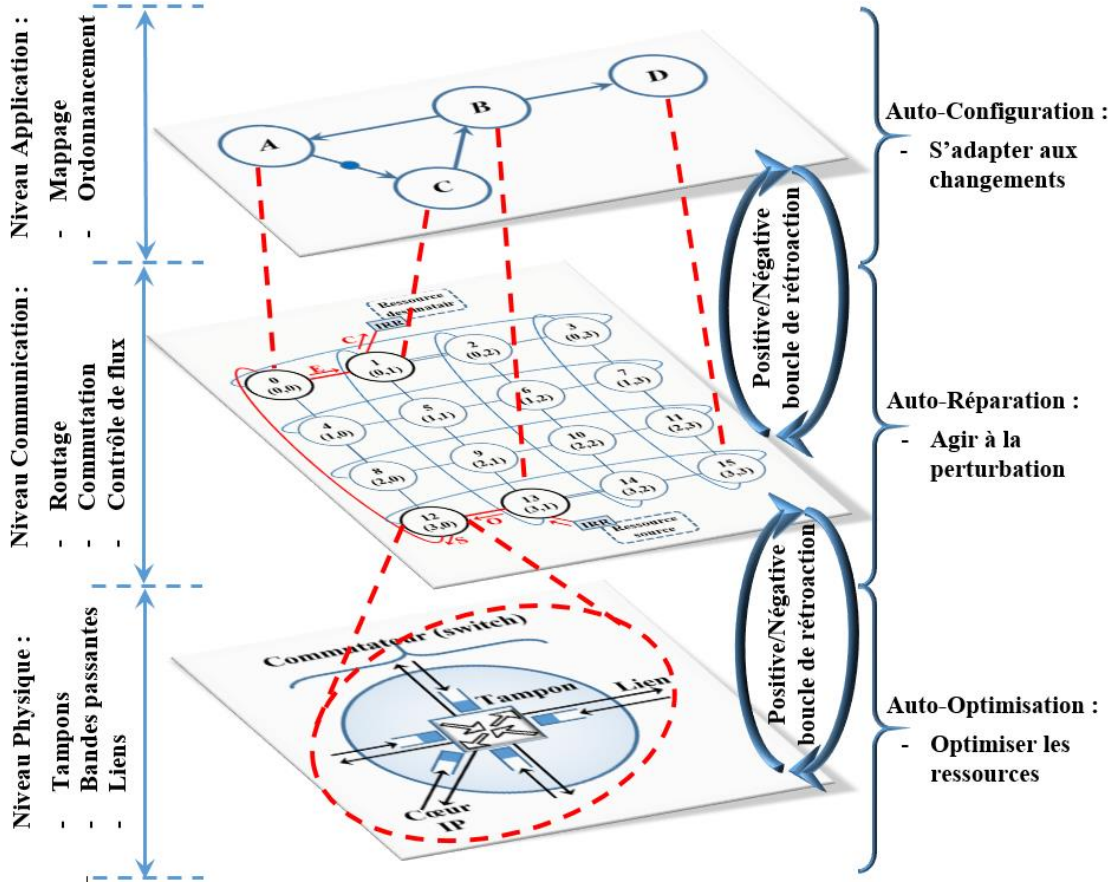


Figure 5.1 : Couches de l'architecture NoC et les mécanismes *Self-CHOP*

Au niveau communication, les capacités d'auto-réparation qui permettent de détecter, de diagnostiquer et de réagir à des perturbations (ex., blocages) sont nécessaires. Cela peut se produire, par exemple, lorsque les flits sont bloqués parce qu'ils attendent les uns les autres pour libérer des tampons ou des canaux. Dans ces situations, les mesures de la congestion doivent être diffusées pour avertir les autres commutateurs pour sélectionner une autre route à cause des points encombrées dans le réseau NoC.

Au niveau infrastructure, différentes techniques d'auto-configuration et de surveillance des ressources (ex., l'allocation de la mémoire tampon et l'adaptation de la bande passante) pourraient être mises en œuvre. Par exemple, le nombre de canaux virtuels actifs peut être ajusté en fonction du trafic prévu afin de diminuer la consommation d'énergie de la mémoire tampon.

IX.2.3. Architectures 3D NoC

Le nombre de composants (ex., processeurs, mémoires, etc.) dans la même puce augmente de façon exponentielle. Lorsque le nombre des composants est important, assurer la connexion entre les différents processeurs dans la même puce constitue un vrai défi en matière de conception et de fabrication. L'utilisation d'un NoC est une solution efficace qui résout les problèmes des moyens classiques de connexion comme le bus, le crossbar et le point à point. Mais, le NoC régulier coûte cher en termes de surface et d'énergie, c'est pourquoi la conception d'une architecture optimale représente un enjeu majeur. En plus, avec la réduction de la taille des transistors, le temps de propagation dans les liens dépasse celui des portes logiques. En effet, il est indispensable de trouver de nouvelles techniques qui permettent de continuer le développement des circuits du semi-conducteur. La conception 3D des circuits intégrés (cf. figure 5.2) est une solution alternative prometteuse qui peut augmenter la densité de transistor pour une application complexe, réduire la longueur des liens, réduire la surface de la puce et qui permet d'utiliser des technologies différentes dans la même architecture, ce qui permet d'augmenter les performances du SoC.

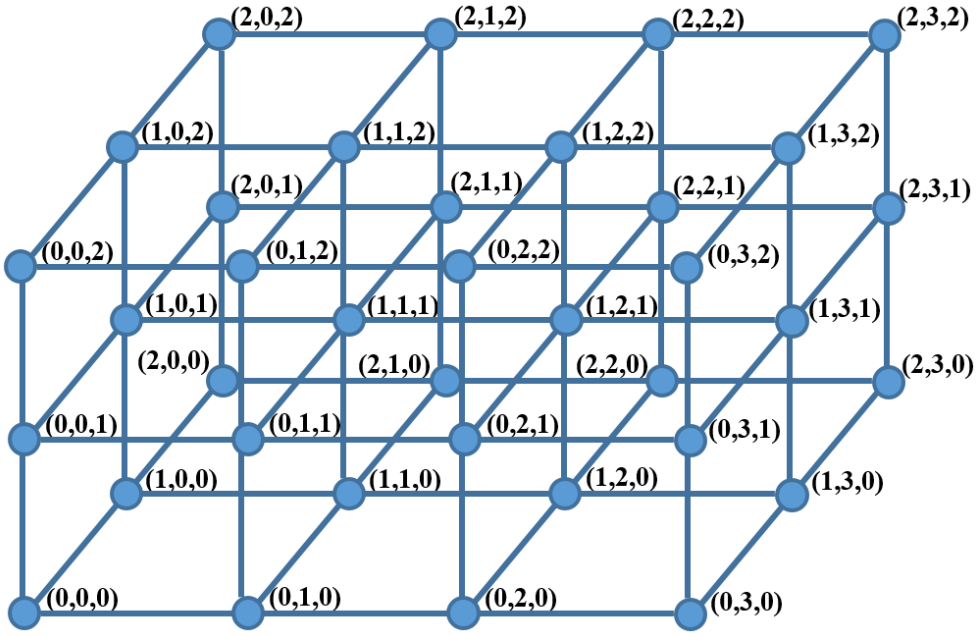


Figure 5.2: Une architecture 3D NoC

Beaucoup de problèmes de conception et d'architecture des 2D NoCs ont été étudiés au cours des dernières années, tels que la conception de flux, l'évaluation de l'implémentation et l'exploration de l'espace de conception [241] [292] [293] [294]. Cependant, l'évaluation architecturale du 3D NoC est limitée, car la technologie est encore en cours de recherche. Par exemple, les auteurs de [295] ont présenté une évaluation des performances d'un 3D NoC pour des données parallèles de l'expérimentation de la conception H.264, en utilisant un simulateur à cycle précis. Les résultats montrent une amélioration d'environ 34% par rapport à l'architecture 2D. Par ailleurs, les auteurs de [296] ont proposé une conception d'un routeur 3D NoC asynchrone basé sur la méthode de la sérialisation. Les résultats d'évaluation montent que le 3D NoC offre une meilleure vitesse de transfert de paquets inter-composants. D'autres travaux d'évaluation des performances de l'architecture 3D ont été réalisés en utilisant la simulation [297] [298] [299] [300] [301]. Plusieurs architectures 3D ont été conçues et fabriqués auparavant. Par exemple, T. Zhang et al. ont développé une architecture 3D SoC pour une application H.264 en utilisant la technologie de Tezzaron [300]. Healy et al. ont fabriqué une architecture multiprocesseur composée de 64 cœurs avec une mémoire d'instructions en une seule couche et un total de 256 Ko de mémoire SRAM dans une autre couche. Une architecture à base de tampons est utilisée pour la communication inter-processeurs [301]. Mais, il existe toujours un besoin de conception et d'implémentation d'une architecture 3D sur FPGA pour mesurer avec précision la performance ainsi que de valider les résultats de simulation.

Vu le manque de conception et d'implémentations des architectures à base de 3D NoC, nous allons étudier de nouvelle méthodologies de conception de ces architectures. Nous considérons que la validation d'un NoC par émulation/simulation nous permet de garantir la bonne fonctionnalité de notre architecture lors de l'implémentation en 3D. Toutefois, plusieurs défis tels que le test des architectures 3D, le placement des connexions verticales, la dissipation de la chaleur et le problème de partitionnement doivent être surmontés avant que la technologie 3D peut être mis en œuvre dans les appareils grand public. Dans ce cadre, nous allons étudier de nouvelles méthodologies pour concevoir une puce multiprocesseur 3D avec une architecture 3D NoC sur *p* niveaux et de mesurer les performances lors de l'exécution des applications, principalement, pour évaluer la communication NoC dans l'architecture 3D.

VII. Liste des publications personnelles

VIII.1. Journaux internationaux avec comité de lecture

[1] **A. Chariete**, M. Bakhouya, J. Gaber, M. Wack, «Towards a Design Space Exploration Methodology for System-On-Chip,» Journal Cybernetics & Information Technologies (CIT) (index SCOPUS). Vol. 14, n° 1, Bulgarian Academy of Sciences, Sofia, 2014.

[2] **A. Chariete**, M. Bakhouya, J. Gaber, M. Wack, E. Coatanea, S. Niar, «A methodology for customizing on-chip interconnect architectures,» Accepted with major change (in revision) in journal Concurrency and Computation: Practice & Experience (index JCR), 2013.

Soumis :

[3] M. Bakhouya, **A. Chariete**, J. Gaber, M. Wack, «A System Dynamics Approach for Congestion Avoidance in Network-on-Chip,» Submitted to the Journal of Systems Architecture (JSA), 2013.

[4] **A. Chariete**, M. Bakhouya, W. Ait-Cheik-Bihi, J. Gaber, R. Kouta, A. Nait-Sidi-Moh, M.Wack, «Data Analysis and User Accepatnce Study of Emergency Calls,» Submitted to the journal IEEE Communications Magazine, 2013.

VIII.2. Conférences internationales avec comité de lecture

[5] **A. Chariete**, M. Bakhouya, J. Gaber, M. Wack, «Towards a Design Space Exploration Methodology for System On-Chip,» Automatics and Informatics scientifics conference, Bulgaria, 2013.

[6] **A. Chariete**, O. Baala, A. Caminada, «The impact of ground-surfaces on the mobile traffic for LTE cellular networks deployment,» International conference Wireless Days (WD), 2013 IFIP, Valencia -Spain, p. 1 – 3, 2013.

[7] **A. Chariete**, M. Bakhouya, J. Gaber, M. Wack, «FracNoC: a Fractal On-Chip Interconnect Architecture For System-on-Chip,» High Performance Computing and Simulation (HPCS), 2013 International Conference on, Helsinki – Finland, p. 213 – 216, 2013.

[8] **A. Chariete**, M. Bakhouya, J. Gaber, M. Wack, «Towards a Design Space Exploration Methodology for Application-Specific NoC,» High Performance Computing and Simulation (HPCS), 2013 International Conference on, Helsinki – Finland, p. 224 – 228, 2013.

[9] **A. Chariete**, M. Bakhouya, J. Gaber, M. Wack, «An approach for customizing on-chip interconnect architectures in SoC design,» High Performance Computing and Simulation (HPCS), 2013 International Conference on, Madrid – Spain, p. 288 – 294, 2012.

[10] M. Bakhouya, **A. Chariete**, J. Gaber, M. Wack, S. Niar, E. Coatanea, «Performance evaluation of a flow control algorithm for network-on-chip,» High Performance Computing and Simulation (HPCS), 2013 International Conference on, Madrid – Spain, p. 281 – 287, 2012.

[11] M. Bakhouya, **A. Chariete**, J. Gaber, M. Wack, «A buffer-space allocation approach for application-specific network-on-chip,» 9th ACS/IEEE International Conference on Computer Systems and Applications (AICCSA), Sharm El Shikh - Egypt, p. 263–267, 2011.

[12] W. Ait-Cheik-Bihi, **A. Chariete**, M. Bakhouya, A. Nait-Sidi-Moh, J. Gaber, M. Wack, «An invehicle eCall platform for efficient road safety,» 8th ITS Europeen Congress (ITS), Lyon, 2011.

VIII.3. Communications

[13] **A. Chariete**, M. Bakhouya, J. Gaber, M. Wack, «Towards an adaptive approach for congestion avoidance in network-on-chip,» 20th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP), Rosenheim - Germany, 2012.

[14] **A. Chariete**, O. Baala, A. Caminada, «L'analyse du handover dans les réseaux LTE,» Congrès de la Société Francophone de Classification (SFC), Marseille, 2012.

[15] **A. Chariete**, O. Baala, A. Caminada, «Analyse des taux de coupures d'appels sur handover pour l'optimisation des réseaux LTE,» Les Journées des Ingénieurs/Doctorants (IngéDoc'12), Belfort, 2012.

[16] **A. Chariete**, M. Bakhouya, J. Gaber, M. Wack, «Une approche de personnalisation des réseaux NOC. Journées Doctorales en Informatique et Réseaux (JDIR),» Belfort, 2011.

[17] **A. Chariete**, O. Baala, A. Caminada, «L'analyse de données pour le diagnostic de dysfonctionnement dans les réseaux LTE,» Les Journées des Ingénieurs-Doctorants (IngéDoc'11), Belfort, 2011.

VIII.4. Rapports scientifiques

[18] **A. Chariete**, O. Baala, A. Caminada, N. Tabia, «Méthode d'identification des stations/créneaux pour l'optimisation automatique,» Livrable L02-1.2 du projet OOL (final). UTBM – Orange Labs, Belfort, 04/2013.

[19] N. Tabia, O. Baala, A. Caminada, **A. Chariete,** «Analyse de Robustesse. Livrable L01-1.2 du projet OOL (final), UTBM – Orange Labs, Belfort, 04/2013.

[20] **A. Chariete**, W. Ait-Cheik-Bihi, M. Bakhouya, J. Gaber, M. Wack, «Report on eCall Large Scale FOT – Evaluation and Assessment – Safety Impact Assessment – Data Analysis and Implications for Safety,» Livrable final du projet Européen TeleFOT, Belfort, 01/2013.

[21] M. Bakhouya, **A. Chariete**, W. Ait-Cheik-Bihi, J. Gaber, M. Wack, «Large scale FOT execution – Field Operational Tests – Detailed FOT Execution – Southern, central and northern test community,» Livrable D3.5.1 du projet Européen TeleFOT, Belfort, 01/2012.

[22] **A. Chariete**, O. Baala, A. Caminada, «Analyse de données GSM et interprétation des résultats,» Livrable L02_1.1 du projet OOL, UTBM – Orange Labs, Belfort, 11/2010.

[23] **A. Chariete**, O. Baala, A. Caminada, «Méthodes d'analyse statistique multi-variables,» Livrable L02_1.0 du projet OOL, UTBM–Orange Labs, Belfort, 07/2010.

VIII. Liste des tableaux

IX. Table des figures

X. Bibliographie

[1] F. Balestra et S. Critoloveanu, «Technologie silicium sur isolant (SOI),» *Techniques de l'ingénieur Technologies des dispositifs actifs*, vol. Référence 42286210, 2013.

[2] W. a. T. B. Dally, «Route Packets, Not Wires: On-chip Interconnection Networks,» chez *Design Automation Conference*, Las Vegas, Nevada, USA, 2001.

[3] P. Pande, C. Grecu, M. Jones, A. Ivanov et R. Saleh, «Performance evaluation and design tradeoffs for networkon-chip interconnect architectures,» *Computers, IEEE Transactions on*, vol. 54, n° 18, p. 1025–1040, 2005.

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

[5] L. Wang, Y. Cao, X. Li et X. Zhu, «Application specific buffer allocation for wormhole routing networks-on-chip,» chez *NoCArc'08, MICRO-41*, 2008.

[6] T. Moscibroda et O. Mutlu, «A case for bufferless routing in on-chip networks,» chez *Proceedings of the 36th International Symposium on Computer Architecture (ISCA)*, 2009.

[7] S. Suboh, M. Bakhouya, J. Gaber et E.-G. El-Ghazawi, «An interconnection architecture for network-on-chip systems,» *Telecommunication Systems*, vol. 37, n° 11-3, p. 137–144, 2008.

[8] M. Stensgaard et J. Sparso, «Renoc: A network-on-chip architecture with reconfigurable topology,» chez *Second ACM/IEEE International Symposium on Networks-on-Chip*, 2008.

[9] M. Modarressi, H. Sarbazi-Azad et M. Arjomand, «A hybrid packet-circuit switched on chip network based on SDM,» chez *Design, Automation & Test in Europe Conference & Exhibition*, 2009.

[10] V. Soteriou et L.-S. Peh, «Exploring the design space of self-regulating power-aware on/off interconnection networks,» *IEEE Transactions on Parallel and Distributed Systems*, vol. 18, n° 13, pp. 393-408, 2007.

[11] M. Faruque, T. Ebi et J. Henkel, «Configurable links for runtime adaptive on chip communication, DATE'09,» chez *Proceedings of the Conference on Design, Automation and Test in Europe*, 2009.

[12] M. Hayenga, N. Jerger et M. Lipasti, «SCARAB: A single cycle adaptive routing and bufferless network,» chez *Proceedings of the 42nd Annual International Symposium on Microarchitecture*, 2009.

[13] S. Suboh, M. Bakhouya, S. Lopez-Buedo et T. El-Ghazawi, «Simulation-based approach for evaluating on-chip interconnect architectures,» chez *Programmable Logic, 2008 4th Southern Conference on*, 2008.

[14] G. Varatkar et R. Marculescu, «Traffic analysis for on-chip networks design of multimedia applications,» chez *Design Automation Conference, 2002. Proceedings. 39th*, 2002.

[15] J. Hu et R. Marculescu, «Application-specific buffer space allocation for networks-on-chip router design,» chez *Proceedings of the 2004 IEEE/ACM International conference on Computer-aided design*, 2004.

[16] U. Ogras et R. Marculescu, «Analytical router modeling for networks-on-chip performance analysis,» chez *Design, Automation & Test in Europe Conference & Exhibition, 2007. DATE '07*, 2007.

[17] M. Gries, «Methods for evaluating and covering the design space during early design development,» *Integration, the VLSI Journal*, vol. 38, n° 12, p. 131 – 183, 2004.

[18] K. Lahiri, A. Raghunathan et S. Dey, «Evaluation of the traffic-performance characteristics of system-on-chip communication architectures,» chez *VLSI Design, 2001. Fourteenth International Conference on*, Bangalore, 2001.

[19] A. Jantsch et H. Tenhunen, Networks on Chip, Kluwer Academic Publishers (ISBN: 1-4020-7392-5), 2003.

[20] D. GaoMing, Z. DuoLi, Y. YongSheng, M. Liang, G. LuoFeng, S. YuKung et G. MingLun, «FPGA prototype design of Network on Chips,» chez *Anti-counterfeiting, Security and Identification, 2nd International Conference on*, 2008.

[21] S. Kumar, A. Jantsch, J. Soininen, M. Forsell, M. Millberg, J. Oberg, K. Tiensyrja et A. Hemani, «A network on chip architecture and design methodology,» chez *VLSI, 2002. Proceedings. IEEE Computer Society Annual Symposium on*, Pittsburgh, 2002.

[22] W. Dally et B. Towles, Principles and Practices of Interconnection Networks, Morgan Kaufmann Publishers an Imprint of Elsevier Inc. ISBN: 0-12-200751-4, 2004, p. 550.

[23] J. Duato , S. Yalamanchili et L. Ni, Interconnection Network: An Engineering Approach, UK: Elsevier Health Sciences, 2003, p. 600.

[24] L. Benini et G. De Micheli, «Networks on Chips: A New SoC Paradigm,» *IEEE Computer Society*, vol. 35, n° 11, pp. 70-78, 2002.

[25] T. Bjerregaard et S. Mahadevan, «A Survey of Research and Practices of Network-on-Chip,» *ACM Computing Surveys (CSUR)*, vol. 38, n° 11, p. 1, 2006.

[26] X. TRAN, «Méthode de Test et Conception en Vue du Test pour les Réseaux sur Puce Asynchrones : Application au Réseau ANOC,» Institut polytechnique de Grenoble, laboratoire CEA-LETI, France, 2008.

[27] S. Evain, «µSpider: Environnement de Conception de Réseau sur Puce,» Institut nationale des sciences appliquées de rennes, 2006.

[28] T. Quarles et W. Christopher, «Spice circuit simulator».

[29] R. Holsmark et M. Högberg, «Modeling and Prototyping of Network on Chip,» Sweden, 2002.

[30] E. Nilsson, «Design and implementation of a hot-potato switch in a network on chip,» Department of Microelectronics and Information Technology, Royal Institute of Technology, IMIT/LECS, Stockholm, Sweden, 2002.

[31] E. Bolotin, I. Cidon, R. Ginosar et A. Kolodny, «QNoC: QoS architecture and design process for network on chip,» *Journal of Systems Architecture*, vol. 50, n° 12-3, p. 105–128, 2004.

[32] G. De Micheli et L. Benini, Networks on Chips: Technology and Tools (Systems on Silicon), 1 éd., L. Benini, Éd., Academic Press, Morgan Kaufmann (Printer), 2006, p. 408.

[33] H. Moussa, «Architectures des réseaux sur puce pour décodeurs canal multiprocesseurs,» l'école nationale supérieure des télécommunications de Bretagne, Bretagne, 2009.

[34] A. Leroy, «Optimizing the on-chip communication architecture of low power systems-on-chip in deep sub-micron technology,» Université Libre de Bruxelles, 2006.

[35] E. Rijpkema, K. Goossen, A. Radulescu, J. Dielissen, P. van Meerbergen, J. Wielage et E. Waterlander, «Trade Offs in the Design of a Router with both Guaranteed and Best-Effort Services for Networks on Chip,» *Computers and Digital Techniques, IEE Proceedings*, vol. 150, n° 15, pp. 294-302, 2003.

[36] A. Mello, L. Tedesco, N. Calazans et F. Moraes, «Virtual Channels in Networks on Chip: Implementation and Evaluation on Hermes NoC,» chez *in Proc. the Symposium on Integrated Circuits and Systems Design*, 2005.

[37] L. Bononi et N. Concer, «Simulation and analysis of network on ship architectures: Ring, Spidergon, and 2D Mesh,» chez *Proc. Automation and Test in Europe*, 2006.

[38] M. Farahabady et H. Sarbazi-Azad, «The WK-Recursive Pyramid: An Efficient Network Topology,» chez *Proceedings of the 8th IEEE International Symposium on Parallel Architectures, Algorithms and Networks*, ISPAN, 2005.

[39] S. Murali, D. Atienza, P. Meloni, S. Carta, L. Benini, G. De Micheli et L. Raffo, «Synthesis of Predictable Networks-on-Chip-Based Interconnect Architectures for Chip Multiprocessors,» chez *Very Large Scale Integration (VLSI) Systems*, 2007.

[40] IBM, Cell broadband engine programming handbook, 2008.

[41] C. Neeb et N. Wehn, «Designing efficient irregular networks for heterogeneous systems-on-chip,» *Journal of Systems Architecture*, p. 54:384–396, 2008.

[42] M. Coppola, R. Locatelli, G. Maruccia, L. Pieralisi et A. Scandurra, «Spidergon: a novel on-chip communication network,» chez *System-on-Chip, 2004. Proceedings. 2004 International Symposium on*, 2004.

[43] P. Guerrier et A. Greiner, «A generic architecture for on-chip packet-switched interconnections,» chez *ACM Proceedings of the conference on Design, automation and test in Europe*, 2000.

[44] P. Doukhan , G. Oppenheim et M. Taqqu, Theory and applications of long-range dependence, B. B. Inc, Éd., Switzerland: Springer, 2003, p. 736.

[45] K. Park et W. Willinger, Self-similar network traffic and performance evaluation, Wiley-Blackwell, Éd., John Wiley and Sons (ISBN: 0471319740), 2000, p. 576.

[46] W. Leland, M. Taqqu, W. Willinger et D. Wilson, «On the self-similar nature of Ethernet traffic (extended edition),» *IEEE/ACM Transactions on Networking*, vol. 2, n° 11, pp. 1-15, 1994.

[47] V. Paxson et S. Floyd, «Wide-area traffic: the failure of Poisson modeling,» *IEEE/ACM Transactions on Networking*, vol. 3, n° 13, pp. 226-244, 1995.

[48] A. Balachandran, G. Voelker, P. Bahl et P. Rangan, «Characterizing user behavior and network performance in a public wireless LAN,» chez *Proceedings of the ACM*

SIGMETRICS international conference on Measurement and modeling of computer systems, 2002.

[49] I. Bucher et D. Calahan, «Models of access delays in multiprocessor memories,» *IEEE Transactions on Parallel and Distributed Systems*, vol. 3, n° 13, pp. 270-280, 1992.

[50] F. Darema-Rogers, G. Pfister et K. So, «Memory access patterns of parallel scientific programs,» *SIGMETRICS '87 Proceedings of the 1987 ACM SIGMETRICS conference on Measurement and modeling of computer systems*, vol. 15, n° 11, pp. 46-58, 1987.

[51] S. Turner, «Performance analysis of multiprocessor interconnection networks using a burst-traffic model,» University of Illinois at Urbana-Champaign, 1995.

[52] P. Kongetira, K. Aingaran et K. Olukotun, «Niagara: a 32-way multithreaded Sparc processor,» *Micro, IEEE*, vol. 25, n° 12, pp. 21-29, 2005.

[53] IBM, «The Power Architecture,» [En ligne]. Available: http://www.ibm.com.

[54] R. Rajwar, M. Herlihy et K. Lai, «Virtualizing transactional memory,» chez *Proceedings of the 32nd International Symposium on Computer Architecture*, 2004.

[55] K. Sankaralingam, R. Nagarajan, H. L., C. K., J. H., D. Burger, S. Keckler et C. Moore, «Exploiting ILP, TLP, and DLP with the polymorphous TRIPS architecture,» *Micro, IEEE*, vol. 23, n° 16, pp. 46 - 51, 2003.

[56] M. Taylor, J. Psota, A. Saraf et N. Shnidman, «Evaluation of the Raw microprocessor: an exposed-wire-delay architecture for ILP and streams,» chez *Proceedings of the 31st International Symposium on Computer Architecture*, 2004.

[57] A. Jalabert, S. Murali, L. Benini et D. De Micheli, «×pipesCompiler: a tool for instantiating application specific networks on chip,» chez *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition*, 2004.

[58] S. Pestana, E. Rijpkema, A. Radulescu, K. Goossens et O. Gangwal, «Cost-performance trade-offs in networks on chip: a simulation-based approach,» chez *Proceedings of the Design, Automation and Test Conference in Europe Conference and Exhibition*, 2004.

[59] D. Wu, B. Al-Hashimi et M. Schmitz, «Improving Routing Efficiency for Network-on-Chip through Contention-Aware Input Selection,» chez *Design Automation, 2006. Asia and South Pacific Conference on*, 2006.

[60] S. Woo, M. Ohara, E. Torrie et J. Singh, «The Splash-2 Programs: Characterization and Methodological Considerations,» chez *Proceedings of International Symposium on Computer Architecture*, 1995.

[61] C. Lee, M. Potkonjak et W. Mangione-Smith, «Mediabench: a tool for evaluating and synthesizing multimedia and communications systems,» chez *Microarchitecture, 1997. Proceedings., Thirtieth Annual IEEE/ACM International Symposium on*, 1997.

[62] SPEC, «The Standard Performance Evaluation Corporation,» 2014. [En ligne]. Available: http://www.spec.org/.

[63] G.-M. Chiu, «The Odd-Even Turn Model for Adaptive Routing,» *IEEE Transactions on Parallel and Distributed Systems*, vol. 11, n° 17, pp. 729 - 738, 2000.

[64] A. Patooghy et H. Sarbazi-Azad, «Performance comparison of partially adaptive routing algorithms,» chez *Advanced Information Networking and Applications, 2006. AINA 2006. 20th International Conference on*, 2006.

[65] M. Palesi, R. Holsmark, S. Kumar et V. Catania, «Application Specific Routing Algorithms for Networks on Chip,» *IEEE Transactions on Parallel and Distributed Systems*, vol. 20, n° 13, pp. 316 - 330, 2009.

[66] R. Holsmark et M. Högberg, «NoC SDL Simulator: Description and User Interface Version 2009-05-20 0.11,» Jönköping University, Sweden, 2009.

[67] G. Varatkar et R. Marculescu, «On-chip traffic modeling and synthesis for MPEG-2 video applications,» *IEEE Transactions of Very Large Scale Integration (VLSI) Systems*, vol. 12, n° 11, pp. 108- 119, 2004.

[68] M. Fulgham et L. Snyder, «Performance of Chaos and Oblivious Routers under Non-Uniform Traffic,» Univ. of Washington.

[69] K. Shin et S. Daniel, «Analysis and Implementation of Hybrid Switching,» chez *Computer Architecture. Proceedings, 22nd Annual International Symposium on*, 1995.

[70] L. Ni et P. McKinley, «A Survey of Wormhole Routing Techniques in Direct Networks,» *IEEE Journal of Computer Science*, vol. 26, n° 12, pp. 62 - 76, 1993.

[71] A. Pullini, F. Angiolini, D. Bertozzi et L. Benini, «Fault tolerance overhead in network-on-chip flow control schemes,» chez *In Proceedings of SBCCI*, 2005.

[72] A. Mostéfaoui, «Architectures Flexibles pour la Validation et l'Exploration de Réseaux Sur Puce,» Institut National Polytechnique De Grenoble, 2009.

[73] J. Flich, P. lopez, M. Malumbers et J. Duato, «Improving the Performance of Regular Networks with Source Routing,» chez *Proceeding of the IEEE International Conference on Parallel Processing*, 2000.

[74] M. Palesi, R. Holsmark, S. Kumar et V. Catania, «A Methodology for Design of Application Specific Deadlock-free Routing Algorithms for NoC Systems. In International Conference on Hardware-Software Codesign and System Synthesis,» chez *International Conference on Hardware-Software Codesign and System Synthesis*, Seoul, Korea, 2006.

[75] Y. Aydogan, C. Stunkel, C. Aykanat et B. Abali, «Adaptive source routing in multistage interconnection networks,» chez *Proceeding of IPPS'96, the 10th International Parallel Processing symposium*, 1996.

[76] J. Lavina, «A Simulator for NoC Interconnect Routing and Application Modeling,» University of Southampton UK & Malaviya National Institute of Technology, UK & India, 2007.

[77] R. Lemaire, «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, 2006.

[78] J. Quartana, «Conception de réseaux de communication sur puce asynchrones : application aux architectures GALS,» Centre pour la Communication Scientifique Directe, 2006.

[79] A. Suboh, «Towards an Adaptive Interconnect for System-on-Chip,» George Washington University, USA, 2010.

[80] L. Benini et G. De Micheli, «Networks on Chips: A New SoC Paradigm,» *IEEE Computer Society Press Los Alamitos, CA, USA*, vol. 35, pp. Issue 1, pp. 70-78, 2002.

[81] M. Bakhouya, «A Bio-Inspired Architecture for Autonomic Network-on-Chip,» *Autonomic Networking-on-Chip: Bio-Inspired Specification, Development, and Verification, Part of the Embedded Multicore Systems (EMS) Book Series,* pp. 1-19, 2012.

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

[83] J. Hu et R. Marculescu, «Energy-Aware Mapping for Tile-based NoC Architectures Under Performance Constraints,» chez *Design Automation Conference, 2003. Proceedings of the ASP-DAC 2003. Asia and South Pacific*, 2003.

[84] U. Ogras et R. Marculescu, «Energy- and performance-driven NoC communication architecture synthesis using a decomposition approach,» chez *Design, Automation and Test in Europe, 2005. Proceedings*, 2005.

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

[86] M. Bakhouya, A. Chariete, J. Gaber et M. Wack, «A Buffer-space Allocation Approach for Application-specific Network-on-Chip,» chez *9th ACS/IEEE International Conference on Computer Systems and Applications*, 2011.

[87] C. Nicopoulos, D. Park, J. Kim, N. Vijaykrishnan, M. Yousif et C. Das, «Vichar: A dynamic virtual channel regulator for network-on-chip routers,» chez *39th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO'06)*, 2006.

[88] M. Al Faruque, T. Ebi et J. Henkel, «Configurable links for runtime adaptive on chip communication,» chez *DATE'09: Proceedings of the Conference on Design, Automation and Test in Europe*, 2009.

[89] E. Carvalho, N. Calazans. et F. Moraes, «Heuristics for dynamic task mapping in noc-based heterogeneous mpsocs,» chez *Proceedings of the 18th IEEE International Workshop on Rapid System Prototyping*, 2007.

[90] S. Murali et G. De Micheli, «An application-specific design methodology for stbus crossbar generation,» chez *In Proceedings of Design*, 2005.

[91] G. Palermo, G. Mariani, C. Silvano, R. Locatelli et M. Coppola, «Mapping and Topology Customization Approaches for Application-Specific STNoC Designs,» chez *Application-specific systems, Architectures and Processors, ASAP. IEEE International Conf. on*, 2007.

[92] C. Bobda, A. Ahmadinia, M. Majer, J. Teich, S. Fekete et J. van der Veen, «DyNoC: A dynamic infrastructure for communication in dynamically reconfugurable devices,» chez *Field Programmable Logic and Applications, 2005. International Conference on*, 2005.

[93] T. Pionteck, R. Koch et C. Albrecht, «Applying partial reconfiguration to networks-on-chips,» chez *Field Programmable Logic and Applications, 2006. FPL '06. International Conference on*, 2006.

[94] M. Hubner, M. Ullmann, L. Braun, A. Klausmann et J. Becker, «Scalable application-dependent network on chip adaptivity for dynamicalrecon_gurable real-time systems,»

chez *International Conference on Field Programmable Logic and Applications (FPL'04)*, 2004.

[95] A. Pinto, L. Carloni et A. Sangiovanni-Vincentelli, «Efficient synthesis of networks on chip,» chez *Proceedings of the 21st International Conference on Computer Design (ICCD)*, 2003.

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

[97] K. Srinivasan, K. Chatha et G. Konjevod, «Linear programming based techniques for synthesis of network on chip architectures,» *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 14, n° 14, p. 407–420, 2006.

[98] J. Xu, W. Wolf, J. Henkel et S. Chakradhar, «A design methodology for application-specific networks-on chip,» *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 5, n° 12, p. 263–280, 2006.

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

[100] U. Ogras et R. Marculescu, «Application-specific network-on-chip architecture customization via long-range link insertion,» chez *Computer-Aided Design, 2005. ICCAD-2005. IEEE/ACM International Conference on*, 2005.

[101] Z. Guz, I. HarWalter, E. Bolotin, I. Cidon, R. Ginosar et A. Kolodny, «Network delays and link capacities in application-specific wormhole NoCs,» *VLSI Design*, vol. 2007, p. 15, 2007.

[102] C. Ciordas, A. Hansson, K. Goossens et T. Basten, «A Monitoring-Aware Network-on-Chip Design Flow,» chez *Digital System Design: Architectures, Methods and Tools, 9th EUROMICRO Conference on*, 2006.

[103] K. Goossens, J. Dielissen et A. Radulescu., «AEthereal network on chip: concepts, architectures, and implementations,» *Design & Test of Computers, IEEE*, vol. 22, n° 15, pp. 414-421, 2005.

[104] J. Wang, Y. Li et H. Li, «An Efficient Link Bandwidth Design Method for Application-specific Network-on-chip,» *IETE Technical Review*, vol. 30, n° 12, p. 102 – 107, 2013.

[105] A. Hansson, M. Wiggers, A. Moonen, K. Goossens et M. Bekooij, «Applying dataflow analysis to dimension buffers for guaranteed performance in networks on chip,» chez *Networks-on-Chip, 2008. NoCS 2008. Second ACM/IEEE International Symposium on*, 2008.

[106] J. Hu, U. Ogras et R. Marculescu, «System-level buffer allocation for application-specific networks-on-chip router design,» *IEEE Tran. On Computer-Aided Design Of Integrated Circuits And Systems*, vol. 25, n° 112, p. 2919 – 2933, 2006.

[107] U. Ogras, J. Hu et R. Marculescu, «Key research problems in NoC design: A holistic perspective,» chez *Hardware/Software Codesign and System Synthesis, 2005. CODES+ISSS '05. Third IEEE/ACM/IFIP International Conference on*, 2005.

[108] P. Bogdan et R. Marculescu, «Quantum-like effects in network-on-chip buffers behavior,» chez *in Proceedings of the 44th Design Automation Conference (DAC)*, 2007.

[109] M. Bakhouya, S. Suboh, J. Gaber et T. El-Ghazawi, «Analytical modeling and evaluation of on-chip interconnects using network calculus,» chez *NoCS Proc*, 2009.

[110] J. Boudec et P. Thiran, Network calculus: A theory of deterministic queuing systems for the internet, Springer Verlag - LNCS 2050, 2012.

[111] R. Cruz, «A Calculus for network delay, Part II: Network analysis,» *IEEE TRANSACTIONS ON INFORMATION TtIEORY*, vol. 37, n° 11, p. 132–141, 1991.

[112] V. Sehgal et D. Chauhan, «State observer controller design for packets flow control in networks-on-chip,» *Journal of Supercomputing*, vol. 54, n° 13, pp. 298-329, 2010.

[113] D. Becker, N. Jiang, G. Michelogiannakis et W. Dally, «Adaptive Backpressure: Efficient Buffer Management for On-Chip Networks,» chez *Proceedings of the 30th IEEE International Conference on Computer*, 2012.

[114] J. Hu, «Design Methodologies for Application Specific Networks-on-Chip,» Carnegie Mellon University, 2005.

[115] T. Schonwald, J. Zimmermann, O. Bringmann et W. Rosenstiel, «Fully adaptive fault-tolerant routing algorithm for network-on-chip architectures,» chez *Proc. Euromicro Conf. Digit. Syst. Des. Archit. Methods Tools*, 2007.

[116] G. Ascia, V. Catania, M. Palesi et D. Patti, «Implementation and analysis of a new selection strategy for adaptive routing in networks-on-chip,» *IEEE Trans. Comput*, vol. 57, n° 16, p. 809–820, 2008.

[117] V. Rantala, T. Lehtonen, P. Liljeberg et J. Plosila, «Hybrid NoC with traffic monitoring and adaptive routing for future 3D integrated chips,» chez *2nd Workshop on Diagnostic Services in Network--on--Chips. Design Automation Conference 200*, USA, 2008.

[118] S. Bourduas et Z. Zilic, «Latency reduction of global traffic in wormhole-routed meshes using hierarchical rings for global routing,» chez *Proc. IEEE International Conference Appllication-Specific Systems, Architecture Process*, 2007.

[119] J. Hollis et C. Jackson, «When does network-on-chip bypassing make sense?,» chez *The 22nd IEEE SoCC Conference*.

[120] Q. Li et W. Jone, «Dyxy-a proximity congestion-aware deadlock-free dynamic routing method for network on chip,» chez *In proc. DAC'06*, 2006.

[121] J. Hu and et R. Marculescu, «Exploiting the routing flexibility for energy/performance aware mapping of regular NoC architectures,» chez *Design, Automation and Test in Europe Conference and Exhibition, 2003*, 2003.

[122] M. Terrence, P. Cheung, K. Lam et L. Wayne, «Adaptive Routing in Network-on-Chips Using a Dynamic-Programming Network,» *IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS*, vol. 58, n° 18, p. 3701 – 3716, 2011.

[123] D. Wiklund et D. Liu, «SoCBus: Switched network on chip for hard real time systems,» chez *Proceedings of the International Parallel and Distributed Processing Symposium (IPDPS)*, 2003.

[124] E. Jerger, P. Li-Shiuan et M. Lipasti, «Circuit-Switched Coherence,» chez *Networks-on-Chip, 2008. NoCS 2008. Second ACM/IEEE International Symposium on*, Newcastle, 2008.

[125] P. Wolkotte, G. Smit, G. Rauwerda et L. Smit, «An Energy-Efficient Reconfigurable Circuit-Switched Network-on-Chip,» chez *Parallel and Distributed Processing Symposium. Proceedings. 19th IEEE International*, 2005.

[126] B. Ahmad, A. T. Erdogan et S. Khawam, «Architecture of a dynamically reconfigurable noc for adaptive recon_gurable mpsoc,» chez *Proceedings of First NASA/ESA Conference on Adaptive Hardware and Systems (AHS'06)*, 2006.

[127] A. Kodi, A. Sarathy et A. Louri, «Adaptive channel buffers in on-chip interconnection networks a power and performance analysis,» *Computers, IEEE Transactions on*, vol. 57, n° 19, pp. 1169-1181, 2008.

[128] H. Ying, A. Jaiswal, T. Hollsteiny et K. Hofmann, «A Fast Congestion-Aware Flow Control Mechanism for ID-Based Networks-on-Chip with Best-Effort Communication,» chez *14th Euromicro Conference on Digital System Design*, 2011.

[129] F. Jafari et M. Yaghmaee, «A novel flow control scheme for best effort traffics in Network-on-Chip based on Weighted Max-Min-fairness,» chez *Telecommunications, IST. International Symposium on*, 2008.

[130] C. Chao, M. Jie, A. Coskun et A. Joshi, «Express Virtual Channels with Taps (EVC-T): A Flow Control Technique for Network-on-Chip (NoC) in Manycore Systems,» chez *High Performance Interconnects (HOTI), 2011 IEEE 19th Annual Symposium on*, 2011.

[131] W. Sung-Tze, C.-H. Chao, I.-C. Wey et A.-Y. Wu, «Dynamic Channel Flow Control of Networks-on-Chip Systems for High Buffer Efficiency,» chez *Signal Processing Systems, 2007 IEEE Workshop on*, 2007.

[132] M. AL Faruque, R. Krist et J. Henkel, «Adam: Run-time agent-based distributed application mapping for on-chip communication,» chez *45th IEEE/ACM/EDA Design Automation Conference (DAC'08)*, 2008.

[133] p. Zipf, G. Sassatelli, N. Utlu, N. Saint-Jean, P. Benoit et M. Glesner, «A decentralised task mapping approach for homogeneous multiprocessor network-on-chips,» *International Journal of Reconfigurable Computing - Selected papers from ReCoSoc'08*, p. 14, 2009.

[134] S. Murali et G. De Micheli, «Bandwidth-Constrained Mapping of Cores onto NoC Architectures,» chez *IEEE DATE-04: Design, Automation, and Test in Europe*, 2004.

[135] M. Nickschas et U. Brinkschulte, «Decentralized task allocation in an organic real-time middleware-an action-based approach,» chez *Ubiquitous, Autonomic and Trusted Computing, 2009. UIC-ATC '09. Symposia and Workshops on*, 2009.

[136] S. Murali, M. Coenen, A. Radulescu, K. Goossens et G. De Micheli, «A methodology for mapping multiple use-cases onto networks on chips,» chez *In Proceedings of DATE'06*, 2006.

[137] G. Ascia, V. Catania et M. Palesi, «Multi-Objective Mapping for Mesh-based NoC Architectures,» chez *CODES-ISSS: Second International Conference on Hardware/Software Codesign and System Synthesis*, 2004.

[138] T. Lei et S. Kumar, «A Two-Step Genetic Algorithm for Mapping Task Graphs to a Network on Chip Architecture,» chez *Euromicro Symposium on Digital Systems Design*, Turkey, 2003.

[139] G. Palermo, G. Mariani, C. Silvano, R. Locatelli et M. Coppola, «A Topology Design Customization Approach for STNoC,» chez *Nano-Net '07: Proceedings of the 2nd international conference on Nano-Networks*, 2007.

[140] A. Kiasari, A. Jantsch et Z. Lu, «Mathematical Formalisms for Performance Evaluation of Networks-on-Chip,» *Journal ACM Computing Surveys (CSUR)*, vol. 45, n° 13, pp. Article 38, 41 pages, 2013.

[141] J. Kim, D. Park, C. Nicopoulos, N. Vijaykrishnan et C. R. Das, «Design and analysis of an NoC architecture from performance, reliability and energy perspective,» chez *Architecture for networking and communications systems, 2005. ANCS 2005. Symposium on*, Princeton, 2005.

[142] U. Ogras, P. Bogdan et R. Marculescu, «An Analytical Approach for Network-on-Chip Performance Analysis,» *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 29, n° 112, pp. 2001-2013, 2010.

[143] Cheng, Y. Pan, X. Yan et R. Huan, «A general communication performance evaluation model based on routing path decomposition,» *Journal of Zhejiang University-SCIENCE C (Computers & Electronics)*, vol. 12, n° 17, pp. 561-573, 2011.

[144] E. Krimera, I. Keslassy, A. Kolodny, I. Walter et M. Erez, «Static Timing Analysis for Modeling QoS in Networks on Chip,» *Journal of Parallel and Distributed Computing*, vol. 71, n° 15, p. 687–699, 2011.

[145] J. Wang, Y. Li et Q. Peng, «A Novel Analytical Model for Network-on-Chip using Semi-Markov Process,» *Advances in Electrical and Computer Engineering*, vol. 11, pp. 111 - 118, 2011.

[146] S. Foroutan, Y. Thonnart, R. Hersemeule et A. Jerraya, «An Analytical Method for Evaluating Network-on-Chip Performance,» chez *Proceedings of the Design, Automation and Test in Europe Conference*, 2010.

[147] S. Foroutan, Y. Thonnart, R. Hersemeule et A. Jerraya, «Analytical Computation of Packet Latency in a 2D-mesh NoC,» chez *Proceedings of the Joint IEEE North-East Workshop on Circuits and Systems and TAISA Conference*, 2009.

[148] F. Jafari, Z. Lu, A. Jantsch et M. Yaghmaee, «Buffer Optimization in Network-on-Chip Through Flow Regulation,» *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 29, n° 112, pp. 1973 - 1986, 2010.

[149] L. Kleinrock, Queueing Systems: theory, vol. 1, Wiley-Interscience, 1975, p. 417.

[150] R. Cruz, «A calculus for network delay, Part I: Network elements in isolation,» *IEEE TRANSACTIONS ON INFORMATION TtIEORY*, vol. 37, n° 11, p. 114–131, 1991.

[151] S. Chakraborty, S. Künzli, L. Thiele, A. Herkersdorf et P. Sagmeister, «Performance Evaluation of Network Processor Architectures: Combining Simulation with Analytical Estimation,» *Elsevier Computer Networks*, vol. 41, n° 15, p. 641–665, 2003.

[152] K. Tindell et J. Clark, «Holistic schedulability analysis for distributed hard real-time systems,» *Journal Microprocessing and Microprogramming - Parallel processing in embedded real-time systems* , vol. 40, n° 12-3, p. 117–134, 1994.

[153] S. Martin, P. Minet et L. George, «End-to-end response time with fixed priority scheduling: trajectory approach versus holistic approach,» chez *Int. J. Communication Systems*, 2005.

[154] M. Hendriks et M. Verhoef, «Timed automata based analysis of embedded system architectures,» chez *Parallel and Distributed Processing Symposium, 2006. IPDPS 2006. 20th International*, 2006.

[155] S. Perathoner, E. Wandeler, L. Thiele, A. Hamann, S. Schliecker, R. Henia, R. Racu, R. Ernst et M. Harbour, «Influence of different system abstractions on the performance analysis of distributed real-time systems,» chez *Proceedings of EMSOFT'2007. Proceedings of the 7th ACM & IEEE international conference on Embedded software*, 2007.

[156] H. Zhang, «Service Disciplines For Guaranteed Performance Service in Packet-Switching Networks,» *Proceedings of the IEEE* , vol. 83, n° 110, p. 1374 – 1396, 1995.

[157] Y. Qian, Z. Lu et W. Dou, «Worst case Flit and Packet Delay Bounds in Wormhole Networks on Chip,» *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, Special Section on VLSI Design and CAD Algorithms*, Vols. 1 sur 292-A, n° 112, pp. 3211-3220, 2009.

[158] M. Bakhouya, S. Suboh, J. Gaber, T. A. El-Ghazawi et S. Niar, «Performance evaluation and design tradeoffs of on-chip interconnect architectures,» *Simulation Modelling Practice and Theory*, vol. 19, n° 16, pp. 1496-1505, 2011.

[159] Y. Qian, Z. Lu et W. Dou, «Applying Network Calculus for Performance Analysis of Self-Similar Traffic in On-Chip Networks,» chez *Proceedings of the IEEE/ACM/IFIP International Conference on Hardware-Software Codesign and System Synthesis (CODES+ISSS)*, 2009.

[160] Y. Qian, Z. Lu et W. Dou, «Analysis of Worst case Delay Bounds for On-Chip Packet-Switching Networks,» chez *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, 2010.

[161] Y. Qian, Z. Lu et Q. Dou, «QoS Scheduling for NoCs: Strict Priority Queueing versus Weighted Round Robin,» chez *Proceedings of the 28th International Conference on Computer Design*, 2010.

[162] L. Jouhet, «Algorithmique du Network Calculus,» École Normale Supérieure de Lyon - Université de Lyon, 2012.

[163] J. Lehoczky, L. Sha et Y. Ding, «The rate monotonic scheduling algorithm: Exact characterization and average case behaviour,» chez *In the Proceedings of the IEEE Real-Time Systems Symposium*, 1989.

[164] J. Leung et J. Whitehead, «On the complexity of fixed priority scheduling of periodic, real-time tasks,» *Elsevier: Performance Evaluation*, vol. 2, n° 14, pp. 237-250, 1982.

[165] C. Liu et J. Layland, «Scheduling algorithms for multiprogramming in a hard-real-time environment,» *Journal of the ACM*, vol. 20, pp. 47-61, 1973.

[166] J. Li et M. Mutka, «Priority based real-time communication for large scale wormhole networks,» chez *In Proceedings of the 8th International Symposium on Parallel Processing*, 1994.

[167] Z. Shi et A. Burns, «Real-Time Communication Analysis for On-Chip Networks with Wormhole Switching,» chez *Proceedings of the ACM/IEEE International Symposium on Networks-on-Chip (NOCS)*, 2008.

[168] P. Balbastre, I. Ripoll et A. Crespo, «Minimum deadline calculation for periodic real-time tasks in dynamic priority systems,» *IEEE Trans. Computers*, vol. 57, n° 11, p. 96–109, 2008.

[169] K. Jeffay, D. Stanat et C. Martel, «On nonpreemptive scheduling of period and sporadic tasks,» chez *Proceedings of the 12 th IEEE Symposium on Real-Time Systems*, 1991.

[170] J. Wu, J. Liu et W. Zhao, «A General Framework for Parameterized Schedulability Bound Analysis of Real-Time Systems,» *IEEE Transactions on Computers*, vol. 59, n° 16, pp. 776-783, 2010.

[171] N. C. Audsley, A. Burns, K. Tindell et A. Wellings, «Applying New Scheduling Theory to Static Priority Preemptive Scheduling,» *Software Engineering Journal*, vol. 8, n° 15, pp. 284-292, 1993.

[172] M. Sjödin et H. Hansson, «Improved Response-Time Analysis Calculations,» chez *Proceedings of the IEEE Real-Time Systems Symposium*, 1998.

[173] S. Balakrishnan et F. Ozguner, «A priority-driven flow control mechanism for real-time traffic in multiprocessor networks,» chez *IEEE Transaction on Parallel and Distributed Systems*, 1998.

[174] H. Song, B. Kwon et H. Yoon, «Throttle and preempt: A new flow control for real-time communications in wormhole networks,» chez *Proceedings of the international Conference on Parallel Processing*, 1997.

[175] D. Kandlur, K. Shin et D. Ferrari, «Real-time communication in multi-hop networks,» *IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEM*, vol. 5, n° 110, pp. 1044-1046, 1994.

[176] J. Li et M. Mutka, «Real-time virtual channel flow control,» *Journal of Parallel and Distributed Computing*, vol. 32, pp. 49-65, 1996.

[177] S. Hary et F. Ozguner, «Feasibility test for real-time communication using wormhole routing,» *IEEE Proceedings - Computers and Digital Techniques*, vol. 144, n° 15, p. 273–278, 1997.

[178] B. Kim, J. Kim, S. Hong et LeeS., «A real-time communication method for wormhole switching networks,» chez *Proceedings of the International Conference on Parallel Processing*, 1998.

[179] Z. Lu, A. Jantsch et I. Sander, «Feasibility analysis of messages for on-chip networks using wormhole routing,» chez *In the Proceedings of the conference on Asia South Pacific design automation*, 2005.

[180] A. Aho, R. Sethi et J. Ullman, Complers : principles, techniques, and tools, Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc, 2006.

[181] E. Lee et T. Parks, «Dataflow process networks,» *the Proceedings of the IEEE*, vol. 83, n° 15, p. 773–799, 1995.

[182] S. Stuijk, M. Geilen, B. Theelen et T. Basten, «Scenario-aware Dataflow: Modeling, Analysis and Implementation of Dynamic Applications,» chez *Proceedings of the International Conference on Embedded Computer Systems*, 2011.

[183] E. Lee et D. Messerschmitt, «Synchronous data flow,» *Proceedings of the IEEE*, vol. 75, n° 19, pp. 1235-1245, 1987.

[184] E. Lee et D. Messerschmitt, «Static scheduling of synchronous data flow programs for digital signal processing,» *IEEE Transaction on Computers*, Vols. 1 sur 2C-36, n° 11, pp. 24-35, 1987.

[185] M. Bekooij, H. R., O. Moreira, P. Poplavko, M. Pastrnak, B. Mesman, J. Mol, S. Stuijk, G. V. Gheorghita et J. and van Meerbergen, «Dataflow Analysis for Real-Time Embedded Multiprocessor System Design,» chez *Dynamic and Robust Streaming Between Connected Consumer-Electronic Devices*, P. v. d. Stok, Éd., Printed in the Netherlands, Kluwer Academic Publishers, © 2005 Springer, 2005, pp. 81-108.

[186] J. Horstmannshoff, T. Grotker et H. Meyr, «Mapping Multirate dataflow to complex RT level hardware models,» chez *Proceedings of the IEEE International Conference on Application-Specific Systems, Architectures and Processors*, 1997.

[187] J. Dennis, «Data flow supercomputers,» *IEEE Computer*, vol. 13, n° 111, pp. 48-56, 1980.

[188] J. Rumbaugh, «A data flow multiprocessor,» *IEEE Transactions on Computers*, Vols. 1 sur 2C-26, n° 12, pp. 138-141, 1977.

[189] K. Parhi, «Algorithm Transformation Techniques for Concurrent Processors,» *Proceedings of the IEEE*, vol. 77, n° 112, p. 1879–1895, 1989.

[190] G. Bilsen, M. Engels, R. Lauwereins et J. Peperstraete, «Cyclo-Static Dataflow,» chez *IEEE Acoustics, Speech, and Signal Processing, 1995. ICASSP-95., International Conference on*, 1995.

[191] R. Lauwereins, P. Wauters, M. Ade et J. A. Peperstraete, «Geometric parallelism and cyclo-static data flow in GRAPE-II,» chez *International Workshop on Rapid System Prototyping*, 1994.

[192] J. Buck, «A Dynamic Dataflow Model Suitable for Efficient Mixed Hardware and Software Implementations of DSP Applications,» chez *Proceedings of Codes/Cashe 94, Third International Workshop on Hardware/Software Codesign*, Grenoble, France, 1994.

[193] J. T. Buck et E. Lee, «Scheduling dynamic dataflow graphs with bounded memory using the Token Flow Model,» chez *Acoustics, Speech, and Signal Processing, 1993. ICASSP-93., 1993 IEEE International Conference on*, USA, 1993.

[194] B. Bhattacharya et S. Bhattacharyya, «Parameterized Dataflow Modeling for DSP Systems,» chez *IEEE Transactions on Signal Processing, ICASSP '00. Proceedings. IEEE International Conference on*, 2001.

[195] M. Wiggers, M. Bekooij et G. Smit, «Buffer capacity computation for throughput constrained streaming applications with data-dependent inter-task communication,» chez *Proceedings of the Real-Time and Embedded Technology and Applications Symposium*, 2008.

[196] M. Wiggers, M. Bekooij et G. Smit, «Buffer capacity computation for throughput constrained modal task graphs,» *ACM Transactions on Embedded Computing Systems*, vol. 10, n° 12, p. 1–59, 2010.

[197] B. Theelen, M. Geilen, T. Basten, J. Voeten, S. Gheorghita et S. Stuijk, «A Scenario-Aware Data Flow Model for Combined Long-Run Average and Worst-Case Performance Analysis,» chez *International Conference on Formal Methods and Models for Co-Design*, 2006.

[198] S. Sriram et S. Bhattacharyya, Embedded Multiprocessors: Scheduling and Synchronization, New York, USA: Marcel Dekker, Inc., 2000.

[199] A. Dasdan, «Experimental Analysis of the Fastest Optimum Cycle Ratio and Mean Algorithms,» *ACM Transactions on Design Automation of Electronic Systems,* vol. 9, n° 14, pp. 385-418, 2004.

[200] A. Dasdan et R. Gupta, «Faster Maximum and Minimum Mean Cycle Algorithms for System Performance Analysis,» *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,* vol. 17, n° 110, pp. 889-899, 1998.

[201] R. Karp, «A characterization of the minimum cycle mean in a digraph,» *Elsevier: Discrete Mathematics,* vol. 23, n° 13, p. 309–311, 1978.

[202] N. Young, R. Tarjan et J. Orlin, Faster parametric shortest path and minimum-balance algorithms, vol. 21, Cambridge, Mass. : Sloan School of Management, Massachusetts Institute of Technology, 1991, p. 205–221.

[203] A. Ghamarian, S. Stuijk, T. Basten, M. Geilen et B. D. Theelen, «Latency Minimization for Synchronous Data Flow Graphs,» chez *Proceedings of the Euromicro Conference on Digital System Design: Architectures, Methods and Tools*, 2007.

[204] M. Bekooij, O. Moriera, P. Poplavko, B. Mesman, M. Pastrnak et J. van Meerbergen, «Predictable Embedded Multiprocessor System Design,» chez *Software and Compilers for Embedded Systems, the 8th International Workshop, SCOPES 2004*, vol. 3199, Amsterdam, The Netherlands, Springer Berlin Heidelberg, 2004, pp. 77-91.

[205] M. Wiggers, M. Bekooij et G. Smit, «Modelling Run-time Arbitration by Latency-rate Servers in Dataflow Graphs,» chez *Proceedings of the International Workshop on Software and Compilers for Embedded Systems*, 2007.

[206] D. Stiliadis et A. Varma, «Latency-rate servers: A general model for analysis of traffic scheduling algorithms,» *IEEE/ACM Transactions on Netwroking,* vol. 6, n° 15, p. 611–624, 1998.

[207] A. Hansson et K. Goossens, On-Chip Interconnect with aelite: Composable and Predicable Systems, Embedded Systems Series, ISBN 978-1-4419-6496-0, Springer, 2010.

[208] A. Hansson, M. Wiggers, A. Moonen, K. Goossens et M. Bekooij, Enabling Application-Level Performance Guarantees in Network-Based Systems on Chip by Applying Dataflow Analysis, vol. 3, Enabling Application-Level Performance Guarantees in Network-Based Systems on Chip by Applying Dataflow Analysis, 2009, pp. 398 - 412.

[209] M. Wiggers, M. Bekooij et G. Smit, «Efficient computation of buffer capacities for cyclo-static dataflow graphs,» chez *Proceedings of the Design Automation Conference*, 2007.

[210] P. Bergé, Y. Pomeau et C. Vidal, «L'ordre dans le chaos. Vers une approche déterministe de la,» Paris, 1984.

[211] J. Forrester, «World Dynamics,» *MIT Press,* p. 13, 1971.

[212] J. Forrester, «Industrial Dynamics - adding structure and relevance to pre-colleg education,» chez *Manning KH (ed) Shaping the future. MIT Press*, Cambridge, MA, 1990.

[213] M. Goodman, Study Notes in System Dynamics, Cambridge, MA: Pegasus Communications, 1983, p. 388.

[214] D. Waldman, «The contributions of total quality management to a theory of work performance,» *Academy of Management Review*, vol. 19, n° 13, pp. 510-546, 1994.

[215] A. Bauer, G. Reiner et R. Schomschule, «Organizational and quality systems development: an analysis via a dynamic simulation model,» *Journal of Total Quality Management*, vol. 11, n° 14, pp. 410-416, 2001.

[216] D. Leonard, R. McAdam et et. al., «A grounded multi-model framework for TQM dynamics,» *International Journal of Quality and Reliability Management*, vol. 19, n° 16, pp. 710-736, 2002.

[217] G. Richardson, L. Alexander et I. Pugh, Introduction to System Dynamics Modeling with Dynamics Modeling with Dynamo, Cambridge: Pegasus Communications, 1981, p. 424.

[218] W. Haddad, V. Chellaboina et Q. Hui, Nonnegative and compartmental, Princeton University Press, ISBN 978-0-691-14411-5, 2010, p. 624.

[219] G. Bastin et V. Guffens, «Congestion control in compartmental network systems,» *Elsevier : Systems & Control Letters*, vol. 55, n° 18, p. 689–696, 2006.

[220] T. Hayakawa, «Compartmental Modeling and Neural Network Control of Stochastic Nonnegative Systems,» chez *American Control Conference, 2007. ACC '07*, New York, 2007.

[221] Y. Hayakawa, S. Hosoe, M. Hayashi et M. Ito, «On the structural controllability of compartmental systems,» *Automatic Control, IEEE Transactions on*, vol. 29, n° 11, pp. 17 - 24, 1984.

[222] M. Higashi, «Residence time in constant compartmental ecosystems,» *Ecological Modelling*, vol. 32, n° 14, pp. 243-250, 1986.

[223] M. Gorunescu et F. Gorunescu, «Modeling the kinetics behind the patients flow,» *Sib. Zh. Vychisl. Mat.*, vol. 10, n° 13, p. 229–235, 2007.

[224] G. Bastin, «Sur la modélisation et le contrôle des réseaux dynamiques conservatifs,» *Revue E-STA, Special CIFA*, vol. 3, n° 12, 2007.

[225] S. Rosenblum, A. Herrod, E. Witchel et A. Gupta, «Complete computer system simulation: The simos approach,» *IEEE Parallel and DistributedTechnology: Systems & Technology*, vol. 3, n° 14, pp. 34-43, 1995.

[226] V. Puente, J. Gregorio et R. Beivide, «Sicosys: an integrated framework for studying interconnection network performance in multiprocessor systems,» chez *Parallel, Distributed and Network-based Processing, 2002. Proceedings. 10th Euromicro Workshop on*, 2002.

[227] V. Puente, J. Gregorio et R. Beivide, «SICOSYS».

[228] P. Magnusson, M. Christensson, J. Eskilson, D. Forsgren, G. Hallberg, J. Hogberg, F. Larsson, A. Moestedt et B. Werner, «Simics : A full system simulation platform, Computer,» 2002.

[229] V. Pai, P. Ranganathan et S. Adve, «RSIM: An Execution-Driven Simulator for ILP-Based Shared-Memory Multiprocessors and Uniprocessors,» chez *Proceedings of the Third Workshop on Computer Architecture Education*, 1997.

[230] D. Perez, G. Mouchard et O. Temam, «Microlib: A case for the quantitative comparison of micro-architecture mechanisms, in Microarchitecture,» chez *MICRO 37 Proceedings of the 37th annual IEEE/ACM International Symposium on Microarchitecture*, 2004.

[231] J. Renau, B. Fraguela, W. Tuck, M. Prvulovic et L. Ceze, «SESC simulator,» 2005.

[232] D. August, J. Chang, S. Girbal, D. Gracia-Perez, G. Mouchard, D. Penry, O. Temam et N. Vachharajani, «UNISIM: An Open Simulation Environment and Library for Complex Architecture Design and Collaborative Development,» Computer Architecture Letters, 2007.

[233] V. Reyes, P. Bautista, G. Marrero, P. Carballo et W. Kruijtzer, «CASSE: a system-level modeling and design-space exploration tool for multiprocessor systems-on-chip,» chez *Digital System Design, 2004. DSD 2004. Euromicro Symposium on*, 2004.

[234] W. Cesario, D. Lyonnard, G. Nicolescu, Y. Paviot, S. Yoo, A. Jerraya, L. Gauthier. et M. Diaz-Nava, «Multiprocessor soc platforms: a component-based design approach,» *Design & Test of Computers, IEEE*, vol. 19, n° 16, pp. 52-63, 2002.

[235] K. Huang, S.-i. Han, K. Popovici, L. Brisolara, X. Guerin, L. Li, X. Yan, S.-l. Chae, L. Carro et A. A. Jerraya, «Simulink-based mpsoc design flow: case study of motion-jpeg and h.264, in DAC '07,» chez *Proceedings of the 44th annual Design Automation Conference*, 2007.

[236] M. Martin, D. Sorin, B. Beckmann, M. Marty, M. Xu, A. Alameldeen, K. Moore, M. Hill et D. Wood, «Multifacet's general execution-driven multiprocessor simulator (GEMS) toolset, SIGARCH Comput,» 2005.

[237] S. Swan, «An Introduction to System Level,» Cadence Design Systems, Inc., 2001.

[238] A. S. Initiative, «Transaction-level Modeling Working Group,,» [En ligne]. Available: http://www.systemc.org/.

[239] G. Beltrame, C. Bolchini, L. Fossati, A. Miele et D. Sciuto, «ReSP: A non-intrusive Transaction-Level Reflective MPSoC Simulation Platform for design space exploration,» chez *Design Automation Conference, 2008. ASPDAC 2008. Asia and South Pacific*, Seoul, 2008.

[240] GAPH-group, «Atlas webpage,» [En ligne]. Available: http://www.inf.pucrs.br/~gaph/AtlasHtml/AtlasIndex_us.html.

[241] S. Evain, J. Diguet et D. Houzet, «ySpider: a CAD Tool for Efficient NoC Design,» Université de Bretagne Sud, LORIENT, 2004.

[242] J. Chan et S. Parameswaran, «NoCGEN: A Template Based Reuse Methodology for Networks on Chip Architecture,» chez *VLSI Design, 2004. Proceedings, 17th International Conference on*, 2004.

[243] M. Palesi, D. Patti et F. Fazzino, «Noxim».

[244] Z. Lu, R. Thid, M. Millberg, E. Nilsson et A. Jantschi, «NNSE: Nostrum Network-on-Chip Simulation Environment,» Royal Institute of Technology, Sweden.

[245] Z. Lu, R. Thid, E. Nilsson et A. Jantsch, «nostrum-NNSE,» [En ligne]. Available: http://www.ict.kth.se/nostrum/NNSE/.

[246] H. Hossain, M. Ahmed, A. Al-Nayeem, T. Islam et M. Akbar, «Gpnocsim: a general purpose simulator for network-on-chip, in Information and Communication Technology,» chez *ICICT '07. International Conference on*, 2007.

[247] L. Niket Agarwal, T. Krishna et N. Jha, «Garnet: A detailed on-chip network model inside a full-system simulator,» chez *IEEE International Symposium on Performance Analysis of Systems and Software*, 2009.

[248] Z. Lu et A. Jantsch, «Traffic configuration for evaluating networks on chips, in System-on-Chip for Real-Time Applications,» chez *Proceedings.5th International Workshop on*, 2005.

[249] N. Jiang, G. Michelogiannakis, D. Becker, B. Towles et W. Dally, «BookSim 2.0,» 2013.

[250] J. Hu et R. Marculescu, «DyAD: Smart Routing for Networks-on-Chip,» chez *Design Automation Conference, 2004 Proceedings.*, 2004.

[251] Y. Ben-Itzhak, E. Zahavi, I. Cidon et A. Kolodny, «HNOCS: Modular Open-Source Simulator for Heterogeneous NoCs,» chez *Embedded Computer Systems (SAMOS), International Conference on*, 2012.

[252] W. Richardson, M. Bailey et W. Sanders, «Using zpl to develop a parallel chaos router simulator,» chez *Simulation Conference, 1996. Proceedings. Winter*, 1996.

[253] E. Altman et T. Jiménez, «NS Simulator for beginners,» Université de Los Andes, Mérida, Venezuela and ESSI, Sophia-Antipolis, France, 2003-2004.

[254] K. Fall et K. Varadhan, «The NS Manual,» The VINT Project, a Collaboration between researchers at UC Berkeley, LBL, USC/ISI, and Xerox PARC, 2011.

[255] A. r. r. © Mentor Graphics, «Mentor Graphics,» [En ligne]. Available: http://www.mentor.com/.

[256] «Coware,» [En ligne]. Available: http://www.coware.com/.

[257] J. Emer, P. Ahuja, E. Borch, A. Klauser, C.-K. Luk, S. Manne, S. Mukherjee, S. Mukherjee, H. Patil, S. Wallace, N. Binkert, R. Espasa et T. Juan, «Asim: A Performance Model Framework,» *IEEE Computer,* vol. 35, n° 12, pp. 68-76, 2002.

[258] J. Cong, K. Gururaj, G. Gururaj, A. Kaplan et M. Naik, «MC-Sim: An e-cient simulation tool for MPSoC designs,» chez *Computer-Aided Design, 2008. ICCAD 2008. IEEE/ACM International Conference on*, 2008.

[259] L. Benini, D. Bertozzi, A. Bogliolo, F. Menichelli et M. Olivieri, «MPARM: Exploring the Multi-Processor SoC Design Space with SystemC,» *Journal of VLSI signal processing systems for signal, image and video technology,* vol. 41, n° 12, pp. 169-182, 2005.

[260] «SoClib,» [En ligne]. Available: https://www.soclib.fr/.

[261] J. Kim, J. Balfour et J. Balfour, «Flattened Butterfly Topology for On-Chip Networks,» *IEEE Computer Architecture Letters,* vol. 6, n° 12, p. 37 – 40, 2007.

[262] J. Hu et R. Marculescu, «Energy- and Performance-Aware Mapping for Regular NoC Architectures,» *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on* , vol. 24, n° 14, pp. 551 - 562, 2005.

[263] W. Hang-Sheng, Z. Xinping, P. Li-Shiuan et S. Malik, «Orion: a power-performance simulator for interconnection networks,» chez *Microarchitecture, 2002. (MICRO-35). Proceedings. 35th Annual IEEE/ACM International Symposium on*, 2002.

[264] A. Varga, «Omnet++ simulator».

[265] S. Evain, J. Diguet, R. Dafali, Y. Eustache et E. Juin, «uSpider CAD tool: Case study of NoC IP generation for FPGA,» 2008.

[266] I. A. R. R. © 2014 Arteris, «ARTERIS - The Network-on-Chip Company,» [En ligne]. Available: http://www.arteris.com/.

[267] X. Li et Hammami., «NOCDEX: Network on Chip Design Space Exploration through Direct Execution and Options Selection through Principal Component Analysis,» chez *Industrial Embedded Systems, International Symposium on*, 2006.

[268] P. Guerrier, «Un réseau d'interconnexion pour systèmes intégrés,» Université Pierre et Marie Curie LIP6, 2000.

[269] R. Thid, «A Network on Chip Simulator,» Royal Institute of Technology (KTH), Stockholm, 2002.

[270] F. Moraes, N. Calazans, A. Mello, L. Moller et L. Ost, «HERMES: an Infrastructure for Low Area Overhead Packet-Switching Networks on Chip,» *Integration, the VLSI Journal*, pp. 38(1): 69-93, 2004.

[271] V. Guffens, G. Bastin et H. Mounier, «Fluid flow network modeling for hop-by-hop feedback control design and analysis,» chez *Proceedings Internetworking*, 2003.

[272] V. Guffens, «Compartmental fluid-flow modelling in packet switched networks with hop-by-hop control,» Prom. : Bastin, Georges, 2005.

[273] S. Ghosh, K. Basu et S. Das, «What a Mesh! Architectures for next generation radio access networks,» *4G Network Technologies for Mobile Telecommunications Special Issue of IEEE Network*, vol. 19, n° 14, pp. 35 - 42, 2005.

[274] A. Balkan, G. Qu et U. Vishkin, «A mesh-of-trees interconnection network for single-chip parallel processing,» chez *Application-specific Systems, Architectures and Processors, 2006. ASAP '06. International Conference on*, 2006.

[275] M. Coenen, S. Murali, A. Ruadulescu, K. Goossens et G. D. Micheli, «A buffer-sizing algorithm for networks on chip using tdma and creditbased end-to-end flow control,» chez *Hardware/Software Codesign and System Synthesis, 2006. CODES+ISSS '06. Proceedings of the 4th International Conference*, 2006.

[276] I. Saastamoinen et J. Alh, «Buffer implementation for proteo networks-on-chip,» chez *Circuits and Systems, 2003. ISCAS '03. Proceedings of the 2003 International Symposium on*, 2003.

[277] M. Bakhouya, «Evaluating the energy consumption and the silicon area of on-chip interconnect architectures,» *Journal of Systems Architecture*, vol. 55, n° 17-9, p. 387 – 395, 2009.

[278] J. Jacquez et C. Simon, «Qualitative theory of compartmental systems,» *SIAM Review*, vol. 35, n° 11, p. 43–79, 1993.

[279] J. Jacquez et C. Simon, «Qualitative theory of compartmental systems xith lags,» *Elsevier: Mathematical Biosciences*, vol. 180, n° 11-2, p. 329–362, 1993.

[280] A. Pitsillides, P. Ioannou, M. Lestas et L. Rossides, «Adaptive nonlinear congestion controller for a differentiated-services framework,» *IEEE/ACM Transactions on Networking*, vol. 13, n° 11, p. 94 –107, 2005.

[281] D. Tipper et S.M.K., «Numerical methods for modeling computer networks under non-stationary conditions,» *IEEE Journal on Selected Areas in Communications*, vol. 8, n° 19, pp. 1682 - 1695, 1990.

[282] A. Borshchev et A. Filippov, «From system dynamics and discrete event to practical agent based modeling: Reasons, techniques, tools,» chez *The 22nd International Conference of the System Dynamics Society*, Oxford, England , 2004.

[283] K. Ogata, System dynamics, 4 éd., Prentice-Hall (ISBN 0-13-142462-9), 2004.

[284] C. Caulfield et S. Maj, «A case for systems thinking and system dynamics,» chez *IEEE International Conference on Systems, Man and Cybernetics*, 2001.

[285] A. Vij, «Energy policy modeling and analysis for the Indian economy,» Indian Institute of Technology, New Delhi, 1990.

[286] E. Roberts, Managerial Applications of System Dynamics, System Dynamics Series éd., Cambridge: Pegasus Communications, Inc., 1981, p. 669.

[287] R. Spencer, «Modelling strategies for corporate growth,» chez *Conference of American Association for the Advancement of Science*, Washington, 1966.

[288] M. Allalouf et Y. Shavitt, «A comparison of token-bucket based multi-color marking techniques,» chez *Proceedings of the 2006 ACM CoNEXT conference (CoNEXT'06)*, 2006.

[289] i. isee systems, «StellaSoftware,» [En ligne]. Available: http://www.iseesystems.com/softwares/Education/StellaSoftware.aspx.

[290] K. Downes, M. Ford, H. Lew, S. Spanier et T. Stevenson, Internetworking Technologies Handbook, 2 éd., Indianapolis: Cisco Systems, 1998, p. 750 .

[291] V. Guffens et G. Bastin, «Using token leaky buckets for congestion feedback control in packet switched networks with guaranteed boundedness of buffer queues,» chez *Proceedings European Control Conference ECC*, 2003.

[292] W. Jie, D. Houzet et S. Huet, «A Programming Model and a NoC-Based Architecture for Streaming Applications, Digital System Design: Architectures, Methods and Tools (DSD),» chez *13th Euromicro Conference on,*, 2010.

[293] L. Zhang, V. Fresse, M. Khalid, D. Houzet et A. Legrand, «Evaluation and Design Space Exploration of a Time-Division Multiplexed NoC on FPGA for Image Analysis Applications,» *EURASIP Journal on Embedded Systems*, 2010.

[294] M. Jabbar et O. Hammami, «Performance Analysis of a NoC-based 16PE Embedded Multicore: Processor Configuration Impact,» chez *International Design and Test, IDT 2010, IEEE*, Abu Dhabi, 2010.

[295] T. Xu, A. Yin, P. Liljeberg et H. Tenhunen, «A study of 3D Network-on-Chip design for data parallel H.264 coding,» chez *NORCHIP*, 2009.

[296] F. Darve, A. Sheibanyrad, P. Vivet et F. Petrot, «Physical Implementation of an Asynchronous 3D NoC,» chez *(ISVLSI), 2011 IEEE Computer Society Annual Symposium on*, 2011.

[297] V. Pavlidis et E. Friedman, «3-D Topologies for Networks-on-Chip,» chez *VLSI Systems, IEEE Transactions on*, 2007.

[298] K. Nomura, K. Abe, S. Fujita, Y. Kurosawa et ,. A. Kageshima, «Performance analysis of 3D-IC for multi-core processors in sub-65nm CMOS technologies,» chez *Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on*, 2010.

[299] B. Feero et P. Pande, «Networks-on-Chip in a Three-Dimensional Environment: A Performance Evaluation, Computers,» chez *IEEE Transactions on, 58*, 2009.

[300] I. Loi, P. Marchal, A. Pullini et L. Benini, «3D NoCs - Unifying inter and intra chip communication,» chez *Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on*, 2010.

[301] C. Mineo, R. Jenkal, S. Melamed et W. Davis, «Inter-die signaling in three dimensional integrated circuits,» chez *Custom Integrated Circuits Conference, CICC 2008. IEEE,*, 2008.

[302] Z. Tao, W. Kui, F. Yi, C. Yan, L. Qun, S. Bing, X. Jing, S. Xiaodi, D. Lian, X. Yuan, C. Xu et L. Youn-Long, «A 3D SoC design for H.264 application with on-chip DRAM stacking,» chez *3D Systems Integration Conference (3DIC), 2010 IEEE International*, 2010.

[303] M. Healy, K. Athikulwongse, R. Goel, M. Hossain, D. Kim, L. Young-Joon, D. Lewis, L. Tzu-Wei, L. Chang, J. Moongon, B. Ouellette, M. Pathak, H. Sane, S. Guanhao, W. Dong Hyuk, Z. Xin, G. Loh, L. H.S., L. Sung Kyu et ., «Design and analysis of 3D-MAPS: A many-core 3D processor with stacked memory,» chez *Custom Integrated Circuits Conference (CICC), 2010 IEEE*, 2010.

XI. Acronymes et Abréviations

Abréviation	Désignation
AdNoC	Adaptative NoC
AHB	Advanced High-speed Bus
APSRA	A Parallel Stereo Reconstruction Algorithm with applications in entomology
AXI	Advanced eXtensible Interface
BDF	Boolean DataFlow
BE	Best Effort
BFT	Butterfly-Fat Tree
C	Core
CABA	Cycle Accurate, Bit Accurate
CAO	Conception assistée par ordinateur
CBR	Constant Bit Rate
CCM	Central Configuration Module
CFF	Compartimental Fluid-Flow
CMOS	Complementary Metal Oxide Semiconductor
CMP	Chip Multi-Processors
Cn	Clustering degree
CPU	Central Processing Unit
CSDF	Cyclo-Static DataFlow
DDF	Dynamic DataFlow
DF	DataFlow Analysis
Dm	Distance moyenne
DMA	Direct Memory Access
DPA	Differential power analysis
DP-Network	Dynamic Programming Network
DS	Dynamique Systems
DSP	Digital Signal Processor
DTD	Document Type Definition
DyXY	Dynamique XY
E	Est
E/S	Entrée/Sortie
EVC-T	Express Virtual Channel with Taps
FAUST	Flexible Architecture of Unified System for Telecom
FIFO	First In First Out
FLIT	FLow control unIT
FPGA	Field Programmable Gate Array
FSL	Fast Simplex Link
FT	Fat-Tree
GALS	Globally Asynchronous, Locally Synchronous
GS	Guaranteed Service
GT	Guaranteed Throughput

HSDF	Homogeneous Synchronous DataFlow
ILP	Instruction-Level Parallelism
IP	Intellectual Property
IRR	Interface Ressource Réseau
LAN	Local Area Network
MOS	Metal Oxide Semiconductor
MPSoC	MultiProcessor System-on-Chip
N	Nord
NC	Network Calculus
NE	North East
NNSE	Nostrum NoC Simulation Environment
NoC	Network-on-Chip
NOP	Node-On-Path
NTTP	NoC Transaction and Transport Protocol
NW	North West
O	Ouest
OSI	Open Systems Interconnection
OCP	Open Core Protocol
PAPS	premier arrivé, premier servi
PCC	Packet Connected Circuit
PDU	Protocol Data Unit
PE	Precessing Elements
PINs	(page 15)
PSDF	Parameterized Synchronous DataFlow
QoS	Quality of Service
QT	Queuing Theory
R&D	Researche & Developpement
ReNoC	Reconfigurable NoC
RF	radio frequency
RNI	Ressource-Network Interface
RoC	Radio-on-Chip
RTL	Register Transfer Level
RTOS	Real-time operating system
S	South
SA	Schedulability Analysis
SADF	Scenario-Aware Dataflow
SCP	Self Complemented Path
SDF	Synchronous DataFlow
SDM	Spatial Division Multiplexing
SE	South East
SGM	Standard Generalized Markup
SimOS	Simulator Operating System
SMP	symmetric shared memory multiprocessor
SoC	System on Chip
SOI	Silicon On Insulator

SPA	Source Path Authentication
SPIN	Scalable, Programmable and Integrated Network
STBUS	Serial Telecom BUS
SW	South West
TBP	Trusted Boomerang Path
TC	Time Coder
TDMA	Time Division Multiple Access
TLM	Transaction Level Modeling
TQM	Total Quality Management
TR	Time-routing
UMARS	Unified MApping, Routing and Slot allocation
VC	Virtual Channel
VCI	Virtual Component Interface
VHDL	Very High Speed Integrated Circuit Hardware Description Language
ViChaR	Virtual Channel Regulator
VLSI	Very-Large-Scale Integration
VPDF	Variable Phased DataFlow
VRDF	Variable Rate DataFlow
VTT	Valtion Teknillinen Tutkimuskeskus (en finois)
WTA	Winner-Takes-All

Résumé :

Les systèmes embarqués sur puce (SoC : Systems-on-Chip) sont devenus de plus en plus complexes grâce à l'évolution de la technologie des circuits intégrés. Des études récentes ont montré que pour améliorer les performances du réseau su puce (NoC : Network-on-Chip), l'architecture de celui-ci pouvait être personnalisée, soit au moment de la conception, soit au moment de l'exécution. L'objectif principal de cette thèse est d'implémenter de nouvelles approches pour améliorer les performances des NoCs, notamment la latence, le débit, la consommation d'énergie, et la simplicité de mise en œuvre.

Nous avons proposé une approche pour permettre aux concepteurs de personnaliser l'architecture d'un NoC par insertion de liens stratégiques, pour qu'elle soit adaptée à de nombreuses applications, sous la contrainte d'un budget limité en termes de nombre de liens. L'évaluation analytique porte sur l'amélioration des paramètres physiques de la topologie du NoC sans tenir compte de l'application qui devrait s'exécuter dessus. L'évaluation par simulation porte sur l'évaluation des performances de communication du NoC. Les résultats de simulations montrent l'efficacité de notre approche pour améliorer les performances du NoC. Nous avons également introduit une approche de modélisation par réseau à compartiments pour allouer les ressources nécessaires pour chaque tampon selon le modèle de trafic de l'application cible. Les résultats de simulations montrent l'efficacité de cette approche de modélisation pour l'allocation optimisée de l'espace tampon. Enfin, nous avons proposé une approche conjointe basée sur la théorie des systèmes dynamiques pour évaluer la performance d'un algorithme de contrôle de flux dans les NoCs. Cet algorithme permet aux éléments du NoC d'ajuster dynamiquement leur entrée en utilisant un mécanisme basé sur le contrôle de flux par rétroaction. Les résultats d'évaluations analytiques et de simulation montrent la viabilité de ce mécanisme pour éviter la congestion dans les NoCs.

Mots clés : réseau sur puce ; system sur puce ; éxploration de l'espace de conception ; insertion de liens ; allocation de l'espace tampon ; personnalisation du réseau sur puce ; fractales ; contrôle de flux ; système dynamique ; simulation et évaluation de la performance.

Abstract:

Systems-on-chip (SoC) have become more and more complex due to the development of integrated circuit technology. Recent studies have shown that in order to improve the performance of a specific SoC application domain, the on-chip inter-connects (OCI) architecture must be customized at design-time or at run-time. Related approaches generally provide application-specific SoCs tailored to specific applications. The aim of this thesis is to carry out new approaches for Network-on-Chip (NoC) and study their performances, especially in terms of latency, throughput, energy consumption and simplicity of implementation.

We have proposed an approach to allow designers to customize a candidate OCI architecture by adding strategic links in order to match large application workload. The analytical evaluation focuses on improving the physical parameters of the NoC topology regardless of the application that should run on. The evaluation by simulation focuses to evaluate the communication performances of the NoC. Simulations results show the effectiveness of this approach to improve the NoC performances. We have also introduced a compartmental Fluid-flow based modeling approach to allocate required resource for each buffer based on the application traffic pattern. Simulations are conducted and results show the efficiency of this modeling method for a buffer space optimized allocation. Finally, we proposed a joint approach based on a system dynamics theory for evaluating the performance of a flow control algorithm in NoCs. This algorithm allows NoC elements to dynamically adjust their inflow by using a feedback control-based mechanism. Analytical and simulation results showed the viability of this mechanism for congestion avoidance in NoCs.

Keywords: network-on-chip; system-on-chip; design space exploration; links insertion; buffer space allocation; customization of on-chip interconnect; fractal; flow control; system dynamics; simulation and performance evaluation.

SPIM

■ École doctorale SPIM - Université de Technologie Belfort-Montbéliard

F - 90010 Belfort Cedex ■ tél. +33 (0)3 84 58 31 39

utbm université de technologie