Modélisation système d'une architecture d'interconnexion RF reconfigurable pour les many-cœurs

Alexandre Brière

UNIVERSITÉ PIERRE ET MARIE CURIE

École Doctorale Informatique, Télécommunications et Électronique
ED130

Laboratoire d'informatique de Paris 6 – Département System on Chip

Modélisation système d'une architecture d'interconnexion RF reconfigurable pour les many-cœurs

Par Alexandre BRIÈRE

Thèse de Doctorat d'Informatique

Dirigée par Pr François PÊCHEUX

Présentée et soutenue publiquement le 8 décembre 2017

Devant le jury composé de :

Dr Fabien CLERMIDY	Ingénieur chercheur (HDR), CEA	Rapporteur
Dr Gilles SASSATELLI	Directeur de recherche, LIRMM	Rapporteur
Dr Roselyne CHOTIN-AVOT	Maître de conférences (HDR), UPMC	Examinatrice
Pr Philippe COUSSY	Professeur, UBS	Examinateur
Pr Lionel LACASSAGNE	Professeur, UPMC	Examinateur
Pr Ian O'CONNOR	Professeur, ECL	Examinateur
Dr Julien DENOULET	Maître de conférences, UPMC	Co-Encadrant
Pr François PÊCHEUX	Professeur, UPMC	Directeur de thèse

THÈSE DE DOCTORAT DE
L'UNIVERSITÉ PIERRE ET MARIE CURIE

Modélisation système d'une architecture d'interconnexion RF reconfigurable pour les many-cœurs

Résumé :
La multiplication du nombre de cœurs de calcul présents sur une même puce va de pair avec une augmentation des besoins en communication. De plus, la variété des applications s'exécutant sur la puce provoque une hétérogénéité spatiale et temporelle des communications. C'est pour répondre à ces problématiques que nous présentons dans ce manuscrit un réseau d'interconnexion sur puce dynamiquement reconfigurable utilisant la Radio Fréquence (RF). L'utilisation de la RF permet de disposer d'une bande passante plus importante tout en minimisant la latence. La possibilité de reconfigurer dynamiquement le réseau permet d'adapter cette puce many-cœur à la variabilité des applications et des communications. Nous présentons les raisons du choix de la RF par rapport aux autres nouvelles technologies du domaine que sont l'optique et la 3D, l'architecture détaillée de ce réseau et d'une puce le mettant en œuvre ainsi que l'évaluation de sa faisabilité et de ses performances. Durant la phase d'évaluation nous avons pu montrer que pour un *Chip Multiprocessor* (CMP) de 1 024 tuiles, notre solution permettait un gain en performance de 13 %. Un des avantages de ce réseau d'interconnexion RF est la possibilité de faire du broadcast sans surcoût par rapport aux communications point-à-point, ouvrant ainsi de nouvelles perspectives en termes de gestion de la cohérence mémoire notamment.

Mots clefs : CMP, NoC, RF, dynamique, reconfigurable, hiérarchique, multi-cœur, many-cœur.

PHD THESIS OF
UNIVERSITY PIERRE AND MARIE CURIE

System modeling of a reconfigurable RF interconnect architecture for manycore

Abstract:
The growing number of cores in a single chip goes along with an increase in communications. The variety of applications running on the chip causes spatial and temporal heterogeneity of communications. To address these issues, we present in this thesis a dynamically reconfigurable interconnect based on Radio Frequency (RF) for intra chip communications. The use of RF allows to increase the bandwidth while minimizing the latency. Dynamic reconfiguration of the interconnect allows to handle the heterogeneity of communications. We present the rationale for choosing RF over optics and 3D, the detailed architecture of the network and the chip implementing it, the evaluation of its feasibility and its performances. During the evaluation phase we were able to show that for a CMP of 1 024 tiles, our solution allowed a performance gain of 13 %. One advantage of this RF interconnect is the ability to broadcast without additional cost compared to point-to-point communications, opening new perspectives in terms of cache coherence.

Keywords: CMP, NoC, RF, dynamic, reconfigurable, hierarchical, multi-core, manycore.

Table des matières

Table des figures

Liste des tableaux

Chapitre 1

Préambule

Sommaire

1.1 Introduction

On assiste depuis plusieurs années à un changement de stratégie dans la quête de puissance des processeurs. Au lieu d'utiliser les transistors supplémentaires pour complexifier un unique cœur de calcul et exécuter plus vite une tâche donnée, on multiplie les cœurs de calcul pour exécuter plus de tâches à la fois. On parle alors de CMP [Olukotun *et al.*, 1996]. Ainsi, le premier processeur de ce type fut le POWER4 d'IBM [Tendler *et al.*, 2002].

Ces puces intégrant de plus en plus de processeurs posent un nouveau problème : Comment faire communiquer ce nombre croissant de processeurs et de bancs mémoire efficacement ?
C'est pour répondre à ce problème que les premiers travaux sur les *Network on Chip* (NoC) ont débuté.

Pour cela, on a commencé par utiliser des technologies filaires classiques. On peut ainsi créer un réseau de routeurs permettant de mettre en place des communications par paquets dans la puce [Guerrier et Greiner, 2000]. Les paquets transitent sur le réseau, routeur après routeur, du nœud source au nœud destination. Ces routeurs sont organisés sous forme d'une grille deux dimensions. La bande passante totale est alors proportionnelle au nombre de routeurs et augmente avec la taille du NoC. La latence est aussi proportionnelle au nombre de routeurs, plus le NoC est grand, et plus le nombre de routeurs qu'il faudra potentiellement traverser pour qu'un message aille de la source à la destination sera important. Si ces NoC filaires sont intéressants en terme de bande passante, ils souffrent de certaines limitations en terme de latence, ce qui limite leur passage à l'échelle.

Pour remédier à cette limitation, il est possible de changer la topologie du réseau afin de diminuer le nombre de routeurs à traverser pour aller de la source à la destination. Dans une grille deux dimensions, on peut par exemple connecter le premier routeur d'une ligne ou colonne avec le dernier de celle-ci [Dally et Towles, 2001].

Une autre piste explorée pour remédier à cette limitation en terme de latence est de remplacer les technologies filaires classiques par de nouvelles technologies comme la 3D [Li *et al.*, 2006], l'optique [O'Connor et Gaffiot, 2004] ou la *Radio Fréquence* (RF) [Chang *et al.*, 2001]. D'une part, l'utilisation de la 3D permet de proposer de nouvelles topologies et de diminuer les distances au sein de la puce. D'autre part, l'optique et la RF permettent de faire circuler plus vite le vecteur physique de l'information et potentiellement de se passer de routeurs intermédiaires.

Si l'utilisation de ces nouvelles technologies apporte une amélioration de la bande passante et/ou de la latence, ce ne sont pas les seuls critères à prendre en compte pour évaluer les performances d'un réseau sur puce. En effet, il faut aussi tenir compte de l'hétérogénéité spatiale et temporelle des communications au sein d'un

CMP. Cette hétérogénéité est provoquée par l'exécution d'applications non régulières, ou tout simplement par l'exécution simultanée de différentes applications. Un réseau sur puce devrait donc être capable de s'adapter dynamiquement aux besoins et à la variété des applications qui s'exécutent sur un CMP.

L'objectif de cette thèse est de prendre en compte cette hétérogénéité afin d'optimiser l'exploitation des gains en débit et en latence que permet l'utilisation des nouvelles technologies dans les NoC. La problématique traitée est donc :
Comment utiliser de manière optimale les gains en terme de latence, de bande passante et de support du broadcast permis par l'utilisation de nouvelles technologies dans un NoC ?

Cette thèse présente le NoC RF *Wired RF Network on Chip reconfigurable on Demand* (WiNoCoD) réalisé dans le cadre du projet ANR du même nom.

1.2 Plan de la thèse

Le chapitre 2 présente le contexte de ces travaux. Il aborde les raisons de l'émergence des multi-cœurs, puis des many-cœurs, sous deux angles. D'une part, les raisons techniques ayant poussé à explorer ce type d'architecture. D'autre part, les raisons logicielles ayant rendu ce type d'architecture nécessaire et viable. Il présente ensuite les différents médiums de communication utilisés dans les multi-cœurs puis dans les premiers many-cœurs en présentant leurs limitations. Finalement, il présente les limitations des premiers NoC pour en extraire la problématique de cette thèse.

Le chapitre 3 présente l'état de l'art des solutions explorées pour répondre aux limitations identifiées à la fin du chapitre 2. Il commence par présenter les différentes solutions permettant d'augmenter la bande passante et/ou de diminuer la latence des NoC. Il présente ensuite les différentes méthodes permettant d'améliorer les performances d'un NoC en utilisant au mieux ses capacités grâce à l'allocation dynamique. Il présente aussi plus particulièrement les solutions explorées pour gérer les messages de type broadcast dans les NoC. Finalement, il présente en quoi ces différentes solutions ne répondent que partiellement à la problématique et les limitations restant donc à traiter.

Le chapitre 4 présente l'architecture globale du CMP utilisant WiNoCoD ainsi que les détails de ce NoC. WiNoCoD utilise une méthode d'allocation de bande utilisée notamment dans les télécommunications, l'*Orthogonal Frequency Division Multiple Access* (OFDMA). Cette technique permet de diviser la bande RF en sous-bandes, créant ainsi des canaux de communications pouvant être utilisés par les différents nœuds du réseau. WiNoCoD utilise un algorithme d'allocation distribué pour allouer dynamiquement ces canaux de communication en fonction des besoins. De

plus l'architecture proposée permet par construction de faire du broadcast sans surcoût par rapport aux communications point à point.

Le chapitre 5 présente le modèle développé pour évaluer l'architecture. Il présente tout d'abord l'environnement utilisé pour développer le modèle puis les détails d'implémentation de ce dernier.

Le chapitre 6 présente l'évaluation de la faisabilité et des performances du réseau sur puce WiNoCoD. L'évaluation de la faisabilité est réalisée grâce à une estimation de la surface et de la consommation de la réalisation physique de cette architecture. L'évaluation des performances est faite de trois façons. Tout d'abord d'un point de vue théorique, en se basant sur les caractéristiques de l'architecture. Ensuite, en générant un trafic synthétique sur le réseau. Finalement, en exécutant des applications directement sur le modèle de l'architecture.

Le chapitre 7 conclut cette thèse. Dans un premier temps, il résume les caractéristiques de la solution proposée et ses apports. Dans un deuxième temps, il aborde des travaux futurs permettant d'évaluer plus amplement le NoC RF WiNoCoD, mais aussi de ses apports possibles en dehors du cadre de l'étude.

Chapitre 2

Contexte et problématique

Sommaire

2.1 Introduction

En 1965, Gordon Moore prédit de manière empirique que la complexité des semi-conducteurs doublerait tous les ans [Moore, 1965]. En 1975, il donna une nouvelle version de ce que l'on appelle maintenant la "loi de Moore". Il prédit alors que le nombre de transistors présents sur un circuit intégré doublerait tous les deux ans [Moore, 1975].

Traditionnellement, l'augmentation du nombre de transistors intégrables sur une seule puce a été utilisée pour améliorer les performances séquentielles des processeurs. Ainsi, la complexification des processeurs a permis d'exécuter plus d'instructions par cycle. On parle alors de parallélisme d'instruction ou *Instruction Level Parallelism* (ILP). Cette méthode était encouragée par un corollaire de la loi de Moore statuant que la fréquence des transistors doublerait elle aussi tous les deux ans. Or, si la loi de Moore est encore vérifiée, son corollaire sur la fréquence des transistors ne l'est plus depuis le début des années 2000.

La fin de l'augmentation de la fréquence des transistors a donc été l'occasion de repenser l'utilisation de ce nombre toujours plus grand de transistors intégrables sur une même puce. On assiste depuis la fin des années 90 à un changement de stratégie dans la quête de puissance des processeurs [Olukotun *et al.*, 1996]. Dans cette nouvelle approche, le nombre de processeurs présents sur une même puce augmente dans le but d'exécuter plus de tâches à la fois. Ces processeurs exploitent un autre type de parallélisme, le parallélisme de thread/tâche ou *Thread Level Parallelism* (TLP). On parle alors de *Chip Multiprocessor* (CMP) ou de processeurs multi-cœurs, les cœurs faisant alors référence aux processeurs intégrés sur une même puce. Ces CMP ouvrent de nouvelles voies de recherche, notamment sur le développement de médiums de communication intégrés directement sur la puce et capables de faire communiquer efficacement les différents cœurs entre eux.

Les points évoqués ci-dessus donnent les raisons matérielles à l'avènement des architectures CMP. Cependant ces architectures répondent aussi à un besoin applicatif. Premièrement, les applications traditionnellement exécutées sur des serveurs de calcul, qui sont déjà pensées pour tirer partie de machines fortement parallèles. Ces applications sont présentes dans les benchmarks tels que PARSEC [Bienia *et al.*, 2008], SPLASH [Woo *et al.*, 1995] et les différentes versions de SPEC [Aslot *et al.*, 2001; Müller *et al.*, 2012]. Deuxièmement, les applications bureautiques ou multimédia destinées aux ordinateurs personnels. Ces dernières expriment un degré de parallélisme moindre [Blake *et al.*, 2010]. L'intérêt pour leur parallélisation est plus récent en raison de l'arrivée plus tardive d'architectures multi-processeurs dans le secteur de l'informatique grand public. Toutefois, même si ces applications expriment moins de parallélisme, elles sont souvent plusieurs à s'exécuter simultanément, augmentant ainsi le parallélisme global du code exécuté. Les CMP ont donc été développés

en partie pour répondre à un besoin logiciel et ont à leur tour influencé le développement du logiciel. La section 2.2 présente les raisons matérielles et logicielles ayant entrainé l'amélioration des performances parallèles des processeurs et l'avènement des architectures CMP.

Le premier processeur multi-cœur commercialisé fut le POWER4 d'IBM [Tendler *et al.*, 2002]. Ce circuit fut le premier à intégrer deux cœurs sur la même puce. Les générations suivantes ont continué dans la même direction à l'image du POWER8 d'IBM et de ses 12 cœurs [Fluhr *et al.*, 2015]. La section 2.3 traite de ce type de puce et des médiums de communication internes qu'elles utilisent en présentant les limitations de ces derniers.

Ces processeurs multi-cœurs utilisent différents types d'interconnexions. Ainsi, le Xeon E5 d'Intel utilise un anneau pour relier ses 18 cœurs [Bowhill *et al.*, 2016], le POWER8 d'IBM utilise 8 bus pour relier ses 12 cœurs [Fluhr *et al.*, 2015] et le Sparc T4 d'Oracle utilise un crossbar pour relier ses 8 cœurs [Shah *et al.*, 2012]. Or ces types d'interconnexion posent un double problème de passage à l'échelle. D'une part, du point de vue des performances pour le bus et l'anneau, puisque la bande passante totale y est répartie entre tous les nœuds. D'autre part, du point de vue de l'intégrabilité, puisque la surface d'un crossbar augmente de manière quadratique avec le nombre d'éléments qu'il relie. Il semble donc nécessaire d'utiliser d'autres types d'interconnexion pour continuer à augmenter le nombre de cœurs intégrés sur une même puce. Ces nouveaux types d'interconnexion sont inspirés des réseaux utilisés dans les systèmes à plus grande échelle, on parle alors de *Network on Chip* (NoC). Dans la suite de cette thèse, nous parlerons de many-cœurs pour les CMP utilisant des NoC alors que le terme multi-cœur sera réservé aux CMP utilisant des interconnexions de type bus, anneau et cross bar.

Les premiers NoC à avoir été proposés utilisent des technologies filaires classiques. Ils sont constitués d'un réseau de routeurs permettant la mise en place de communications par commutation de paquets [Guerrier et Greiner, 2000]. Dans ce type de circuit, l'information est transmise sous forme de paquets composés des données à transmettre et de la destination de ces données. Chaque nœud du réseau peut ainsi savoir vers lequel des nœuds qui lui sont directement connectés il doit transmettre le paquet. Cette approche s'oppose aux réseaux par commutation de circuit qui établissent un chemin complet entre la source et la destination préalablement à l'envoie des données. Dans la suite de cette thèse, nous parlerons de NoC filaires classiques pour qualifier ce type de réseau à commutation de paquet. Dans ces NoC filaires classiques, la bande passante totale est proportionnelle au nombre de routeurs et augmente avec la taille du NoC. La latence est aussi proportionnelle au nombre de routeurs : plus le NoC est grand, et plus le nombre de routeurs qu'il faudra potentiellement traverser pour qu'un message aille de la source à la destination sera important. Si ces NoC filaires classiques sont intéressants en terme de bande

passante, ils peuvent donc souffrir de certaines limitations en terme de latence, ce qui limite leur passage à l'échelle. Par ailleurs, toutes les communications circulant sur un NoC ne sont pas nécessairement point à point. Certains messages doivent ainsi être envoyés à plusieurs cœurs ou même à la totalité des cœurs, on parle de multicast et de broadcast. De plus, il faut aussi tenir compte de l'hétérogénéité spatiale et temporelle des communications au sein d'un CMP. Cette hétérogénéité est provoquée par l'exécution d'applications non régulières, ou tout simplement par l'exécution simultanée de plusieurs applications différentes. Les NoC filaires classiques présentent donc certaines limitations en terme de latence, de bande passante, de support du broadcast et d'adaptabilité. La section 2.4 traite des many-cœurs, de l'introduction des premiers NoC et de leurs limitations.

Finalement, la section 2.5 résume les différents points de ce chapitre en insistant plus particulièrement sur les limitations des premiers NoC. La problématique de cette thèse est alors définie en s'appuyant sur ces constatations.

2.2 Les besoins applicatifs

Cette section présente le fonctionnement haut niveau d'un ordinateur et les raisons ayant poussé au développement des architectures CMP. La sous-section 2.2.1 présente l'architecture d'un ordinateur. La sous-section 2.2.2 présente les différents niveaux de parallélisme. La sous-section 2.2.3 présente les caractéristiques des applications susceptibles d'être exécutées par un ordinateur.

2.2.1 Architecture d'un ordinateur

Un ordinateur est une machine capable d'exécuter de manière contrôlée une séquence d'instructions traitant des données, le tout stocké dans une mémoire, en interagissant avec son environnement. Il est à minima composé d'une *Unité de Contrôle* (UC) ou *Control Unit* (CU) en anglais, d'une *Unité Arithmétique et Logique* (UAL) ou *Arithmetic Logic Unit* (ALU) en anglais, d'une mémoire et d'entrées/sorties. On parle d'architecture de von Neumann [1945]. L'UC orchestre le fonctionnement des autres composants de l'ordinateur. L'UAL effectue les traitements indiqués par les instructions sur les données qui lui sont fournies. L'UC et l'UAL forment l'*Unité Centrale de Traitement* (UCT) ou *Central Processing Unit* (CPU) en anglais. La mémoire stocke les instructions et les données. Finalement, le périphérique d'entrées/sorties permet à l'ordinateur de communiquer avec son environnement.

Si nous regardons de plus près le fonctionnement de l'UCT, l'exécution d'une instruction peut être décomposée en plusieurs étapes. L'instruction est tout d'abord chargée depuis la mémoire, puis décodée pour connaitre le traitement à effectuer et

les données concernées. Il faut ensuite charger ces données, les traiter, puis potentiellement écrire le résultat du traitement en mémoire. Ces différentes étapes sont effectuées par différentes parties de l'UCT, celles-ci ne sont donc pas utilisées en permanence mais à tour de rôle. C'est pour améliorer cette utilisation qu'ont été proposées les premières UCT pipelinées. Dans ce type d'architecture, l'UCT n'attend pas que l'exécution d'une instruction soit terminée pour commencer le traitement de la suivante. Une étape de l'exécution d'une instruction peut commencer dés que cette étape est terminée pour l'instruction précédente. Chaque étape d'exécution constitue alors un étage du pipeline du processeur. Cela permet d'augmenter la fréquence d'horloge du processeur, un cycle ne correspond plus au temps de traitement d'une instruction complète mais au temps maximum de traitement d'un étage du pipeline. Le nombre d'étages du pipeline peut varier, allant des 3 étages du premier processeur pipeliné, le Stretch d'IBM [Bloch, 1959], jusqu'aux 31 étages de certains Pentium 4 d'Intel [Jagannathan *et al.*, 2005]. Cependant, les processeurs actuels sont revenus à des pipelines de taille intermédiaire et utilisent majoritairement des pipelines allant de 14 à 20 étages.

2.2.2 Les différents niveaux de parallélisme

2.2.2.1 Parallélisme d'instruction

Comme nous l'avons vu dans la sous-section 2.2.1, le flot d'exécution d'un processeur peut être découpé en plusieurs étages pour former un pipeline. Or, il existe des contraintes sur les interactions entre les différents étages du pipeline : on parle d'aléas. Il existe trois types d'aléas : structurels, de données et de contrôle.

Un aléa structurel correspond à l'utilisation d'un même composant matériel par plusieurs étages du pipeline. L'exemple le plus simple est l'accès simultané par deux étages différents aux instructions et aux données contenues en mémoire. Cet aléa peut être résolu par l'utilisation de deux mémoires caches séparées, une pour les instructions et une pour les données.

Un aléa de données correspond à l'utilisation d'une même donnée par plusieurs étages du pipeline. Cet aléa peut-être réglé de plusieurs manières, la manière la plus simple étant de retarder le lancement de la deuxième instruction jusqu'à la fin de la première. Cependant de meilleures solutions existent : il est par exemple possible d'exécuter les instructions dans un ordre différent de celui décidé par le programmeur lors de l'écriture du code. Cela permet d'exécuter effectivement une instruction par cycle en insérant des instructions indépendantes entres deux instructions présentant des dépendances entre elles. On parle alors d'exécution dans le désordre [Tomasulo, 1967].

Finalement, les aléas de contrôle correspondent aux instructions de branchement conditionnelles utilisées pour implémenter les boucles et les tests. Or le choix de l'instruction à exécuter après une instruction de branchement conditionnel dépend directement du résultat de cette dernière. Si le branchement est pris, il ne faut pas exécuter l'instruction se trouvant à l'adresse suivante mais celle qui se trouve à l'adresse spécifiée par le branchement. Ceci implique de devoir soit attendre la fin de l'instruction de branchement pour lancer la suivante, soit de pouvoir annuler l'exécution de l'instruction suivante si le branchement est pris. C'est pour s'affranchir de cette contrainte que les prédicteurs de branchement ont été proposés. Cette solution part du constat que le résultat d'un branchement donné a de fortes chances d'être plusieurs fois de suite le même.

Au regard de ces différents aléas, il est possible de séparer le flot d'instructions exécutées par le processeur en deux catégories. D'une part, les instructions pouvant introduire des aléas dans le pipeline : ces instructions doivent être exécutées avec un certain délai d'attente entre chacune et dans un ordre précis. D'autre part, les instructions qui ne présentent pas de dépendances entre elles : ces instructions peuvent donc être lancées dans le pipeline au rythme d'une par cycle, potentiellement dans un ordre différent de celui du code de départ et même en parallèle.

L'exécution dans le désordre et la prédiction de branchement permettent de maintenir une utilisation optimale du pipeline. Ils peuvent être couplés à une autre méthode d'amélioration des performances : l'utilisation de plusieurs pipelines en parallèle. On parle alors de processeur superscalaire. Ces trois techniques sont alors complémentaires. D'une part, l'exécution dans le désordre et la prédiction de branchement permettent l'utilisation optimale des pipelines d'un processeur superscalaire. D'autre part, l'utilisation de plusieurs pipelines permet de tirer pleinement partie des deux autres méthodes capables de fournir plus d'une instruction par cycle. Il devient alors possible d'exécuter réellement plusieurs instructions par cycle. On parle alors de parallélisme d'instruction ou *Instruction Level Parallelism* (ILP).

2.2.2.2 Parallélisme de tâche

L'utilisation des pipelines permet d'augmenter la fréquence de fonctionnement d'un processeur. L'utilisation de techniques telles que l'exécution dans le désordre et la prédiction de branchement permet au pipeline d'approcher un fonctionnement idéal d'une instruction chargée par cycle. L'utilisation de processeurs superscalaires permet d'améliorer encore les performances et de charger plusieurs instructions par cycle. Cependant ces techniques présentent certaines limites. D'une part, ces méthodes ont un coût matériel important. Par exemple, plus le nombre de pipelines d'un processeur superscalaire augmente, plus le module d'exécution dans le désordre doit être efficace, devenant de plus en plus complexe. D'autre part, le code

exécuté impose lui-même certaines limitations. Il présente toujours un caractère séquentiel plus ou moins important et certaines instructions doivent impérativement être exécutées l'une après l'autre dans un ordre précis.

Il est alors nécessaire de chercher un autre moyen d'extraire du parallélisme du code exécuté pour exploiter efficacement le nombre croissant de transistors disponibles. L'ILP est un parallélisme bas niveau mais le code exécuté par un ordinateur présente d'autres niveaux de parallélisme. Ainsi, à plus haut niveau il est possible de faire apparaître du parallélisme de tâche ou *Thread Level Parallelism* (TLP). Le terme thread étant bien à prendre ici au sens large de tâche puisqu'il peut s'agir d'un thread au sens logiciel, d'un processus ou même d'une application entière. Le terme de tâche comme thread logiciel, ou processus léger, correspond généralement à un découpage à grain fin du fonctionnement d'une application. Ce découpage peut par exemple correspondre à un traitement identique réalisé sur des sous-parties d'un ensemble de données, ou encore à des traitements différents sur un même ensemble de données. Le terme de tâche en tant que processus classique correspond généralement à un découpage à plus gros grain, il peut par exemple s'agir de la gestion de différentes fonctionnalités d'une application. Finalement, le terme de tâche peut se référer à une application complète, le TLP correspondra ici à l'exécution simultanée de plusieurs applications sur un même ordinateur.

C'est l'exploitation de ce type de parallélisme qui a poussé au développement des processeurs à plusieurs cœurs de calcul. Si les optimisations architecturales qui cherchent à exploiter le parallélisme d'instruction gardent toujours un unique *Program Counter* (PC), ce n'est pas le cas des CMP. Une des caractéristiques principales d'un CMP est justement d'avoir un PC par cœur afin de pouvoir exécuter plusieurs tâches en parallèle. Ainsi, chaque cœur peut-être vu comme une UCT à part entière capable d'exécuter une tâche de manière autonome. Il est donc important de mettre en place un médium de communication reliant efficacement ces différents cœurs.

2.2.3 Les différentes classes d'applications

Avant de décrire plus en détail l'architecture des différents CMP, il est nécessaire de caractériser les applications qu'ils exécutent. La première approche retenue est de diviser ces applications en trois catégories [Wang *et al.*, 2009]. Les applications de bureautique, les applications multimédia et les applications de calcul scientifique ou *High Performance Computing* (HPC). La deuxième approche retenue est de suivre la classification proposée par Asanović *et al.* [2006], les "nains de Berkeley". Les applications sont alors classées selon les caractéristiques de l'algorithme utilisé, en terme de calculs et de communications.

2.2.3.1 Classification par domaine applicatif

Il est possible d'évaluer le degré de parallélisme d'une application de deux manières : soit par rapport à son speedup théorique, soit par rapport à son TLP effectif. Dans les deux cas, la valeur est obtenue en supposant un nombre de cœurs/processeurs suffisamment grand pour que ce soit le logiciel qui limite le parallélisme et non le matériel. Le speedup théorique d'une application correspond à la diminution maximale de son temps d'exécution. Le TLP effectif correspond au nombre maximal de tâches d'une même application pouvant s'exécuter simultanément.

Applications bureautiques Nous rangeons dans cette catégorie les applications telles que la compression de fichier, la compilation, les suites bureautiques ou les navigateurs internet. D'après Wang *et al.* [2009], les applications telles que la compression ou la compilation ont un speedup théorique maximal de $\times 1,4$. De plus, il est possible d'obtenir 80 % de cette valeur théorique maximale avec 2 cœurs. D'après Blake *et al.* [2010], le TLP effectif des applications telles que les suites bureautiques est de 1,2 alors que celui des navigateurs internet est de 2,0. Dans les deux cas, il ressort donc que les applications de bureautique n'exploitent pas un grand nombre de cœurs et ne semblent même pas en avoir besoin, tel qu'elles sont programmées actuellement. Cependant, même si ces valeurs sont assez basses, ces applications peuvent être plusieurs à s'exécuter en même temps.

Applications multimédia Nous rangeons dans cette catégorie les applications de traitement et de compression d'image, de son ou de vidéo ainsi que les jeux vidéo. D'après Wang *et al.* [2009], le speedup théorique maximal de ce type d'application est de $\times 3,3$. Il est possible d'atteindre 60 % de ce speedup avec 2 cœurs et 80 % avec 8 cœurs. D'après Blake *et al.* [2010], le TLP effectif est de l'ordre de 2,9 pour ce type d'application. Tout comme pour les applications bureautiques, les applications de type multimedia peuvent être plusieurs à s'exécuter en même temps sur une machine. Il peut donc y avoir un TLP au niveau application sur la machine auquel il faut ajouter un TLP au niveau thread au sein de chaque application.

Applications HPC Pour caractériser les applications de type HPC nous avons retenu les benchmarks tels que SPLASH [Woo *et al.*, 1995] et PARSEC [Bienia *et al.*, 2008]. D'après Woo *et al.* [1995], le speedup moyen atteignable pour les applications de SPLASH est de $\times 50$ sur une machine à 64 processeurs. La pente de la courbe représentant le speedup en fonction du nombre de processeurs ne change quasiment pas entre 32 et 64 processeurs pour 7 applications sur 12. D'après Bienia *et al.* [2008], le speedup moyen atteignable pour les applications de PARSEC est de $\times 14$ sur une machine à 16 cœurs. Tout comme pour SPLASH, et même si le nombre de cœurs est moindre, la pente de la courbe représentant le speedup en fonction du nombre de cœurs ne change quasiment pas entre 8 et 16 cœurs pour 9 applications sur 12. Ainsi ces deux benchmark montrent que les applications de type HPC peuvent tirer

parti d'architectures à 64 cœurs et laissent à penser qu'il serait possible d'exploiter un encore plus grand nombre de cœurs. De plus, si les benchmark SPLASH et PARSEC ont été développés dans le but d'évaluer les architectures à mémoire partagée cohérente, ce n'est pas le seul modèle de CMP possible. Il est tout à fait possible d'envisager les architectures CMP fonctionnant par passage de messages. De telles architectures pourraient alors exécuter des applications de calculs scientifiques capables de tirer parti de plusieurs milliers de cœurs [Raponi *et al.*, 2011]

Analyse Suivant cette classification, il apparait qu'en fonction du domaine considéré, les applications peuvent ou non exploiter un grand nombre de cœurs. Ainsi les applications bureautiques peuvent se contenter de deux cœurs et les applications multimédia de huit cœurs. Cependant, ces applications peuvent être plusieurs à s'exécuter sur une même machine, augmentant ainsi le nombre de cœurs utilisés à quelques dizaines. De plus, le dernier domaine applicatif, le HPC, peut tirer parti de machines ayant une centaine de cœurs ou plus. Il apparait donc que le développement de CMP à plusieurs dizaines ou même centaines de cœurs peut être justifié par l'utilisation effective d'autant de cœurs par les applications exécutées.

2.2.3.2 Classification des applications : les nains de Berkeley

Cette classification a été introduite par Asanović *et al.* [2006] dans le cadre de travaux réalisés par un groupe pluridisciplinaire de chercheurs de l'Université de Californie à Berkeley. Le but de ces travaux était de définir des pistes d'évolution pour l'architecture des ordinateurs et des modèles de programmation associés suite au passage des architectures mono-cœur aux CMP.

Le principe est de classifier les différents algorithmes existants pour ensuite caractériser les calculs et les communications qu'ils génèrent. Cette classification doit permettre de s'abstraire des dépendances des benchmarks vis-à-vis de leur implémentation en proposant une méthode plus générale d'évaluation des performances d'une architecture. Un algorithme est ainsi rangé dans la classe dont il est le plus proche parmi les 13 identifiées. Une première version de cette classification ne comportait que 7 classes d'applications, c'est ce nombre qui est à l'origine du nom en référence aux 7 nains de "Blanche-Neige". Le Tableau 2.1 présente la principale limitation pour chacune de ces 13 classes parmi la puissance de calcul, la bande passante et la latence. Cependant, les limitations des algorithmes de type *Machines à états* n'y apparaissent pas car ces algorithmes sont difficilement parallélisables.

Applications limitées par la puissance de calcul Parmi les 13 classes d'algorithmes, 7 sont limitées par la puissance de calcul. C'est par exemple le cas des algorithmes de type *MapReduce*. Ce type d'algorithme est utilisé pour les problèmes où il est possible de traiter les données indépendamment les unes des autres. Il permet par exemple de compter le nombre d'occurrences d'un symbole dans un ensemble

Classe d'algorithme	**Facteur limitant**		
	Puissance de calcul	Bande passante	Latence
Algèbre linéaire (matrices denses)	✓		
Algèbre linéaire (matrices creuses)	✓	✓	
Méthodes spectrales			✓
Problèmes à N corps	✓		
Grilles structurées		✓	
Grilles non structurées			✓
MapReduce	✓		
Logique combinatoire	✓	✓	
Parcours de graphe			✓
Programmation dynamique			✓
Branch-and-Bound	✓		✓
Modèle graphique	✓		
Machines à états	–	–	–

Tableau 2.1 – Facteur limitant les performances de chaque classe d'algorithmes parallélisé ***(extrait des travaux d'Asanović* et al. *[2006])***

de fichiers. L'étape *Map* consiste alors à compter indépendamment dans chaque fichier le nombre d'occurrences du symbole alors que l'étape *Reduce* consiste à sommer les résultats. Les communications ne sont nécessaires qu'au début pour distribuer le travail et à la fin pour mettre en commun les résultats, faisant de la puissance de calcul le critère limitant.

Applications limitées par la bande passante Parmi les 13 classes d'algorithmes, 3 sont limitées par la bande passante. C'est par exemple le cas des algorithmes de type *Grilles structurées*. Les problèmes traités par cette classe d'algorithmes sont stockés sous la forme de matrice à deux dimensions ou plus, découpée en sous-matrices stockées localement. Chaque point de la matrice est mis à jour en fonction de ses proches voisins. Pour les points situés sur les bords des sous-matrices, il faut donc accéder aux points situés sur les bords des sous-matrices adjacentes. Cet accès peut être anticipé de manière à en masquer la latence mais la bande passante nécessaire est importante et constitue donc le facteur limitant.

Applications limitées par la latence Parmi les 13 classes d'algorithmes, 5 sont limitées par la bande passante. C'est par exemple le cas des algorithmes de type *Programmation dynamique*. Comme pour d'autres classes d'algorithmes, le but de la programmation dynamique est de découper un problème en sous-problèmes pouvant être résolus séparément. Un exemple de programmation dynamique est la résolu-

tion du problème bien connu du voyageur de commerce. Le plus court chemin entre deux villes A et B est composé des plus courts chemins entre plusieurs couples de villes A_i et B_i. Or, la différence avec les autres classes d'algorithme vient de la possibilité de réutiliser un résultat intermédiaire d'un sous-problème dans un autre. Dans le cas du voyageur de commerce, le chemin entre deux villes A_i et B_i est un résultat intermédiaire puisqu'il peut être utilisé dans plusieurs chemins possibles entre A et B. La latence des communications est alors déterminante pour permettre de communiquer ces résultats intermédiaires le plus vite possible pour ne pas bloquer la résolution des sous-problèmes qui en dépendent.

Analyse Le Tableau 2.1 montre que la puissance de calcul apparait comme un facteur limitant pour sept classes d'algorithmes. Par ailleurs, que ce soit en terme de bande passante ou de latence, les communications apparaissent comme un facteur limitant pour huit classes d'algorithmes. Il apparait donc que la puissance de calcul et les performances des communications sont tout aussi importantes l'une que l'autre. Par ailleurs, si l'analyse est recentrée sur les communications, la latence semble être plus souvent un facteur limitant que la bande passante. La latence devrait donc être un critère important dans le développement des médiums de communication utilisés dans les CMP.

2.2.3.3 Synthèse

Dans cette sous-section, nous avons analysé les besoins logiciels ayant entrainé le développement des CMP et guidant leur évolution. Pour ce faire, nous nous sommes basés sur deux méthodes de classification des applications existantes. D'une part, la classification par domaine applicatif nous a permis d'évaluer la capacité des applications à utiliser des puces possédant plusieurs cœurs. Ainsi, en fonction du domaine et de l'exécution simultanée ou non de plusieurs applications, il est possible de tirer parti de plusieurs dizaines de cœurs ou même plusieurs centaines. D'autre part, la classification par type d'algorithme nous a permis d'identifier les critères à prendre en compte dans le développement des CMP. Nous avons ainsi pu constater que la puissance de calcul et les performances des communications avaient une part identique d'importance dans les performances d'un CMP. Finalement, nous avons remarqué que parmi les différents critères de performances des communications, la latence semblait plus souvent être un facteur limitant que la bande passante.

2.2.4 Conclusion

Dans cette section, nous avons présenté les raisons matérielles et logicielles de l'émergence des CMP ainsi qu'une première évaluation des critères à prendre en compte

dans leur développement. D'une part, les multi-cœurs répondent à une problématique venant du logiciel : exploiter les différents niveaux de parallélisme. Alors que l'utilisation de processeurs superscalaires permet d'exploiter l'ILP, l'utilisation de plusieurs cœurs permet d'exploiter le TLP. D'autre part, les multi-cœurs répondent à une problématique venant du matériel : continuer à offrir un gain en performance proportionnel à l'augmentation du nombre de transistors. Le nombre de transistors par unité de surface continue pour l'instant d'augmenter conformément à la loi de Moore mais il ne s'accompagne plus d'une augmentation en fréquence systématique. De plus, plus le degré d'un processeur superscalaire est important plus le matériel permettant de maintenir les pipelines en charge devient complexe. La multiplication des cœurs de calcul d'un processeur apparait alors comme un moyen simple d'augmenter le nombre d'instructions exécutées par cycle. Finalement, l'analyse des différents types d'algorithmes exécutés nous a montré qu'hormis la puissance de calcul des cœurs, le principal frein à l'amélioration des performances permise par l'exploitation du parallélisme de tâche semblait être la latence des communications.

2.3 Les multi-cœurs

Comme nous l'avons vu dans la section précédente, le passage des processeurs mono-cœur aux processeurs multi-cœurs s'est fait pour deux raisons. Coté matériel, il apparait que c'est l'exploitation de plus en plus complexe du parallélisme d'instruction qui a ouvert la voie à l'exploitation du parallélisme de tâches. Coté logiciel, il apparait que c'est justement la présence de parallélisme de tâche dans le code exécuté qui a justifié le développement de puces possédant plusieurs cœurs. Dans cette section, nous présentons un aperçu des processeurs multi-cœurs existants, de leurs médiums de communications ainsi que des limitations de ces derniers. Les CMP utilisant des NoC rentrent dans la famille des many-cœurs et seront présentés dans la section suivante.

2.3.1 Architectures existantes

Dans cette sous-section, nous présentons un panorama des architectures multi-cœurs existantes. Ce panorama montre l'évolution de la conception, du nombre de cœurs et des caches des processeurs multi-cœurs.

2.3.1.1 Architectures multi-cœurs par assemblage

La méthode la plus simple pour concevoir une architecture multi-cœur est de réutiliser les processeurs mono-cœurs existant. Avec cette méthode, les communications entre les cœurs ne se font pas directement sur la puce mais via un médium externe.

SUN UltraSPARC IV En 2004, SUN lance son premier processeur multi-cœur, l'UltraSPARC IV [Krewell, 2003]. Ce processeur est basé sur la génération précédente de processeurs SUN et est constitué de deux UltraSPAR III. Chaque cœur possède un cache L1 pour les données et un pour les instructions, ainsi qu'un contrôleur de cache L2. Cependant le cache L2 lui-même se trouve en dehors de la puce. Comme sur la plupart des CMP possédant un faible nombre de cœurs, la cohérence se fait par snooping.

Intel Pentium D En 2005, Intel lance son premier CMP, le Pentium Extreme Edition 840 [Douglas, 2005], premier représentant de la série des Pentium D. Il possède deux cœurs ayant chacun un cache L2 et deux caches L1, un pour les instructions et un pour les données. Cependant, les caches L2 de chaque cœur ne pouvant pas communiquer au sein de la puce, la gestion de la cohérence passe par le bus système et le contrôleur mémoire.

2.3.1.2 Architectures multi-cœurs par conception

Si l'assemblage de processeurs mono-cœurs permet de développer rapidement une puce multi-cœur, le résultat est aussi plus proche d'une solution multi-processeur que d'un réel CMP. Ainsi, dans les puces multi-cœurs conçues comme telles, les communications passent soit par un médium de communication interne, soit par des caches partagés.

IBM POWER4 En 2001, le premier processeur multi-cœur est lancé par IBM, le POWER4 [Tendler *et al.*, 2002]. Il possède deux cœurs ayant chacun un cache instruction et un cache donnée. Ces deux cœurs partagent un cache L2 auquel ils sont reliés par un crossbar. Il possède aussi un contrôleur de cache lui permettant d'accéder via un bus à un cache L3 situé en dehors de la puce. La cohérence du POWER4 est assurée par snooping.

SUN UltraSPARC T1 (Niagara) En 2005, SUN lance un processeur à huit cœurs, l'UltraSPARC T1 [Kongetira *et al.*, 2005]. Contrairement à son prédécesseur, l'UltraSPARC IV, le deuxième niveau de cache se trouve ici intégralement sur la puce. Le processeur possède quatre bancs de cache L2 accessibles par tous les cœurs grâce à un crossbar. La gestion de la cohérence est assurée par le cache L2 et repose sur un protocole à base de répertoire.

Intel Core En 2006, Intel lance plusieurs processeurs basés sur la micro-architecture Core [Doweck, 2006]. Contrairement au Pentium D, les cœurs situés sur une même puce peuvent maintenant communiquer sans passer par le bus système. Pour ce faire, les différents cœurs du CMP partagent un cache L2 unifié.

2.3.1.3 L'augmentation du nombre de cœurs

Si les premiers CMP n'étaient constitués que de deux cœurs, ce nombre a augmenté avec les architectures suivantes. De plus, cette augmentation du nombre de cœurs s'est accompagnée d'une augmentation du nombre de niveaux de cache. L'utilisation de plusieurs niveaux de cache permet alors de faciliter les communications entre les cœurs et de partager plus efficacement les données.

Intel Haswell En 2013, Intel lance Haswell, la quatrième génération de micro-architecture Core [Hammarlund *et al.*, 2014]. Les processeurs utilisant cette micro-architecture peuvent posséder de 2 à 22 cœurs. Chaque cœur possède maintenant des caches L1 et L2 privés alors que le cache L3 est partagé par tous les cœurs de la puce. Les caches L3 sont reliés entre eux par un anneau permettant aux différents cœurs du CMP de communiquer entre eux. Comme pour les générations précédentes de CMP Intel, la cohérence mémoire fonctionne par snooping.

Oracle UltraSPARC T5 En 2013, Oracle lance un processeur à seize cœurs, l'UltraSPARC T5 [Feehrer *et al.*, 2013]. Chaque cœur possède un cache L1 instruction et un cache L1 données ainsi qu'un cache L2. De plus, la puce possède un cache L3 partagé par tous les cœurs. Les communications entre le cache L3 et les cœurs passent par un crossbar. Comme pour l'UltraSPARC, la gestion de la cohérence est assurée par un mécanisme de répertoire mis en place dans les caches L1 et L2.

IBM POWER8 Depuis le POWER4, les processeurs IBM ont continué d'évoluer pour arriver à la génération actuelle, le POWER8 [Starke *et al.*, 2015]. Il possède 12 cœurs ayant chacun un cache L1 pour les instructions et un pour les données ainsi qu'un cache L2. Contrairement au POWER4, le cache L3 est maintenant intégré sur le CMP. De plus, le POWER8 introduit un niveau supplémentaire de cache L4 situé sur une puce dédiée qui est chargée de gérer les communications avec la mémoire vive. Les communications à l'intérieur du CMP utilisent huit bus. Pour limiter le trafic généré par la cohérence, le POWER8 utilise un protocole hybride basé à la fois sur des répertoires et du snooping.

2.3.1.4 Synthèse

Dans cette sous-section, nous avons présenté l'évolution des CMP à travers différents processeurs proposés par IBM, Intel et SUN. Les premières puces proposées par SUN et Intel, respectivement en 2004 et 2005, n'étaient pas réellement des processeurs multi-cœurs mais un assemblage de deux processeurs mono-cœur. De son côté, IBM proposa en 2001 le premier CMP, le POWER4. Ce processeur était déjà intégralement conçu comme une puce multi-cœur, ce qui ne sera le cas qu'à partir de la deuxième génération des processeurs SUN et Intel, respectivement en 2005 et

2006. Finalement, pour ces trois constructeurs, la tendance actuelle est à l'augmentation du nombre de cœurs qui s'accompagne d'une augmentation du nombre de niveaux de cache.

2.3.2 Médiums de communication utilisés

2.3.2.1 Bus

C'est le type d'interconnexion le plus simple à implémenter. Un contrôleur reçoit les demandes d'allocation de la part des différents maîtres connectés au bus. Une fois le bus alloué par le contrôleur à un maître, celui-ci peut commencer à émettre vers une cible identifiée par son adresse. Le bus présente l'avantage d'avoir une latence des communications fixe quel que soit le maître et la cible. Cependant, comme le bus est alloué à un unique maître à la fois, le nombre d'éléments connectés a un impact direct sur la bande passante et la latence. Plus il y a d'éléments connectés, plus la bande passante disponible par élément diminue et plus un élément devra potentiellement attendre avant de se voir attribuer le bus. De plus, la diminution de la finesse de gravure permet une augmentation de la taille des circuits à l'échelle du transistor. Il devient donc de plus en plus contraignant d'implémenter un réseau pouvant traverser toute la puce en un seul cycle. Ce type d'interconnexion ne passe donc pas à l'échelle.

2.3.2.2 Anneau

Dans un anneau, chaque nœud n'est connecté directement qu'à ses deux plus proches voisins. Il est donc possible d'établir plusieurs communications en parallèle. Cependant, si un nœud doit transmettre un message à un nœud n'étant pas son voisin direct, le message doit obligatoirement passer par les nœuds intermédiaires. Ce fonctionnement présente donc deux inconvénients. D'une part, un message consomme une part variable de la bande passante totale en fonction de la distance entre la source et la destination. D'autre part, la latence moyenne des communications augmente linéairement avec le nombre de nœuds connectés. Ainsi, même si le coût de ce type d'interconnexion varie linéairement avec le nombre de nœuds, l'augmentation de la latence limite le passage à l'échelle des anneaux.

2.3.2.3 Crossbar

Un crossbar permet à des paires distinctes de communiquer en parallèle ce qui permet à ce type de réseau d'offrir une bande passante très importante. De plus, la latence des communications est la même quelque soit la source et la destination Le

nombre d'éléments connectés n'influence donc ni la bande passante ni la latence des communications. Cependant, le passage à l'échelle de ce type d'interconnexion est limité car sa complexité matérielle augmente quadratiquement avec le nombre d'éléments connectés.

2.3.3 Limitations

La principale limitation des médiums de communications utilisés dans les architecture multi-cœurs est le passage à l'échelle. Que ce soit en terme de coût matériel, comme pour le crossbar, ou en termes de performances, comme pour le bus et l'anneau, ces médiums s'adaptent difficilement à l'augmentation du nombre de cœurs. C'est donc pour répondre aux limitations de ces médiums de communication que les premiers NoC ont été proposés et ont permis le développement des architectures many-cœurs que nous verrons dans la section suivante.

2.4 Les many-cœurs

Comme expliqué dans la section 2.1, nous avons séparé les CMP en deux catégories en fonction du type d'interconnexion utilisé. Nous appelons multi-cœurs les CMP utilisant des interconnexions et many-cœurs ceux basés sur des NoC. Dans la section précédente, nous avons présenté plusieurs architectures multi-cœurs et les médiums de communications utilisés, bus, anneaux et crossbar. Or, toutes ces interconnexions présentent à des degrés différents le même inconvénient : elles passent difficilement à l'échelle et limitent donc le nombre de cœurs intégrables sur un CMP. Cette section présente un panorama des architectures many-cœurs et des NoC utilisés.

2.4.1 Les architectures many-cœurs existantes

Dans cette sous-section, nous nous concentrons sur les many-cœurs utilisant les réseaux filaires de type grille qui sont les premiers NoC à avoir été proposés. Nous présentons plusieurs solutions existantes sans rentrer dans les détails des réseaux utilisés qui seront présentés pour certains dans la sous-section suivante.

RAW Ce CMP, proposé par Taylor *et al.* [2002], est composé de 16 cœurs reliés par un NoC de type grille. Il possède ainsi un nombre de cœurs équivalent à certains multi-cœurs présentés dans la section précédente bien qu'ayant été développé 10 ans plus tôt. De plus, le fait qu'il utilise un NoC le classe dans la catégorie des many-cœurs selon les critères retenus. Ce processeur est divisé en tuiles qui sont arrangées suivant une grille 4×4. Chaque tuile contient un cœur, un cache instruction, un cache donnée ainsi que les routeurs nécessaires à ces différents réseaux. Ainsi, le

processeur RAW possède quatre réseaux, deux statiques et deux dynamiques. Les réseaux statiques utilisent des routeurs programmables pour faire de la commutation de circuit. Ainsi à chaque cycle, tous les routeurs d'un réseau statique exécutent une instruction et adoptent une certaine configuration de chemins possibles. Les réseaux dynamiques fonctionnent par commutation de paquet. Les données envoyées sur ces réseaux sont précédées d'un en-tête indiquant la destination, permettant ainsi à chaque routeur de choisir le destinataire suivant du paquet.

Tile64 Ce CMP orienté multimédia est développé par la société Tilera, il dispose de 64 cœurs et utilise un NoC de type grille Bell2008. Chaque cœur est inclus dans une tuile qui contient en plus un cache L1 instruction, un cache L1 donnée, un cache L2 et les routeurs permettant d'accéder à ses réseaux. Il possède cinq réseaux, un statique par commutation de circuit et quatre dynamiques par commutation de paquet Parmi les réseaux dynamiques, deux sont utilisés pour le transfert de données sur un modèle de commande/réponse permettant le transfert de ligne de cache. Les deux autres réseaux dynamiques fonctionnent par passage de messages. Il est intéressant de noter que le réseau statique et un des deux réseaux dynamiques par passage de messages sont directement accessibles au logiciel.

SCC Le *Single-chip Cloud Computer* est un CMP de 48 cœurs développé par Intel [Howard *et al.*, 2010]. Tout comme les deux précédents CMP, le SCC est découpé en tuiles arrangées sous forme de grille, il présente cependant plusieurs différences. Une tuile contient ici deux cœurs et un routeur, chaque cœur ayant deux caches L1 et un cache L2. Ces routeurs permettent de créer un unique réseau physique. Ce réseau est virtualisé pour éviter les inter-blocages et fonctionne par passage de message pour éviter le trafic généré par la gestion matérielle de la cohérence. La cohérence mémoire doit donc ici être assurée par le logiciel.

Platform2012 Le CMP proposé par Melpignano *et al.* [2012] est un many-cœur orienté vers l'accélération matérielle et est présenté dans une configuration à 64 cœurs. Il est aussi organisé par tuiles mais celles-ci sont bien plus complexes que celles des trois CMP précédents. Une tuile contient 16 cœurs ayant chacun un cache L1 instruction de 16 Kio et partageant un unique cache L1 donnée de 256 Kio, évitant ainsi d'avoir à gérer la cohérence mémoire à ce niveau. Chaque tuile possède un DMA et un cœur supplémentaire chargé de la gestion de la tuile. Finalement, les quatre tuiles sont reliées entre elles par un réseau composé des routeurs proposés par Thonnart *et al.* [2010].

MPPA Le *Massively Parallel Processor Array* est un CMP développé par Kalray [De Dinechin *et al.*, 2013]. Il présente certaines similitudes avec Platform2012 : il est organisé en tuile de 16 cœurs possédant chacune un DMA et un cœur dédié à la gestion. Cependant, chaque cœur possède ici un cache L1 pour les instructions et un pour les données, chacun d'une capacité de 8 Kio. Il n'y a pas de cache L2 mais une

mémoire partagée distribuée, et la gestion de la cohérence est laissée au logiciel. Les tuiles sont reliées entres elles par deux NoC de type Torus, un réseau de type grille où les routeurs des bords opposés sont reliés entre eux.

Epiphany-V Finalement, le CMP proposé par Olofsson [2016] est composé de 1 024 cœurs. S'il montre une évolution vers un nombre de cœurs plus grand, il semble aussi mettre en avant un retour vers une architecture plus simple. Ainsi, chaque tuile contient un unique cœur sans cache, une fraction des 64 Mio de mémoire partagée distribuée et un routeur pour chacun de ses réseaux.

2.4.2 Les premiers NoC filaires

Dans cette sous-section, nous présentons différents NoC de type grille 2D et leurs particularités. Ils correspondent pour la plupart aux NoC utilisés dans les many-cœurs présentés dans la sous-section précédente.

SPIN et DSPIN Le réseau *Scalable Programmable Integrated Network* (SPIN) a été proposé par Guerrier et Greiner [2000] car, comme nous l'avons vu dans la section précédente, les bus ne passent pas à l'échelle. SPIN repose sur un réseau de routeurs par commutation de paquet permettant chacun de relier cinq éléments. La connexion avec chaque élément repose sur une FIFO de réception et une FIFO d'émission. Ces routeurs ne permettent pas uniquement de réaliser des topologies de type grille 2D, ils ont par exemple été utilisés pour mettre en place un Fat-Tree [Adriahantenaina *et al.*, 2003]. Finalement, Miro-Panades *et al.* [2006] ont proposé une amélioration appelée *Distributed SPIN* (DSPIN). Elle a été développée pour créer des *System on Chip* (SoC) découpés en tuiles et organisés en grille 2D. Elle permet aussi de répondre au problème de la distribution des horloges en autorisant le déphasage des horloges des tuiles entre elles. Pour ce faire, le routeur est éclaté sur les quatre bords de la tuile de façon à ce que les FIFO de deux routeurs adjacents puissent être bi-synchrones. Ainsi, les fils les plus longs ne sont plus entre deux tuiles mais à l'intérieur d'une tuile.

iMesh Le NoC utilisé dans le CMP Tile64 est décrit par Wentzlaff *et al.* [2007]. Ce réseau présente la particularité d'être composé de cinq grilles dont une fonctionne par commutation de circuit alors que les autres fonctionnent par commutation de paquet. Tout comme la version non-distribuée de SPIN, le transfert des messages entre les différents ports du routeur est géré par un crossbar central.

ANOC Le réseau *Asynchronous Network On Chip* (ANOC) a été proposé par Thonnart *et al.* [2010] et est utilisé dans le CMP Platform2012. Tout comme DSPIN, le but de ce NoC est de permettre l'implémentation de SoC respectant le paradigme *Globally Asynchronous Locally Synchronous* (GALS). Pour ce faire, le NoC repose sur

trois composants : les routeurs, les liaisons et les interfaces GALS. Les routeurs possèdent cinq ports d'entrées/sorties et supportent deux canaux virtuels. Les liaisons entre les routeurs utilisent des répétiteurs pour supporter l'allongement des fils et sont asynchrones. Finalement, les interfaces GALS permettent à la fois de gérer la différence entre l'horloge du NoC et l'horloge locale, mais aussi de générer cette dernière. Alors que DSPIN permet la gestion d'horloges locales déphasées, ANOC permet de gérer différents domaines d'horloges.

Mesh du SCC Le réseau utilisé par le SCC d'Intel a été proposé par Salihundam *et al.* [2011]. Il repose sur un réseau de routeurs utilisant des jetons pour favoriser alternativement certains chemins à travers le NoC. De plus, l'une des principales préoccupations ayant mené au développement de ce NoC est la consommation énergétique. Dans ce but, le SoC est découpé en plusieurs zones pouvant fonctionner à différentes tensions.

2.4.3 Limitations

Nous nous sommes volontairement limités aux NoC filaires de type grille 2D car ce sont les premiers à avoir été proposés et également ceux utilisés dans les many-cœurs commercialisés actuellement. Nous allons maintenant explorer les différentes limitations de ce type de NoC.

2.4.3.1 Bande passante : entre sous-utilisation et saturation

L'un des principaux avantages des NoC filaires de type grille 2D est que la bande passante augmente proportionnellement à la taille du réseau. Cependant, elle est uniformément répartie et la bande passante entre deux nœuds reste constante. Bien que la bande passante globale soit importante ce type de réseau peut donc saturer. Il existe autant de chemins que de couples de nœuds pouvant communiquer ensemble. Il est donc courant que plusieurs chemins partagent un certain nombre de routeurs. Ainsi le transfert d'un paquet par un routeur donné peut être retardé par le transfert d'autres paquets. Ce qui peut créer des points de contention plus ou moins importants en fonction du routage utilisé et de l'architecture des routeurs. Finalement, la bande passante de ce type de réseau est répartie de manière homogène. Or, en fonction du code exécuté, tous les nœuds du réseau n'ont pas nécessairement besoin de communiquer en même temps. Il peut ainsi y avoir des portions entières de la grille qui ne transmettent pas ou peu de données et dont la bande passante est sous-utilisée alors que d'autres portions sont saturées.

2.4.3.2 Latence

S'il est possible de réaliser des CMP de grande taille grâce aux NoC filaires de type grille 2D, cela a un coût en terme de latence. Le temps nécessaire pour envoyer une donnée d'un nœud à un autre dépend directement du nombre de routeurs traversés. Or, plus la taille du CMP augmente, plus le nombre moyen de routeurs séparant deux nœuds quelconques augmente. Une grille 2D présente donc deux inconvénients en terme de latence. D'une part, la latence maximale est de plus en plus grande. D'autre part, selon la taille du CMP, la latence peut varier de un à deux ordres de grandeur en fonction de la source et de la destination.

2.4.3.3 Peu adapté au broadcast et au multicast

Les communications circulant dans un CMP ne sont pas uniquement point à point. Un nœud peut avoir besoin d'envoyer une même donnée à plusieurs autres nœuds ou même à tous les nœuds du réseau. Ce cas de figure apparait notamment dans le maintien de la cohérence mémoire. De plus, la proportion de messages de multicast et de broadcast semble augmenter avec le nombre de cœurs [Abadal *et al.*, 2015]. Or pour réaliser un multicast dans une grille 2D, le message doit être émis autant de fois qu'il y a de destinataires. La réalisation d'un broadcast est différente puisqu'il est possible de n'émettre le message qu'une fois, celui-ci est cependant propagé par tous les routeurs. Ces deux types de communications consomment donc une part importante de la bande passante d'un NoC filaire de type grille 2D.

2.4.3.4 Manque de plasticité

Les communications peuvent présenter des variations spatiales et temporelles. Spatiales d'une part, les communications générées par une application peuvent en effet varier localement, à cause de variables partagées ou de barrières de synchronisation notamment. De plus, il est aussi possible d'exécuter en même temps dans différentes parties du CMP plusieurs applications générant chacune son propre profil de communication. Temporelles d'autre part, les communications générées par une même application peuvent varier au cours du temps, en fonction de l'exécution d'une section parallèle ou séquentielle du code par exemple. De plus un nœud donné peut voir ses besoins varier au cours du temps si l'application exécutée change. Or dans une grille 2D, la bande passante entre deux nœuds du réseau est fixe.

2.5 Conclusion et problématique

Dans ce chapitre, nous avons pu constater que les raisons de l'apparition des CMP se trouvaient à la fois du côté des applications exécutées et de celui du développement

même de ces architectures. De plus, ces deux raisons ont une influence mutuelle l'une sur l'autre. D'une part, les architectures CMP ont pu se développer car il y avait un besoin applicatif pour des architectures capables de plus de parallélisme de tâches. D'autre part, la possibilité d'améliorer plus facilement les performances parallèles que les performances séquentielles a orienté les modèles de programmation vers une recherche de parallélisme de tâche.

En analysant l'évolution des CMP nous avons pu constater que les premières architectures proposées reposaient sur les médiums d'interconnexion tels que les bus, les anneaux et les crossbars. Nous avons retenu le terme de multi-cœur pour parler de cette catégorie de CMP. Cependant ce type d'interconnexion ne passe pas à l'échelle et limite donc le nombre de cœurs intégrables sur une même puce.

C'est pour remédier à ce problème de passage à l'échelle que les premiers NoC ont été proposés. Ils sont constitués de routeurs reliés par des connexions filaires classiques et organisés sous forme de grille 2D. Nous avons retenu le terme de many-cœur pour parler de cette deuxième catégorie de CMP. Si cette première génération de NoC permet effectivement l'intégration d'un grand nombre de cœurs sur une même puce, ce n'est pas sans contre-partie.

Tout d'abord, la bande-passante totale offerte est importante mais ne reflète pas directement le nombre de communications que le réseau peut supporter. Ainsi plus la distance entre deux nœuds est importante, plus la bande passante consommée par un unique transfert est importante. Il serait donc intéressant de développer des NoC où toutes les communications auraient le même coût.

Ensuite, la latence moyenne des communications augmente en même temps que la taille de la grille 2D. De plus, la latence d'une communication varie fortement en fonction de la source et de la destination. Il serait donc intéressant de proposer des NoC où la latence des communications serait fixe quelque soit le chemin parcouru et indépendante du nombre de cœurs.

En outre, les grilles 2D sont peu adaptées aux communications de type multicast et broadcast. Dans ce type de NoC, le message doit être envoyé autant de fois qu'il y a de destinataires. Il serait donc intéressant de développer des NoC permettant de faciliter la diffusion de ce type de message.

Finalement, de part la technologie utilisée et leur topologie, les ressources de communications fournies par les grilles 2D sont réparties de manière homogène et statique. Elles ne peuvent pas s'adapter aux variations des communications en faisant varier spatialement et temporairement la bande passante. Il serait donc intéressant de proposer des NoC capables d'allouer dynamiquement leurs ressources de communication en fonction des besoins.

Les limitations de cette première génération de NoC peuvent être regroupées en deux catégories : statiques et dynamiques. D'une part, les contraintes en terme

de bande passante, de latence et de support du broacast sont principalement des contraintes statiques dépendant des caractéristiques intrinsèques du médium de communications. D'autre part, les contraintes en terme d'allocation des ressources de communications sont par nature des contraintes dynamiques s'appliquant au médium de communication. Le chapitre suivant présentera donc les différentes solutions explorées pour répondre à la double problématique suivante :

— Comment améliorer la latence, la bande passante et le support des messages de type multicast et broadcast dans un NoC ?

— Comment mettre en place des mécanismes de reconfiguration dynamique permettant à un NoC de s'adapter dynamiquement aux variations de trafic ?

Chapitre 3

État de l'art

Sommaire

3.1 Introduction

Dans le chapitre précédant, nous avons présenté les raisons matérielles et logicielles ayant favorisé le développement des *Chip Multiprocessor* (CMP). D'une part, la difficulté croissante à augmenter la fréquence des processeurs et à exploiter plus de parallélisme d'instruction. D'autre part, la présence de parallélisme de tâche dans le code exécuté. Nous avons ensuite présenté l'évolution de ces puces et nous les avons classées en deux catégories, les multi-cœurs et les many-cœurs. Les multi-cœurs correspondent aux CMP utilisant des médiums d'interconnexion classiques tels que les bus, les anneaux ou les crossbars. Cependant, le manque de scalabilité de ces derniers a poussé au développement des *Network on Chip* (NoC), des réseaux directement intégrés sur la puce et inspirés des systèmes à plus grande échelle. Les many-cœurs correspondent ainsi aux CMP basés sur ces NoC. Finalement, nous avons présenté les limitations de la première génération de NoC filaires. Tout d'abord, en terme de bande passante, même si elle augmente linéairement avec la taille du réseau, la portion consommée par une communication augmente avec la distance à parcourir. Deuxièmement, en terme de latence puisque celle-ci augmente avec la distance à parcourir et donc avec la taille du réseau. Troisièmement, ce type de réseau étant découpé en un ensemble de communications point à point, il est peu adapté aux communications de type broadcast, Finalement, l'adaptabilité de ce type de réseau aux besoins en communications est limité part sa structure homogène et le routage déterministe utilisé. Dans ce chapitre, nous présentons différentes solutions existantes adressant une ou plusieurs de ces limitations et analysons en quoi elles répondent ou non à la double problématique identifiée dans le chapitre précédant :

— Comment améliorer la latence, la bande passante et le support des messages de type multicast et broadcast dans un NoC ?

— Comment mettre en place des mécanismes de reconfiguration dynamique permettant à un NoC de s'adapter dynamiquement aux variations de trafic ?

Il existe plusieurs solutions pour améliorer la bande passante et la latence. Il est par exemple possible de changer la topologie du réseau tout en conservant un réseau 2D de routeurs filaires auquel on ajoute des connexions [Dally et Towles, 2001]. Une autre piste explorée pour remédier à ces limitations est d'utiliser un réseau 3D de routeurs filaires permettant de proposer de nouvelles topologies et de diminuer les distances au sein de la puce [Li *et al.*, 2006]. Finalement, il est possible d'utiliser de nouvelles technologies telles que l'optique [O'Connor et Gaffiot, 2004] ou la *Radio Fréquence* (RF) [Chang *et al.*, 2001]. Ces deux technologies permettent de faire circuler plus vite le vecteur physique de l'information et donc potentiellement de se passer de routeurs intermédiaires. La section 3.2 présente ces différentes solutions qui permettent d'améliorer les caractéristiques intrinsèques des NoC que sont la bande-passante et la latence.

Si les solutions précédentes apportent une amélioration de la bande passante et/ou de la latence, ce ne sont pas les seuls critères à prendre en compte pour évaluer les performances d'un réseau sur puce. Il faut aussi évaluer le support des communications de type multicast et broadcast utilisées, entre autre, dans les protocoles de cohérence de cache. Ce type de communication génère un trafic très important dans une grille 2D puisqu'un même message doit être réémis un grand nombre de fois. Soit par la source qui, dans le cas d'un multicast, émet autant de fois le message qu'il y a de destinataires. Soit par l'ensemble des routeurs du réseau qui, dans le cas d'un broadcast, se chargent de répliquer le message autant de fois que nécessaire pour qu'il se propage dans l'ensemble du réseau. La section 3.3 présente les solutions proposées pour permettre aux NoC de supporter plus efficacement le broadcast.

La dernière limitation des NoC filaires de première génération est la prise en charge de l'hétérogénéité spatiale et temporelle des communications au sein d'un CMP. Cette hétérogénéité est provoquée par l'exécution d'applications non régulières, ou tout simplement par l'exécution simultanée de plusieurs applications différentes. La section 3.4 présente les solutions proposées pour permettre à un réseau ayant des capacités données de s'adapter au mieux aux besoins en communication.

Finalement, la section 3.5 montre en quoi les solutions proposées répondent ou non aux limitations en terme de bande passante, de latence, de reconfigurabilité et de support du broadcast. Nous partons ensuite de cette analyse pour affiner la problématique identifiée au chapitre précédant et présenter les grands principes de la solution que nous proposons pour y répondre.

3.2 Améliorations de la bande-passante et de la latence

La première génération de NoC est basée sur des réseaux de routeurs filaires arrangés en grille 2D. Dans ce type de NoC, le temps nécessaire à l'envoi d'un message de la source à la destination dépend du nombre de routeurs à traverser, du temps nécessaire pour traverser un routeur et de la disponibilité du réseau. Les approches présentées dans cette section ont pour but de diminuer le nombre de routeurs traversés ou le temps de traversée de ces routeurs. Ces deux types d'amélioration permettent une augmentation de la quantité de données que le réseau peut transmettre durant un laps de temps donné.

La première piste d'amélioration est de conserver un réseau 2D de routeurs filaires et de modifier soit la topologie du réseau, soit l'architecture des routeurs. La modification de la topologie permet de diminuer le nombre de routeurs à traverser, alors que la modification de l'architecture des routeurs permet de diminuer le temps de traversée de chaque routeur. La sous-section 3.2.1 présente les solutions exploitant ce type d'approche.

L'autre piste d'amélioration est d'utiliser de nouvelles technologies pour dépasser les limitations des réseaux filaires classiques afin de répondre aux besoins croissants en bande passante et en latence. On peut regrouper ces solutions utilisant de nouvelles technologies en deux catégories. D'une part, celles qui utilisent des puces en 3D [Li *et al.*, 2006; Pavlidis et Friedman, 2007] pour diminuer les distances à parcourir, présentées dans la sous-section 3.2.2. D'autre part, celles qui utilisent des connexions optiques [O'Connor, 2004; Pan *et al.*, 2009] ou RF [Chang *et al.*, 2008; Ganguly *et al.*, 2011] pour augmenter la vitesse de propagation de l'information. Ces approches sont présentées respectivement dans les sous-sections 3.2.3 et 3.2.4.

3.2.1 Approches filaires 2D

Dans cette sous-section, nous allons présenter différentes approches utilisées pour améliorer la latence et la bande passante des NoC. Toutes les approches présentées ici se basent sur l'utilisation de réseaux filaires 2D.

3.2.1.1 Grille optimisée

Pour adresser les limitations des NoC filaires de type grille 2D, une première solution est d'essayer d'améliorer ces réseaux. Ainsi, en partant de la constatation que les routeurs sont plus ou moins sollicités en fonction de leur localisation dans la grille, Mishra *et al.* [2011] proposent d'utiliser des NoC hétérogènes. Pour ce faire, les routeurs situés au centre de la grille ainsi que sur ses deux diagonales ont une bande passante doublée et trois fois plus de canaux virtuels. Les autres routeurs ont quant à eux des caractéristiques inférieures à celles des routeurs utilisés dans une configuration homogène. Cela permet à la version hétérogène d'avoir une surface et une consommation inférieure à la version homogène tout en permettant un gain en performance de 10 % pour une grille 8×8. Cependant, même si cette solution améliore les performances par rapport à une grille 2D classique, ses caractéristiques évoluent de la même façon avec l'augmentation du nombre de cœurs.

3.2.1.2 Torus

Une autre piste d'amélioration des NoC basée sur une grille 2D est de relier le premier routeur au dernier dans chaque ligne et chaque colonne. La représentation dans l'espace de ce réseau donne un tore, ou torus en anglais, solide géométrique à l'origine du nom de ce type de réseau [Dally et Towles, 2001]. Ainsi, l'absence de bord fait du Torus un réseau plus homogène que la grille 2D et permettant de mieux répartir le trafic. Pour un CMP de $N \times N$ tuiles, cette topologie permet d'avoir une distance maximale de N contre $2N$ avec une grille 2D. Cependant, la latence reste la

même pour les communications entre deux routeurs initialement séparés par moins de $N/2$ routeurs dans une grille 2D. L'inconvénient de ce type de réseau est la longueur du fil utilisé pour connecter directement le premier élément d'une colonne ou d'une ligne au dernier. Ainsi le transfert d'un message sur ce type de fil prend plus de cycles qu'entre deux routeurs adjacents et non périphériques. De plus la présence de ces connexions entre le premier et le dernier élément de chaque ligne et chaque colonne augmente la complexité du routage.

3.2.1.3 Fat Tree

Cette topologie a été utilisée par Guerrier et Greiner [2000] pour le développement d'un des tous premiers NoC. Ce type de réseau suit une organisation en arbre équilibré où les tuiles sont les feuilles et les routeurs les nœuds. Le qualificatif de "Fat" vient du fait que chaque nœud possède une connexion avec son père capable de transmettre autant de bits que la somme des connexions avec ses fils. La Figure 3.1 représente le nombre de routeurs et de connexions nécessaires pour réaliser différentes tailles de CMP avec une grille 2D ou un Fat Tree. Les deux réseaux utilisent des routeurs à cinq ports d'entrées/sorties tels que décrit par Guerrier et Greiner [2000]. On constate alors que par rapport à une grille 2D, un Fat Tree utilise plus de fils à partir de 16 tuiles et plus de routeurs à partir de 256 tuiles. Si le Fat Tree permet de diminuer la latence des communications, il ne faut donc pas oublier que son coût matériel est plus élevé.

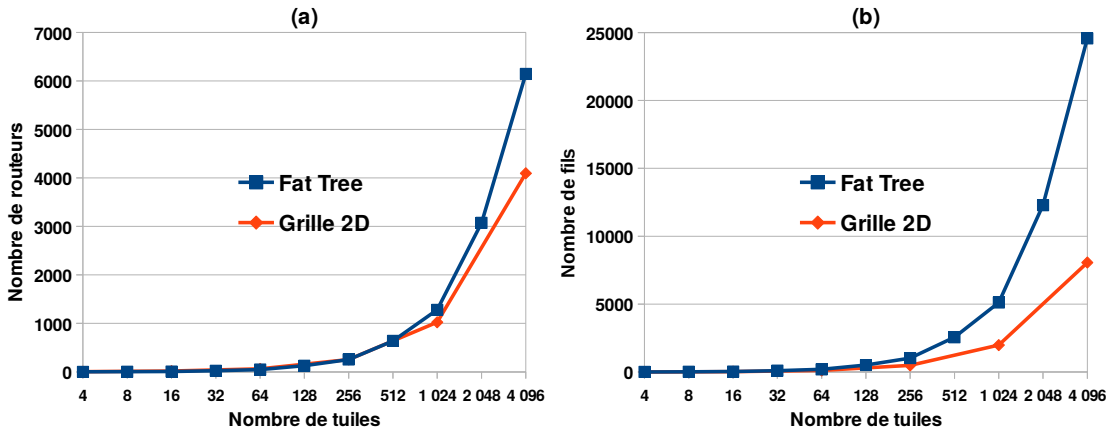

Figure 3.1 – Nombre de routeurs (a) et de connexions filaires (b) dans un CMP utilisant un Fat Tree ou une grille 2D

3.2.1.4 Spidergon

Ce type de réseau a été proposé par Coppola *et al.* [2004]. La base du Spidergon est un réseau en anneau possédant un nombre pair de nœuds. En plus des connexions

avec les deux plus proches voisins, chaque routeur est connecté au routeur diamétralement opposé. L'avantage d'un Spidergon par rapport à un mesh est sa simplicité et sa régularité. Chaque routeur possède exactement trois voisins alors qu'un mesh en possède entre deux et quatre. Grâce à sa structure symétrique, le nombre de routeurs à traverser pour atteindre le nœud le plus éloigné est le même quelque soit le nœud de départ. Ainsi, un Spidergon offre exactement les mêmes caractéristiques à tous les nœuds qui lui sont connectés. Le Spidergon offre donc l'avantage d'être plus simple à mettre en place et de répartir les ressources de communication de manière encore plus homogène que la grille 2D. Cependant, ce n'est pas une solution à retenir si le but est d'augmenter la bande passante et de diminuer la latence puisque dans ces deux domaines, il est moins performant qu'une grille 2D.

3.2.1.5 Crossbar

Ce type d'interconnexion a été écarté assez tôt des médiums utilisables pour les many-cœurs en raison de l'augmentation quadratique de sa complexité en fonction du nombre d'éléments connectés. Cependant, ses avantages en terme de latence ont poussé plusieurs équipes à continuer l'exploration de cette piste. Ainsi, Passas *et al.* [2012] proposent un crossbar permettant de relier 128 tuiles pour une surface de 16 mm^2 et une consommation de 7 W en utilisant une finesse de gravure de 90 nm. Cependant, l'augmentation du nombre d'éléments connectés entraine une augmentation de la latence. Il faut ainsi entre 11 et 15 cycles pour faire communiquer deux tuiles en fonction de leur placement sur la puce. Dans une grille 2D de même dimension, il faut entre 3 et 45 cycles. Si ce crossbar reste donc plus intéressant qu'une grille 2D pour faire communiquer les tuiles éloignées, ce n'est pas le cas des communications avec les tuiles proches. Selon la localité du trafic, ce crossbar peut donc offrir de meilleurs ou de moins bonnes performances qu'une grille 2D.

3.2.1.6 Flattened butterfly

Bien que le crossbar puisse être utilisé comme un réseau en-soi, c'est aussi la brique de base utilisée pour construire les routeurs des NoC de type grilles 2D, Fat Tree, Torus et Spidergon. Ces réseaux utilisent des routeurs ayant au maximum 5 ports d'entrées/sorties. Or, sans aller jusqu'au crossbar à 128 ports présenté au paragraphe précédent, il est possible de réaliser des routeurs possédant un plus grand nombre de ports. C'est en partant de cette constatation que Kim *et al.* [2007] ont proposé un NoC de type Flattened Butterfly. Dans ce réseau, les tuiles sont organisées comme avec une grille 2D mais chaque routeur est connecté à tous les routeurs situés sur la même ligne ou sur la même colonne. Ce réseau permet à n'importe quelle paire de tuiles de communiquer en utilisant au maximum trois routeurs diminuant ainsi

la latence par rapport à une grille 2D simple. Il ne faut cependant pas négliger le coût des connexions filaires plus longues et plus nombreuses que dans une grille 2D. D'une part, la longueur des fils empêche de transmettre un message entre deux routeurs adjacents en un seul cycle. D'autre part ces connexions filaires ont un coût important et augmentent les contraintes pesant sur le routage.

3.2.1.7 Limitations

Dans cette sous-section, nous avons présenté plusieurs solutions filaires reposant sur des modifications topologiques pour améliorer les performances par rapport à un réseau filaire de type grille 2D. Cependant, si certaines de ces solutions permettent d'augmenter la bande passante et/ou de diminuer la latence, c'est au prix d'une augmentation du coût matériel du réseau. Ainsi celui-ci peut s'avérer rédhibitoire pour des CMP de plusieurs centaines de cœurs pour le Fat Tree, le Crossbar et le Flattened butterfly. De plus, tout comme dans une grille 2D, la latence augmente inévitablement avec le nombre de nœuds du réseau. Cet inconvénient est notamment présent pour des réseaux tels que les grilles optimisées et les Torus. Finalement, les NoC de type Spidergon ont un coût matériel plus faible que les grilles 2D mais aussi de moins bonnes performances. Dans les sous-sections suivantes nous allons présenter les solutions reposant sur l'utilisation des nouvelles technologies telles que la 3D, l'optique et la RF.

3.2.2 La 3D

L'utilisation des NoC 3D ouvre la voie à plusieurs pistes d'amélioration des CMP. Tout d'abord, ils permettent de diminuer les distances au sein d'une puce. De plus, il est possible de produire séparément les différentes couches de manière à augmenter la surface totale de la puce par rapport à une puce 2D et/ou à en diminuer les coûts de production. Par ailleurs, certaines couches peuvent être dédiées à la mémoire de manière à améliorer sa bande passante. Finalement, les couches successives peuvent être réalisées en utilisant différentes technologies de manière à tirer parti des possibilités de chacune.

3.2.2.1 Technologies 3D

Il est possible de diviser les méthodes permettant de réaliser des puces 3D en deux catégories [Xie *et al.*, 2006]. La première consiste à fabriquer directement la puce en 3D, couche après couche. Si cette technologie permet une granularité de connexions 3D très fine et proportionnelle à la finesse de gravure utilisée, elle est aussi plus éloignée des procédés de fabrication actuels. La seconde consiste à fabriquer chaque

couche séparément puis à les empiler. Les couches peuvent être reliées grâce à différentes méthodes telles que les micro-bump ou les TSV. Les micro-bumps sont des plots de connexion entre deux couches successives. Les *Through Silicon Via* (TSV) sont des connexions traversant une couche pour la relier à la couche inférieure.

3.2.2.2 Réduction des distances

L'un des principaux leviers d'amélioration des performances offert par l'utilisation des NoC 3D est la diminution des distances au sein d'une puce. Pour créer un NoC 3D, la solution la plus simple est de partir d'une grille 2D et d'ajouter des connexions verticales à ses routeurs. Alors que les connexions Nord, Sud, Est et Ouest sont utilisées pour communiquer dans le plan, les connexions basses et hautes permettent de relier les différentes couches du circuit entre elles. Ainsi, pour un CMP de 4 096 tuiles, il est possible d'utiliser une grille 2D de 64×64 ou une grille 3D de $16 \times 16 \times 16$. Alors que le chemin le plus long dans la grille 2D est composé de $2\sqrt{N} - 1$ routeurs, soit 127, il n'en comporte que $3\sqrt[3]{N} - 2$, soit 46, dans le cas d'une grille 3D.

3.2.2.3 Optimisation des routeurs

Tout comme pour les grilles 2D, la latence des communications dans une grille 3D dépend du nombre de routeurs à traverser et du temps nécessaire à la traversée d'un routeur. Or, si l'utilisation de la 3D permet de diminuer la distance à parcourir, elle ne change pas directement le temps de traversée d'un routeur. C'est donc pour agir sur ce deuxième levier que Fernandes *et al.* [2015] proposent un routeur capable de traiter un paquet en un cycle contre trois pour les routeurs classiques tels que ceux proposés par Guerrier et Greiner [2000]. Cependant, si cette architecture est capable de traiter un *FLow control unIT* (FLIT) en un cycle, c'est au prix d'un chemin critique plus long de 63 %.

3.2.2.4 Optimisation des connexions verticales

Une deuxième piste d'amélioration des NoC 3D est d'optimiser les connexions verticales. Le traitement symétrique des connexions dans le plan et entre couches ne tire pas partie de la plus faible distance entre deux routeurs superposés qu'entre deux routeurs adjacents dans la même couche. C'est pour tirer partie de cette plus faible distance que Li *et al.* [2006] proposent d'utiliser un bus pour les connexions verticales. Cette solution permet à un routeur de communiquer en un cycle avec les routeurs adjacents du même plan ainsi qu'avec tous les routeurs des autres couches

situés à sa verticale. Cependant cette solution présente certaines limitations puisqu'il ne faut pas oublier le manque de scalabilité des bus. Ainsi, d'après Li *et al.* [2006] la solution qu'ils proposent est viable jusqu'à 9 couches.

3.2.2.5 Intégration multi-technologies

Pour finir, puisque les différentes couches d'une puce 3D peuvent être réalisées séparément, il est possible d'utiliser différentes technologies pour certaines de ces couches. Ainsi, dans la continuité de l'exploration de la 3D comme piste d'amélioration de la mémoire, Dong *et al.* [2008] proposent d'utiliser la 3D pour faciliter l'intégration de *Magnetic Random Access Memory* (MRAM) dans les puces. Par ailleurs, certaines approches mêlent la 3D à d'autres technologies utilisées dans les NoC : l'optique [Ye *et al.*, 2013] et la RF [More et Taskin, 2010]. Les approches peuvent alors être différentes, Ye *et al.* [2013] proposent d'utiliser la 3D pour réaliser une couche dédiée à l'optique alors que More et Taskin [2010] propose d'utiliser la RF pour relier les différentes couches.

3.2.2.6 Limitations

Si la 3D présente de nombreux avantages, elle comporte aussi certaines limitations. Ainsi, si le nombre de couches superposées est trop important, des problèmes de dissipation thermique peuvent apparaître. Or, c'est un point non négligeable puisque d'après Esmaeilzadeh *et al.* [2013], la dissipation thermique des architectures many-cœurs risque de devenir telle qu'il sera impossible d'utiliser simultanément la totalité des cœurs. Ce qui obligerait à éteindre alternativement les différents cœurs, on parle alors de *Dark Silicon*.

3.2.3 L'optique

L'exploration de solutions optiques pour réaliser des NoC part de la même constatation que la 3D. Le rapport entre la taille totale des puces et celle d'un transistor devient de plus en plus grand, ce qui rend les distances de plus en plus grandes à l'échelle du transistor. Alors que la 3D permet de diminuer les distances, l'optique permet aux données de parcourir une même distance plus rapidement.

3.2.3.1 Intégration de composants optiques

L'idée d'utiliser l'optique pour réaliser les connexions au sein d'une puce a été proposée pour la première fois par Goodman *et al.* [1984]. Cependant ce n'est qu'une

vingtaine d'années plus tard que l'idée à été de nouveau exploitée pour le développement des NoC, notamment par O'Connor et Gaffiot [2004]. D'un point de vue haut niveau, un NoC optique tel que schématisé par Bogaerts *et al.* [2013] est constitué à minima de trois types de composants :

— des émetteurs capables de convertir un signal électrique en signal optique ;
— un support physique permettant le transport du signal lumineux ;
— des récepteurs capables de convertir le signal optique en signal électrique.

Émetteur Ce composant est chargé de transformer le signal électrique en signal optique pour le transmettre au reste du CMP à travers le NoC optique. Il peut être réalisé de deux manières, soit en utilisant un laser par émetteur, soit en utilisant une unique source lumineuse dont le signal sera modulé par chaque émetteur. De plus, la source lumineuse peut être externe ou interne à la puce. Le point critique de l'utilisation d'une source lumineuse externe est la jonction avec la puce. Celui de l'utilisation d'une source lumineuse interne est de devoir utiliser d'autres matériaux semi-conducteurs que ceux utilisés traditionnellement dans les puces numériques. Finalement, il faut veiller à ce que le signal émis arrive à destination dans le cadre d'émetteurs multiples. Comme nous le verrons dans la section 3.4, plusieurs solutions sont envisageables telles que la modulation du signal par l'émetteur ou l'établissement d'un chemin entre la source et la destination sur le support physique.

Support physique Ce composant est chargé d'acheminer le signal optique au sein de la puce. Le moyen le plus simple de réaliser cette fonction efficacement est d'utiliser un guide d'ondes optique en silicium parcourant l'ensemble de la puce. Ce matériau présente l'avantage d'être à la fois un bon conducteur de lumière et facilement intégrable avec les technologies CMOS actuelles. En fonction des choix faits pour la réalisation des émetteurs et des récepteurs, il pourra avoir un rôle actif ou passif dans le routage du signal de la source vers la destination. Il est important de tenir compte de la sensibilité de ce composant aux variations thermiques pouvant influer sur son comportement face aux différentes longueurs d'onde transmises.

Récepteur Ce composant est chargé de transformer le signal optique reçu à travers le guide d'ondes en signal électrique. Ce photodétecteur peut être réalisé selon différentes méthodes et avec différents matériaux. Cependant les solutions utilisant d'autres matériaux semi-conducteurs que ceux utilisés dans les puces numériques classiques, semi-conducteurs de classe III-V ou Germanium, sont plus efficaces.

3.2.3.2 Intégration 3D

Comme évoqué dans la sous-section 3.2.2, l'optique et la 3D peuvent être utilisées conjointement. Ainsi, les solutions proposées sont souvent composées au minimum de deux couches, une utilisant une technologie CMOS standard pour les cœurs, caches, etc. et une dédiée aux composants optiques utilisant une technologie ad hoc.

Vantrease *et al.* [2008] proposent par exemple un NoC optique 3D en quatre couches. La couche supérieure contient des tuiles de quatre cœurs ainsi que leurs caches L1 pour faciliter la dissipation thermique. La deuxième couche contient le cache L2 ainsi que le contrôleur mémoire et le répertoire utilisé pour la cohérence mémoire. La troisième couche contient les circuits électroniques chargés de faire les conversions nécessaires entre le monde numérique des deux premières couches et le monde analogique de la dernière. Finalement, la couche inférieure contient les composants dédiés à l'optique.

3.2.3.3 NoC hybrides filaire/optique

L'utilisation de l'optique a été proposée pour remédier notamment aux problèmes de latence induits par une augmentation des distances à l'échelle du transistor. Cependant, la latence du réseau optique est composée du transfert du routeur filaire au routeur optique, de la communication optique elle même et du transfert du routeur optique au routeur filaire. Or le transfert d'un message d'un routeur filaire à un routeur optique correspond au temps de communication entre deux routeurs filaires. Il n'est donc pas intéressant d'utiliser le réseau optique pour les communications courtes distance, des solutions hybrides électrique/optique ont donc été proposées.

Pan *et al.* [2009] proposent ainsi une architecture hiérarchique utilisant un réseau filaire au niveau local et un réseau optique pour les communications globales. Le CMP est divisé en tuiles comprenant entre 16 et 32 cœurs en fonction du nombre total de cœurs. Les communications intra-tuile passent par un réseau filaire de type grille 2D dans laquelle chaque routeur est partagé par plusieurs cœurs, on parle de grille concentrée. Les communications inter-tuile passent par un réseau optique basé sur un modèle *Single Write Multiple Read* (SWMR). Côté émission, chaque tuile dispose de son propre canal d'émission qu'elle utilise quelque soit la tuile de destination des données. Côté réception, chaque tuile décode les données reçues sur l'ensemble des canaux et ce n'est qu'après analyse de l'en-tête d'un paquet qu'une tuile sait si ce paquet lui est destiné ou non.

3.2.3.4 NoC purement optique

Comme nous venons de le voir les NoC optiques sont souvent associés à des NoC filaires. Ce choix s'explique par un coût d'accès au réseau optique supérieur à la

durée des communications à faible distance via le réseau filaire.

Cependant, certain travaux proposent de diminuer ce coût d'accès pour pouvoir utiliser des solutions purement optiques. Ainsi, Kirman et Martínez [2010] proposent d'utiliser un Torus optique pour relier les différents nœuds du réseau. L'architecture proposée utilise un multiplexage fréquentiel, les paquets sous forme de signal optique sont routés vers leur destination uniquement en fonction de la longueur d'onde du signal. Cette solution évite de devoir décoder le signal optique à chaque routeur intermédiaire. Elle part de la constatation que c'est la conversion électrique/optique et optique/électrique qui augmente la latence des communications et la consommation dans un NoC optique. La contrepartie de cette solution est qu'un couple source/destination doit toujours utiliser une unique et même longueur d'onde pour communiquer. Cependant il est possible d'optimiser le réseau pour n'utiliser que $N/2$ longueurs d'onde au lieu de N^2. Cette diminution est possible car à chaque instant, le protocole réseau garantit qu'un nœud ne participe qu'à une seule communication. L'inconvénient de cette solution est qu'elle nécessite une phase de requête de l'émetteur et d'allocation du récepteur avant l'établissement d'une communication.

3.2.3.5 Répercutions sur les protocoles de cohérence mémoire

L'une des principales répercussions de l'utilisation de l'optique dans les NoC concerne les caches et leurs protocoles de cohérence. L'adoption des premiers NoC filaires s'est accompagnée d'une modification des protocoles de cohérence. Cette modification à été nécessaire pour prendre en compte l'augmentation du nombre de caches à maintenir cohérents mais aussi le manque d'adaptation de ces réseaux aux multicast et au broadcast. Or ce n'est pas nécessairement le cas des NoC optiques qui rendent possible l'utilisation de protocoles fortement basés sur ce type de communications. Par exemple, dans certains NoC optiques, les fréquences sont allouées aux émetteurs et tous les récepteurs doivent recevoir et analyser chaque message pour savoir s'il en est la destination. Chaque message étant reçu par l'ensemble des nœuds du réseau, il peut facilement être transformé en multicast ou en broadcast selon le traitement à la réception.

Kurian *et al.* [2010] proposent ainsi d'utiliser un NoC optique dans le but de développer de nouveaux protocoles de cohérence mémoire. Le CMP utilise un réseau filaire pour les communications à courte distance et le réseau optique pour les communications longue distance. Le réseau filaire relie des tuiles composées d'un cœur, d'un cache et d'un répertoire utilisé pour la cohérence. Ces tuiles sont regroupées par grappe de 16 pour se partager un point d'accès au réseau optique. Chaque nœud optique possède un modulateur lui permettant d'émettre sur la bande qui lui est attribuée et un démodulateur lui permettant de recevoir l'intégralité des bandes

utilisées dans le système. Kurian *et al.* [2010] proposent un protocole basé sur un répertoire où chaque ligne peut être dans un état parmi *Modified*, *Owned*, *Exclusive*, *Shared* et *Invalid* (MOESI). Ce répertoire est distribué de manière statique entre les différentes tuiles selon la plage d'adresse qui lui est attribuée. Le broadcast étant ici utilisé pour invalider une ligne de cache quand elle est partagée par plus d'un certain nombre de caches. Si la solution proposée tire partie des capacités de broadcast du réseau, elle reste basée sur un protocole assez classique et il serait intéressant d'aller plus loin dans l'adaptation du protocole de cohérence aux capacités du NoC.

3.2.3.6 Limitations

Les principales limitations des NoC optiques sont d'ordre technologiques. Tout d'abord, les composants optiques utilisés peuvent nécessiter l'utilisation de matériaux semi-conducteurs différents de ceux utilisés dans les technologies CMOS classiques pour être plus performants. Il en découle que l'utilisation des NoC optiques est fortement liée à celle de la 3D puisque cette technologie est souvent nécessaire pour relier la couche de calcul à la couche optique. Par ailleurs, il peut être nécessaire d'utiliser une source lumineuse externe en fonction de la technologie utilisée. Finalement, les composants optiques ont une sensibilité thermique importante dont il faut tenir compte.

3.2.4 La Radio Fréquence

L'utilisation de la RF dans les NoC a été proposée pour la première fois par Chang *et al.* [2001]. Les NoC RF offre une réponse similaire aux NoC optiques quant au problème de la réduction de la latence des communications. Ainsi, qu'elle soit RF ou optique, une onde électromagnétique offre un temps de propagation bien inférieur à celui d'un signal électrique. L'avantage de la RF vient de sa plus grande compatibilité avec les technologies CMOS actuelles. La faisabilité d'interconnexions RF a d'ailleurs déjà été prouvée pour une bande de 10 GHz centrée à 60 GHz par Wu *et al.* [2012] et pour des bandes de 16 GHz centrées à 31 GHz, 57 GHz et 120 GHz par Deb *et al.* [2012].

L'onde RF peut être transmise, à travers une ligne de transmission, par le biais d'une antenne ou plus récemment par ondes de surface. Les solutions utilisant des antennes offrent un plus grand degré de liberté dans les architectures réalisables mais introduisent une plus grande sensibilité aux interférences par rapport à l'utilisation d'un guide d'ondes ou d'ondes surfaciques. On peut par exemple réaliser des puces 2D ou 3D où les différentes parties ou couches sont reliées grâce aux antennes et peuvent donc être fabriquées séparément.

3.2.4.1 Guide d'ondes

Les premiers travaux ayant proposé d'utiliser la RF pour les communications au sein d'une puce reposaient sur l'utilisation d'un guide d'ondes [Chang *et al.*, 2000]. L'utilisation d'un support physique pour propager les ondes RF était alors préféré à une solution basée sur des antennes. Chang *et al.* [2001] ayant estimé une surface de l'ordre de 1 mm^2 pour chaque antenne, cette deuxième solution fut écartée dans un premier temps. Si depuis les avancées technologiques ont rendu l'utilisation d'antennes viable comme nous le verrons dans la sous-section suivante, des solutions à base de guide d'ondes sont toujours explorées.

Ito *et al.* [2008] proposent ainsi un NoC RF utilisant un guide d'ondes pour relier les différents composants d'un *System on Chip* (SoC). L'architecture utilise un guide d'ondes ouvert serpentant sur la puce auquel sont connectés les différents émetteurs-récepteurs. Les ondes circulent dans les deux sens sur le guide et une onde émise par un nœud peut être reçue par plusieurs autres nœuds, ouvrant ainsi des possibilités de multicast et de broadcast dont nous parlerons dans la section 3.3.

Plus récemment, Wu *et al.* [2012] ont proposé un guide d'ondes possédant plusieurs interrupteurs de manière à pouvoir isoler des parties du circuit entre elles. Cette solution est un premier pas vers la reconfiguration dynamique des NoC pouvant être mise en place à faible coût. Il existe cependant des solutions offrant un plus grand degré de liberté comme nous le verrons dans la section 3.4. Chang *et al.* [2008] proposent par exemple de garder une unique ligne de transmission tout en utilisant plusieurs émetteurs-récepteurs par nœud fonctionnant à différentes fréquences. Dans cette architecture, le réseau RF est superposé à un réseau filaire de type grille 2D pour diminuer la latence globale en fournissant des raccourcis pour les communications longue distance.

3.2.4.2 Antenne

Les avancées technologiques ont permis une diminution de la surface des antennes et leur utilisation dans les NoC RF. Cette diminution de la surface a notamment été permise par l'augmentation de la fréquence qui est inversement proportionnelle à la longueur d'onde et donc à la surface de l'antenne.

Lee *et al.* [2009] proposent un CMP ayant comme base une grille 2D filaire sur laquelle vient se superposer le NoC RF. Pour ce faire, le CMP est découpé en 16 grappes possédant chacune un routeur RF central. Une grappe est composée de 16 tuiles, les 4 tuiles centrales contiennent chacune 4 caches L2 alors que les 12 tuiles en périphérie contiennent chacune 4 cœurs. Dans cette architecture, chaque routeur RF possède un émetteur pour émettre sur sa fréquence propre et autant de récepteurs que nécessaire pour recevoir chaque fréquence propre des autres routeurs RF.

De plus chaque émetteur et chaque récepteur possède sa propre antenne intégrée dans une couche supplémentaire au dessus de la dernière couche de la technologie CMOS standard. Pour limiter la surface occupée, les auteurs proposent donc que chaque routeur ne soit capable de communiquer qu'avec un sous-ensemble du réseau. Il peut alors être nécessaire de passer par des routeurs intermédiaires pour relier deux points du réseau ce qui diminue les gains en latence.

Ganguly *et al.* [2011] proposent eux aussi une approche hiérarchique dans laquelle ils optimisent l'accès à la RF. Le CMP est divisé en grappes de 4×4 cœurs où chacun d'eux dispose d'un accès direct en un cycle au routeur RF situé au centre de la grappe sans avoir besoin de passer par la grille 2D filaire. Les routeurs RF utilisent des antennes en nano-tubes de carbone pour minimiser la surface utilisée. Dans cette solution chaque fréquence RF est partagée par un unique couple de grappes, ses performances reposent donc sur l'optimalité du choix de ces couples

3.2.4.3 Ondes de surface

Une troisième solution pour transmettre les ondes RF a été explorée récemment. Il s'agit des ondes de surface, une solution à mi-chemin entre le guide d'ondes et les antennes. Alors que la propagation le long d'un guide d'ondes représente une solution à une dimension et que l'utilisation des antennes entraine une propagation dans les trois dimensions, les ondes de surface se propagent en deux dimensions dans un plan.

Karkar *et al.* [2016] proposent ainsi un NoC exploitant ces ondes de surface. Le CMP présenté utilise un réseau filaire de type grille 2D auquel est ajouté le NoC RF utilisant des ondes de surface. Karkar *et al.* [2016] ont fait le choix des ondes de surface pour répondre au problème des communications de type multicast et broadcast que nous verrons plus en détail dans la section 3.3. Ainsi, le NoC RF est utilisé pour ces deux types de communications alors que le réseau filaire est utilisé pour toutes les autres. Dans l'architecture proposée, tous les nœuds sont capables de recevoir des données mais seuls quelques uns peuvent en émettre. Ce choix présente l'inconvénient de faire circuler un message de multicast ou de broadcast sur le réseau filaire avant de le faire passer sur le NoC RF alors que c'est le seul but de ce réseau. De plus, il est dommage de ne pas utiliser le réseau RF pour les communications longues distances plutôt que le réseau filaire alors que c'est l'un des points faibles de ce dernier.

3.2.4.4 Intégration 3D

Les NoC RF présentent l'avantage de pouvoir être réalisés intégralement avec les technologies CMOS standards 2D. Wu *et al.* [2012] ont par exemple réalisé un prototype de NoC RF en 65 nm. Il peut tout de même être intéressant d'utiliser une

couche dédiée à la RF et les autres couches aux calculs pour que chacune puisse bénéficier de la technologie la plus optimale. Par ailleurs, le problème peut être inversé puisque la RF sans fil peut être elle même une solution pour fabriquer des puces 3D.

Lee *et al.* [2011] proposent par exemple de créer un CMP 3D où les différentes couches seraient reliées par couplage inductif. L'intérêt de la méthode est de pouvoir fabriquer et tester chaque couche séparément ce qui ouvre la voie à des architectures modulables. Matsutani *et al.* [2013] proposent d'utiliser cette méthode pour pouvoir remplacer une couche défaillante ou pour pouvoir faire évoluer le système en y ajoutant des couches supplémentaires.

3.2.4.5 Répercutions sur les protocoles de cohérence mémoire

Tout comme les NoC optiques, les NoC RF peuvent être utilisés pour diminuer le coût du maintien de la cohérence mémoire. Ainsi, les NoC RF peuvent mettre en place une allocation des fréquences du côté émetteur, ce qui oblige tous les récepteurs à recevoir l'ensemble des messages. De ce fait, chaque communication point à point peut facilement être transformée en multicast ou broadcast. Il est ainsi possible d'utiliser ces capacités de multicast et de broadcast intrinsèques à certains NoC RF pour mettre en place des protocoles de cohérence différents de ceux employés dans les NoC filaires.

Hu *et al.* [2015] proposent ainsi un CMP répartissant de manière optimale les communications entre sa grille 2D filaire et son NoC RF basé sur un guide d'ondes. Le CMP proposé comporte 64 tuiles chacune composée d'un cœur, de ses deux caches L1, d'un cache L2 partagé, du répertoire utilisé pour le maintien de la cohérence et d'un routeur. Les routeurs permettent à chacune de ces tuiles de communiquer via le réseau filaire et via le réseau RF. Le réseau RF est utilisé pour les requêtes de lectures, plus sensibles à la latence, et pour les communications longue distance. Cette répartition des communications n'est pas seulement bénéfique pour celles générées par le protocole de cohérence mais pour l'ensemble des communications au sein du CMP. Cependant, il est possible d'aller plus loin dans le développement conjoint du NoC et du protocole de cohérence comme nous le verrons dans la section 3.3.

3.2.4.6 Limitations

Les principales contraintes de la RF sont liées aux composants utilisés. Il faut ainsi veiller à minimiser la surface et la consommation des composants RF. De plus, il faut se prémunir contre les pertes liées aux interférences, principalement lors de l'utilisation de NoC RF sans-fil. Pour cela, il faut soit diminuer l'occurrence d'erreurs, soit mettre en place des mécanismes de détection et de correction.

3.2.5 Synthèse

Dans cette section, nous avons présenté les solutions explorées pour améliorer la bande passante et la latence par rapport à un NoC filaire de type grille 2D. Tout d'abord, les approches conservant un réseau filaire 2D peuvent permettre une amélioration de la bande passante et/ou de la latence. Cependant, le coût matériel de ces solution et/ou l'augmentation inéluctable de la latence avec le nombre de nœuds limite leur passage à l'échelle. Trois solutions basées sur l'utilisation de nouvelles technologies ont donc été proposées pour repousser ces limites.

Premièrement, l'utilisation de la 3D permet de diminuer les distances au sein d'une puce. Ainsi, pour un réseau reliant N nœuds, la distance maximale entre deux nœuds est de l'ordre de $\sqrt[2]{N}$ dans un réseau 2D alors qu'elle n'est que de l'ordre de $\sqrt[3]{N}$ dans un réseau 3D. Cependant, l'augmentation du nombre de couches augmente les contraintes de dissipation thermique.

Deuxièmement, l'utilisation de l'optique permet non pas de diminuer les distances, mais de diminuer le temps nécessaire pour les parcourir. Cette diminution est rendue possible par l'utilisation d'une onde lumineuse comme support de l'information à la place d'un signal électrique. Les principales limitations de cette technologie concernent la source lumineuse et la sensibilité thermique des composants. Ainsi, les solutions proches des technologies actuelles nécessitent l'utilisation d'une source lumineuse externe alors que les technologies pouvant s'en passer ne sont pas encore intégrable actuellement. Finalement, il est possible d'améliorer les performances d'un réseau optique en utilisant deux couches dédiées, une pour les composants optiques et l'autre pour les composants électroniques. Cependant, cette solution implique d'utiliser conjointement une technologie 3D.

Troisièmement, l'utilisation de la RF permet, tout comme l'optique, de diminuer le temps nécessaire pour parcourir une distance donnée. De plus, tout comme l'optique, la RF peut bénéficier avantageusement de l'utilisation d'une couche dédiée aux composants RF. Cependant, un réseau sur puce RF présente l'avantage d'être réalisable grâce aux technologies 2D actuelles.

Finalement, si ces différentes solutions permettent d'améliorer les caractéristiques d'un NoC en termes de latence et de bande passante, ce ne sont pas les seules caractéristiques à prendre en compte. Ainsi, nous verrons dans les sections suivantes comment ces différentes technologies peuvent être utilisées pour créer des réseaux reconfigurables dynamiquement et supportant efficacement les messages de type multicast et broadcast.

3.3 Support du multicast et du broadcast

Dans la section précédente, nous avons vu les solutions apportées pour améliorer la bande passante et la latence. Ces solutions peuvent consister en une amélioration des caractéristiques intrinsèques d'un NoC ou dans une amélioration de l'utilisation des ressources de communications qu'il fournit. Cependant, nous n'avons pour l'instant pas distingué les communications point à point des communications de type multicast ou broadcast. Le coût de ces dernières peut être grandement supérieur à celui des communications point à point.

Or les communications de type multicast et broadcast sont nécessaires dans les deux paradigmes de programmation multi-tâches, que ce soit par threads ou par passage de message. De plus, avec l'augmentation du nombre de cœurs, les broadcasts doivent atteindre un nombre de nœuds de plus en plus grand. C'est aussi le cas dans une moindre mesure pour les multicasts : plus l'architecture comprend de cœurs, plus le nombre moyen de destinations de ce type de communications tend à augmenter. En outre, l'augmentation du nombre de cœurs peut changer la proportion de broadcast et de multicast dans les communications. C'est par exemple le cas quand un CMP utilise un protocole de cohérence mémoire par répertoire. Si le nombre de caches partageant une donnée dépasse un certain seuil, les mises à jour sont faites par broadcast et non plus par multicast.

Pour finir, limiter l'émission de messages de multicast et de broadcast limite le développement de nouveaux protocoles de cohérence ou de nouveaux modèles de programmation. Il est donc intéressant de développer des NoC capables de limiter le surcoût des communications de type multicast ou broadcast par rapport aux communications point à point. Dans cette section nous allons présenter les différentes méthodes proposées pour intégrer le support du multicast et du broadcast aux NoC. Nous présenterons d'abord les méthodes utilisées dans les NoC filaires, puis celles utilisées dans les NoC optiques et finalement celles utilisées dans les NoC RF. Dans les réseaux filaires, les méthodes utilisées reposent principalement sur la modification de la topologie. Dans les réseaux optiques et RF, c'est sur les caractéristiques de ces deux technologies que reposent les méthodes proposées.

3.3.1 NoC filaires

Jerger *et al.* [2008] proposent un CMP utilisant une grille 2D filaire possédant un grand nombre de canaux virtuels. Ce sont ces canaux virtuels qui sont utilisés pour créer les arbres nécessaires à la propagation des messages de multicast. La création de ces chemins se fait progressivement lors de la diffusion du premier multicast ayant une source et un jeu de destination donné. Cette solution semble difficilement

utilisable dans le cadre de CMP à plusieurs centaines de cœurs. Le nombre de canaux virtuels, et donc de buffers par routeur ayant alors un coût trop important.

Manevich *et al.* [2009] propose un CMP utilisant une grille 2D filaire et un bus chargé des communications de types multicast et broadcast. Ce bus est en fait un réseau de routeurs filaires organisés en arbre binaire et fonctionnant par commutation de circuits. Pour un broadcast, une fois le bus attribué à un nœud, celui-ci envoie le message à émettre jusqu'à la racine de l'arbre et à partir de là le message redescend dans l'ensemble du réseau. Le multicast est un peu plus coûteux puisque la racine doit reconfigurer les différents noeuds du réseau pour supprimer certains chemins avant de faire redescendre le message de multicast. Si cette approche semble efficace pour les CMP de taille raisonnable, l'utilisation d'un arbre binaire la rend peu scalable. L'augmentation du nombre de routeurs nécessaires à la réalisation de l'arbre augmentera nécessairement la surface du réseau ainsi que le nombre de cycles pour réaliser ce type de transaction.

3.3.2 NoC optiques

Morris *et al.* [2014] proposent d'utiliser un NoC optique pour pouvoir continuer à utiliser un protocole de cohérence par scrutation grâce aux capacités de broadcast du réseau. Le CMP de 4×4 tuiles utilise un réseau optique en deux parties. Une première partie est composée de 4 guides d'ondes traversant chacun une colonne. Ces guides d'ondes partent tous du milieu du bord inférieur de la puce et rejoignent le milieu du bord supérieur. La deuxième partie est composée d'un arbre binaire de guides d'ondes dont la racine est le milieu du bord supérieur de la puce où tous les guides d'ondes de la première partie se rejoignent. Les nœuds du réseau émettent les messages de broadcast sur la première partie du réseau en respectant un protocole d'allocation par jeton alors que la deuxième partie du réseau se charge de transmettre effectivement le message à l'ensemble du réseau. Si cette solution est intéressante pour des CMP ayant un nombre limité de cœurs, l'utilisation d'un accès optique par cœur empêche son utilisation pour de plus grosses configurations.

Kurian *et al.* [2010] ont ainsi proposé un CMP utilisant un NoC optique superposé à une grille 2D filaire. Le CMP est découpé en grappes de 16 cœurs possédant chacune un hub optique leur permettant d'être reliées entre elles par le réseau optique. L'accès à ce hub optique se fait par l'intermédiaire de la grille 2D filaire en émission. En réception, les données circulent sur un troisième réseau organisé en arbre qui propage les données depuis le hub vers tous les cœurs Les données circulant sur le NoC optique peuvent être lues de manière non destructive par tous les nœuds ce qui fait de chaque communication un broadcast potentiel. Cependant, les communications circulant sur le réseau optique peuvent être transformées en communications point à point en positionnant un bit particulier à 1 dans le paquet transmis. Le paquet est

alors conservé uniquement par le hub optique de destination qui le propage ensuite à l'ensemble de ses cœurs. Chaque cœur se charge ensuite de jeter le paquet s'il n'en est pas la destination. L'architecture proposée apporte une solution efficace au support des communications de type broadcast. Elle manque cependant de reconfigurabilité puisqu'elle utilise une configuration de type SWMR où une fréquence donnée ne peut être utilisée que par un unique nœud en émission et reçue par l'ensemble du réseau.

3.3.3 NoC RF

Tout comme pour la mise en place de la reconfiguration dynamique, le support des communications de multicast et de broadcast dans les NoC RF et optiques présentent certaines similitudes. Ainsi les NoC RF utilisant un guide d'ondes sont assez proches des NoC optiques. Ils utilisent un support physique partagé par tous les nœuds du réseau, permettant d'envoyer indifféremment un signal à destination d'un unique nœud ou à l'ensemble des nœuds du système.

Dai *et al.* [2015] proposent d'utiliser un NoC RF utilisant des antennes. Le CMP proposé comporte 16 grappes contenant chacune 4 cœurs. Les cœurs au sein d'une grappe sont reliés à un routeur RF central par un réseau filaire en étoile. Le support du multicast et du broadcast se fait en créant des groupes de routeurs RF. Au sein de ces groupes, un unique routeur est autorisé à émettre alors que tous les autres sont en mode réception. L'utilisation d'un NoC RF pour les communications intergrappe présente des avantages en terme de latence, de support du multicast ou de reconfigurabilité. Cependant, il est dommage de ne pas tirer parti des performances du réseau filaire pour les communications locales en limitant son rayon d'action aux quatre cœurs constituant une grappe.

Karkar *et al.* [2016] proposent un NoC RF utilisant une propagation par ondes de surface. Dans cette architecture, le NoC RF est utilisé en complément de la grille 2D filaire pour transmettre les communications de multicast et de broadcast. Au sein du NoC RF tous les nœuds sont capables de recevoir des données, mais seuls quelques uns peuvent en émettre. Ce choix permet d'économiser des ressources matérielles des émetteurs mais présente l'inconvénient de faire circuler un message de multicast ou de broadcast sur le réseau filaire avant de le faire passer sur le NoC RF. De plus, le NoC RF proposé se limite à la diffusion des messages de multicast et de broadcast alors qu'il pourrait être utilisé aussi pour les communications longue distance pour lesquelles une grille 2D filaire est inefficace.

3.3.4 Synthèse

Dans cette section, nous avons présenté les solutions proposées pour améliorer le support des messages de type multicast et broadcast dans les NoC. Les solutions basées sur les NoC filaires présentées fonctionnent par commutation de circuits. Il faut tout d'abord construire le chemin vers l'ensemble des destinataires avant d'envoyer le message, cette contrainte limite le passage à l'échelle de ces solutions. Les solutions basées sur les NoC optiques et RF se rapprochent plus d'un fonctionnent par commutation de paquets. Elles reposent principalement sur les caractéristiques intrinsèques de ces deux technologies qui peuvent permettre de partager un même support physique entre tous les nœuds du réseau quelque soit sa taille. Les technologies optiques et RF offrent donc une meilleure base pour construire des NoC supportant efficacement les messages de type multicast et broadcast.

3.4 Reconfiguration dynamique

L'utilisation des nouvelles technologies apporte des gains en termes de latence et de bande passante. Cependant la demande en communication n'est pas forcément uniforme et constante au sein d'un CMP. En fonction du code s'exécutant, les différents cœurs n'ont pas les mêmes besoins en communication, on parle alors d'hétérogénéité spatiale. De plus, ces besoins peuvent varier dans le temps si les traitements effectués par les différentes tâches changent ou si celles-ci sont déplacées sur d'autres cœurs d'exécution, on parle alors d'hétérogénéité temporelle.

La prise en charge de cette hétérogénéité des communications est une des limitations des NoC filaires classiques utilisant des communications point à point telles que les grilles 2D. Même en augmentant la priorité de certaines communications par rapport à d'autres, il est difficile de s'affranchir de la nécessité de passer par des routeurs intermédiaires. De plus, la bande passante des portions inoccupées du réseau est difficilement ré-attribuable aux portions qui en auraient besoin. Quant à la 3D, si elle permet bien d'améliorer la latence et la bande passante, elle ne permet pas de créer des ressources de communication faciles à allouer.

De leur côté, l'optique et la RF permettent d'établir des canaux de communication directs entre source et destination sans intermédiaire, quelque soit leur emplacement sur la puce. En superposant un réseau optique ou RF à un réseau classique, on crée ainsi des raccourcis entres différents points de la puce. Pour contrôler l'utilisation de ces raccourcis, l'approche la plus simple est de les répartir de manière homogène sur la puce, de façon à diminuer la latence des communications entre points distants. Les raccourcis étant fixés par l'implémentation de la puce, leur nombre et leur répartition ne peuvent varier. Ils ne peuvent donc être utilisés que par un jeu de nœuds prédéfinis lors de la conception du circuit.

Cependant ces raccourcis correspondent aux ressources physiques que sont les antennes, les guides d'ondes, mais aussi et surtout les bandes de fréquence. Une approche plus évoluée va ainsi consister à diviser la bande passante en plusieurs canaux correspondant à différentes fréquences. Ces canaux sont mis en commun et sont alloués en fonction des besoins. Les nœuds doivent alors exprimer leurs besoins et le NoC doit être capable d'allouer les canaux en tenant compte de ceux-ci. La première chose à fixer est la façon dont les nœuds communiquent leurs besoins, via le réseau optique/RF [Xiao *et al.*, 2013] ou via le réseau électrique classique [Le Beux *et al.*, 2014]. Il faut ensuite utiliser un algorithme d'allocation, centralisé ou réparti, et choisir la granularité temporelle de ré-allocation des ressources de communication. Le développement de cet algorithme est fortement lié à l'architecture matérielle et à la flexibilité que l'on souhaite apporter aux canaux de communication.

3.4.1 NoC filaires

Mishra *et al.* [2013] sont partis de la constatation qu'il était possible de diviser les applications en deux catégories. Celles qui ont besoin de beaucoup de bande passante et celles qui ont besoin de pouvoir communiquer avec une faible latence. Ils proposent donc de mettre en place un CMP utilisant deux NoC de type grille 2D en parallèle. Le premier, optimisé pour la latence, fonctionne à une fréquence $3\times$ supérieure au reste du système. Le deuxième, optimisé pour la bande passante, utilise des connexions de 256 bits entre chaque routeur contre 64 bits pour le premier. Chaque cœur est relié à un routeur de chacun des deux réseaux par une interface chargée d'analyser le trafic et de choisir quel réseau utiliser. De plus, au sein d'un même réseau, chaque message se voit attribuer une priorité en fonction de l'application émettrice. Le choix du réseau à utiliser et de la priorité attribuée dépend de deux caractéristiques de l'application émettrice : la durée d'une phase de communication et la quantité de données échangées durant cette phase. Plus les phases de communication sont longues et le volume de données important, plus il est probable d'utiliser le réseau optimisé pour la bande passante et d'avoir une faible priorité. À l'inverse, plus les phases de communication sont courtes et le volume de données faible, plus il est probable d'utiliser le réseau optimisé pour la latence et d'y bénéficier d'une priorité élevée.

Les travaux de Mishra *et al.* [2013] montrent qu'il est plus intéressant d'utiliser plusieurs réseaux pour une bande passante donnée. Ainsi, la consommation d'un réseau d'une largeur de 256 bits est supérieure de 40 % à celle de deux réseaux de 128 bits chacun. Cependant, en terme de speedup, les gains semblent mitigés, notamment à cause du manque de plasticité du support physique des communications. Une configuration statique utilisant un réseau de 320 bits fait mieux qu'une configuration dynamique utilisant deux réseaux de 256 bits et 64 bits (sans augmen-

tation de fréquence) et à peine moins bien que celle où la fréquence du réseau de 64 bits est 3× supérieure.

3.4.2 NoC optiques

Dans sa version la plus simple, chaque nœud d'un NoC optique possède des composants lui permettant l'émission et la réception d'une ou plusieurs longueurs d'onde sur un ou plusieurs guides d'ondes. Il est donc nécessaire de mettre en place un mécanisme d'allocation des ressources que constituent les longueurs d'onde et les guides d'ondes. Pour ce faire, les deux premières approches utilisées sont *Multiple Write Single Read* (MWSR) et *Single Write Multiple Read* (SWMR).

Vantrease *et al.* [2009] proposent d'utiliser un mécanisme MWSR basé sur l'utilisation de jetons. Le NoC optique dispose de plusieurs guides d'ondes pouvant chacun faire circuler plusieurs longueurs d'onde. Un canal de communication correspond ici à une longueur d'onde donnée sur un guide d'ondes particulier. Puisque la méthode utilisée est de type MWSR, chaque canal ne permet d'envoyer des données que vers un unique nœud de destination. Pour pouvoir émettre, un nœud source doit intercepter le jeton correspondant à un canal du nœud de destination. Cette interception est rendue possible par la lecture destructrice réalisée par le photodétecteur. Le nœud ré-émet ensuite le jeton quand il n'a plus de données à émettre ou après l'émission d'un certain nombre de paquets pour ne pas générer de famine sur le réseau.

Pan *et al.* [2009] proposent d'utiliser un mécanisme SWMR. Chaque canal est donc utilisé ici par un unique nœud source pour communiquer avec le nœud destination de son choix. L'avantage du SWMR par rapport au MWSR est qu'il n'y a pas besoin de mécanisme d'allocation des canaux puisqu'un seul nœud peut émettre sur chaque canal. Cependant, le SWMR présente l'inconvénient de forcer chaque récepteur à décoder les données reçues sur l'ensemble des canaux pour identifier celles qui lui sont destinées.

Si les configuration SWMR et MWSR présentent des caractéristiques différentes, elles partagent la même limitation. Les canaux de communication ne peuvent pas être alloués librement pour n'importe quelle communication puisqu'un canal appartient soit à un émetteur soit à un récepteur donné. Pour un plus grand degré de liberté d'allocation dynamique il est donc nécessaire de disposer d'une configuration Multiple Write Multiple Read (MWMR). Le Beux *et al.* [2014] proposent ainsi un NoC optique où chaque nœud du réseau possède les composants lui permettant d'émettre et de recevoir chaque longueur d'onde de chaque guide d'ondes. Cette architecture repose sur un mécanisme de réservation des canaux de communications utilisant le réseau filaire. La dépendance du réseau optique vis-à-vis du réseau

filaire pour l'allocation des canaux de communication pourrait limiter les gains permis par l'utilisation de l'optique. Ainsi l'augmentation de la latence du réseau filaire entrainée par l'augmentation du nombre de cœurs pourrait diminuer la réactivité des mécanismes d'allocation et donc leur efficacité.

3.4.3 NoC RF

Les configurations MWSR, SWMR et MWMR utilisées en optique peuvent tout aussi bien l'être en RF. Le NoC RF proposé par Lee *et al.* [2009] repose par exemple sur une configuration SWMR puisque chaque nœud possède un émetteur pour sa fréquence dédiée et un récepteur par fréquence utilisée dans le réseau.

Xiao *et al.* [2013] proposent quand à eux un NoC RF reposant sur une configuration MWMR. Ce NoC repose sur deux réseaux utilisant chacun son propre guide d'ondes, un pour les données et un pour transmettre les besoins en communications. Le réseau de donnée utilise un guide d'ondes ouvert serpentant sur la puce par lequel chaque nœud du réseau est capable d'émettre et de recevoir des données suivant un fonctionnement classique. Le réseau transmettant les besoins utilise un guide d'ondes en boucle que chaque nœud à la possibilité de couper en ouvrant son interrupteur. La première phase du mécanisme d'allocation consiste à faire circuler à travers tous les nœuds du NoC un train d'arbitrage constitué d'un champ de bit. Chaque nœud se voit attribué une région spécifique dans ce train pour y indiquer s'il est capable de recevoir un paquet et s'il veut en émettre un, dans ce cas, il indique aussi le destinataire. Dans un deuxième temps, ce même train d'allocation effectue un deuxième tour durant lequel il s'arrête dans chaque nœud. Le nœud décide alors d'émettre ou non en fonction de ses besoins, de ceux des autres nœuds et de la disponibilité de la source. Si ce mécanisme permet une allocation équitable des ressources de communication, sa complexité risque de diminuer les performances en terme de latence. Ainsi la durée entre le moment où un nœud exprime un besoin d'émettre et celui où il est effectivement autorisé à émettre peut être assez important, et augmente avec le nombre de nœuds constituant le réseau.

Mansoor *et al.* [2016] proposent un NoC RF permettant d'améliorer la réactivité de l'allocation dynamique des ressources de communication. Pour ce faire, le réseau utilise un mécanisme de prédiction basé sur l'historique des besoins de chaque nœud. L'allocation des ressources est réalisée en faisant circuler un jeton et c'est lorsqu'un nœud acquiert ce jeton qu'il décide du temps durant lequel il va le garder, tout en respectant une limite haute. Si ses besoins sont en train d'augmenter, il gardera le jeton plus longtemps et inversement s'il détecte que ses besoins sont en train de diminuer. Cette démarche apporte une réelle adaptation du réseau aux besoins en communication. Il serait cependant possible de l'améliorer en permettant

à chaque nœud de tenir compte non seulement de ses propres besoins mais aussi directement de ceux du reste du réseau.

3.4.4 Synthèse

Dans cette section, nous avons présenté les solutions proposées pour allouer dynamiquement les ressources de communication dans un NoC. Les approches utilisées dans les NoC filaires reposent sur la superposition de plusieurs réseaux, un optimisé pour la latence et un pour la bande passante par exemple. Les données sont alors émises sur l'un des réseaux en fonction des caractéristiques que l'on cherche à optimiser et de la disponibilité de ces réseaux. Les approches utilisée dans les NoC optique et RF reposent sur le découpage de la bande passante totale en sous-bandes. Ces sous-bandes peuvent ensuite être allouées aux différents nœuds du réseau en fonction des besoins. Ainsi, les solutions utilisées dans les NoC filaires permettent d'adapter localement le réseau au trafic alors que les solutions optique et RF permettent une adaptation globale du réseau au trafic. Cependant, les solutions présentés reposent principalement sur phases de requête et d'attribution pouvant nuire à la réactivité de l'allocation dynamique et à la latence des communications.

3.5 Conclusion

Dans ce chapitre, nous avons tout d'abord étudié les différentes solutions proposées pour améliorer la bande passante et la latence. Nous avons ensuite présenté comment les NoC pouvaient s'adapter à des communications autres que celles point à point en présentant des architectures facilitant le multicast et le broadcast. Finalement, nous avons étudié les solutions permettant d'améliorer les performances pour une quantité de ressources de communication donnée en allouant ces ressources dynamiquement en fonction des besoins.

Le tableau 3.1 résume les avantages et inconvénients des différentes technologies utilisées dans les NoC.

Les réseaux filaires 2D, et dans une moindre mesure ceux en 3D, s'adaptent plus difficilement à des CMP comprenant un grand nombre de cœurs que les NoC optiques et RF. Cependant ils sont bien plus adaptés que ces derniers à un trafic local et homogène.

La RF, comme l'optique, offre la possibilité de créer simultanément plusieurs canaux de communication sur un même support physique. Ces canaux peuvent ensuite être alloués dynamiquement pour adapter le NoC aux besoins en communication. Cette allocation dynamique est plus difficile à mettre en place dans un NoC filaire 2D ou 3D puisque les ressources de communication utilisées dépendent directement des

Technologies	Avantages	Inconvénients
Filaire 2D	- simple et compatible CMOS - adapté au trafic local	- faible scalabilité - peu adapté au broadcast - difficilement reconfigurable
Filaire 3D	- diminution distances - spécialisation des couches - adapté au trafic local	- dissipation thermique - peu adapté au broadcast - difficilement reconfigurable
Optique	- temps de propagation - support broadcast - reconfigurabilité	- source externe - logique additionnelle - trafic local
RF	- temps de propagation - support broadcast - reconfigurabilité - compatible CMOS	- logique additionnelle - consommation à limiter - trafic local

Tableau 3.1 – Avantages et inconvénients des différentes technologies

nœuds communicant ensemble et du chemin physique les reliant. Cependant, parmi les architecture proposées, beaucoup reposent sur une phase de « requête »permettant aux nœuds du réseau de communiquer leurs besoins aux composants chargés d'allouer les canaux de communication. Il s'ensuit une phase d'« allocation »permettant à chaque nœud de savoir quels canaux utiliser avant de pouvoir émettre sur le réseau. La mise en place de la reconfiguration dynamique semble alors se faire au détriment de la diminution de la latence.

Par ailleurs, la plupart des NoC optiques et RF proposés offrent la possibilité d'établir des communications point à point entre deux nœuds quelconques du réseau. Cette propriété repose alors sur le fait qu'un message émis sur ces réseaux est reçu par l'ensemble des nœuds. Il est donc facile de transformer ces communications point à point en broadcast, ce qui n'est pas le cas avec les NoC filaires 2D ou 3D.

Pour finir, les NoC filaires 3D augmentent les contraintes de dissipation thermique par rapport aux NoC filaires 2D, RF ou optiques. Cependant, il ne faut pas oublier qu'une implémentation optimale d'un NoC optique ou RF passe souvent par l'utilisation d'une couche dédiée à cette technologie et donc à la création d'une puce 3D possédant uniquement deux couches.

Dans ce chapitre, nous avons présenté différentes solutions proposées pour répondre aux critères d'amélioration des performances identifiés au chapitre précédent :

— Augmentation de la bande passante
— Diminution de la latence
— Support des communications de type multicast et broadcast

— Allocation dynamique des ressources de communication

Cependant, comme nous venons de le voir, les architectures présentées apportent une solution à un ou plusieurs de ces points sans en traiter la totalité. Cela nous permet donc de reformuler la double problématique identifiée au chapitre précédant de la manière suivante :
Comment allouer dynamiquement les ressources de communication dans un NoC de manière cohérente avec l'amélioration de la latence et du débit ainsi que le support du broadcast ?

La RF étant, avec l'optique, la technologie offrant le plus de possibilités en terme de reconfiguration dynamique tout en étant intégrable physiquement à plus court terme que l'optique. C'est donc la technologie que nous avons choisi comme support du NoC que nous présenterons dans le chapitre suivant.

Chapitre 4

Architecture WiNoCoD

Sommaire

4.1 Introduction

Dans le chapitre précédent, nous avons présenté l'état de l'art des *Network on Chip* (NoC) en nous focalisant sur les solutions proposées pour améliorer un ou plusieurs des critères suivants :

— Augmentation de la bande passante
— Diminution de la latence
— Support des communications de type multicast et broadcast
— Allocation dynamique des ressources de communication

L'analyse de ces différentes solutions et de leurs limitations nous a permis de faire émerger la problématique suivante :
Comment allouer dynamiquement les ressources de communication dans un NoC de manière cohérente avec l'amélioration de la latence et de la bande passante ainsi que le support du broadcast ?

Dans ce chapitre, nous allons présenter la solution proposée pour répondre à cette problématique. Nous commencerons par présenter les principales caractéristiques de la solution dans la section 4.2. Nous présenterons ensuite l'architecture d'un *Chip Multiprocessor* (CMP) utilisant WiNoCoD dans la section 4.3. Finalement nous présenterons l'architecture du NoC RF dans la section 4.4.

4.2 Principes de l'architecture

4.2.1 Un réseau RF reconfigurable dynamiquement

Le but de l'architecture WiNoCoD est de fournir des mécanismes d'adaptation dynamique du réseau sur puce aux besoins en communication des applications s'exécutant sur un CMP.

Le premier prérequis à la mise en place d'un réseau reconfigurable dynamiquement est que la technologie utilisée lui permette d'être utilisable indifféremment par tous les nœuds composant le réseau. Il faut, d'une part, que le médium de communication permette à un nœud d'émettre vers n'importe quel autre nœud et, d'autre part, que la qualité de la communication soit constante quelque soit la source et de la destination. Or, c'est le cas de la RF, comme nous avons pu le voir dans le Chapitre 3. Le temps de propagation d'une onde électromagnétique est très court à l'échelle d'une puce, de l'ordre de 0,5 ns pour 100 mm. Il peut donc être considéré comme quasi constant, quelles que soient la source et la destination. Dans les NoC basés sur des technologies filaires, les temps de communication varient significativement en fonction de la distance entre la source et la destination. Ainsi dans ces architectures, le temps d'accès à une donnée peut varier de manière importante en fonction du

banc mémoire dans lequel elle se trouve. On parle alors d'architecture *Non Uniform Memory Access* (NUMA), et cette non uniformité des accès mémoire augmente avec la taille du réseau. L'utilisation de la RF, grâce à son temps d'accès quasi constant, permet donc d'augmenter la taille d'un CMP sans pour autant le rendre de plus en plus NUMA . Le choix de la RF et sa mise en place dans l'architecture proposée a donc pour but de permettre aux communications sur le NoC de ne dépendre ni de la source, ni de la destination.

Le deuxième prérequis à la mise en place d'un réseau reconfigurable dynamiquement est de pouvoir distribuer la bande passante disponible. Diviser cette bande passante en sous-bandes est donc une manière de créer des ressources de communication partageables que l'on peut répartir entre les différents nœuds du réseau en fonction des besoins. La bande passante offerte par les NoC RF, correspond directement à l'utilisation d'une certaine plage de fréquences. Ainsi pour répartir la bande passante de ces réseaux il est possible d'utiliser des techniques de multiplexage fréquentiel telles que l'*Orthogonal Frequency Division Multiplexing* (OFDM). Une autre solution serait d'utiliser un multiplexage temporel où la bande passante est allouée successivement aux différents nœuds du réseau. Cependant cette technique présente l'inconvénient de ne pas permettre des communications réellement parallèles et donc d'augmenter la latence des communications [Chang, 2005]. De plus le multiplexage temporel nécessite une synchronisation plus forte des différents nœuds du réseau que le multiplexage fréquentiel. L'utilisation d'un multiplexage fréquentiel dans l'architecture proposée doit donc permettre de créer les ressources de communication attribuables par le mécanisme d'allocation dynamique.

Ainsi, la RF offre d'une part la possibilité de créer un canal de communication utilisable indifféremment par tous les nœuds du réseau, et d'autre part la possibilité de diviser et de répartir la bande passante totale entre ces différents nœuds. De plus, la RF est directement intégrable avec les technologies CMOS actuelles. Finalement, l'utilisation d'un guide d'ondes physique comme support de l'onde RF permet de limiter les interférences et donc la puissance d'émission nécessaire par rapport aux solutions sans-fil. C'est pour ces raisons que nous avons retenu un médium utilisant une onde RF circulant sur un guide d'ondes comme technologie support du NoC reconfigurable WiNoCoD.

4.2.2 Un réseau hiérarchique

Comme nous venons de le voir, le temps de propagation d'une onde RF est quasi constant à l'échelle d'une puce. De plus, comme nous le verrons dans la section 4.4, le temps nécessaire aux traitements effectués à l'émission et à la réception du signal RF ne dépend ni de la source, ni de la destination. L'utilisation de la RF dans un

NoC offre donc la possibilité de transmettre des données en tout point du circuit en temps quasi constant.

Si le temps de transmission des données sur le NoC RF est quasi constant, il est cependant supérieur au temps nécessaire pour envoyer une donnée d'un routeur à son plus proche voisin dans un réseau de routeurs filaires. C'est pour cette raison que l'architecture WiNoCoD n'utilise la RF qu'à partir d'une certaine distance entre la source et la destination. Cette distance peut être estimée en fonction des caractéristiques du réseau filaire et du NoC RF, comme nous le verrons dans le Chapitre 6.

Le choix d'une architecture hiérarchique où la RF permettrait d'effectuer les communications longue distance et où les réseaux filaires permettraient de transporter les communications locales est dans ce cadre tout à fait naturel.

4.2.3 L'OFDMA

Nous utilisons l'*Orthogonal Frequency Division Multiple Access* (OFDMA) pour diviser la bande disponible en autant de sous-bandes que nécessaires au bon fonctionnement des mécanismes d'allocation dynamique. Cette technique basée sur l'OFDM permet d'avoir une occupation optimale de la bande tout en s'assurant qu'il n'y ait pas d'interférence entre les différentes sous-bandes. Des sous-bandes avec de telles caractéristiques peuvent être générées par une IFFT en émission et traduites par une FFT en réception. Le nombre de sous-bandes dépend de la largeur de la bande et de celle des sous-bandes. Ces paramètres sont dimensionnés par la technologie de réalisation du circuit, ainsi que par les performances des émetteurs/récepteurs RF. À largeur de bande constante, plus le nombre de sous-bandes est important, plus celles-ci seront étroites. Il est alors nécessaire d'avoir des composants RF plus performants, afin d'être en mesure de distinguer une sous-bande de ses voisines. Une solution alternative serait d'utiliser le *Single-Carrier Frequency Division Multiple Access* (SC-FDMA) comme schéma d'accès. Cette solution permet de diminuer le *Peak-to-Average Power Ratio* (PAPR). Si le SC-FDMA permet de relâcher des contraintes sur les émetteurs-récepteurs, il nécessite, par rapport à une architecture OFDMA, l'ajout d'une FFT supplémentaire en émission et d'une IFFT supplémentaire en réception. Ces composants étant les plus gourmands en surface et en énergie, comme nous le verrons dans le Chapitre 6, nous avons choisi d'utiliser l'OFDMA.

De par le découpage de la bande totale en sous-bandes, c'est la largeur des sous-bandes qui fixe la durée d'un symbole OFDMA et donc sa périodicité d'émission. Cette période, aussi appelée cycle OFDM, correspond au cycle de notre NoC RF, on parlera par la suite de cycle RF. Ainsi, dans le cas d'étude présenté dans les Chapitres 5 et 6, pour une bande de 20 GHz divisée en 512 sous-bandes nous obtenons des sous-bandes de 39 MHz, soit un cycle réseau de 25,6 ns

4.2.4 L'algorithme d'allocation dynamique distribué

L'adaptation dynamique du NoC aux communications repose sur un algorithme d'allocation dynamique distribué. Les sous-bandes créées grâce à l'OFDM sont regroupées pour former des canaux. Ce sont ces canaux de communication qui sont alloués par l'algorithme d'allocation aux différents nœuds du réseau.

Cet algorithme s'exécute de manière distribuée dans chaque nœud du réseau en analysant l'évolution du trafic provenant du réseau filaire. Chaque nœud du NoC RF possède une file d'attente servant de tampon entre le réseau filaire et le réseau RF. C'est l'analyse du taux de remplissage de cette file d'attente qui permet d'évaluer le trafic provenant du réseau filaire.

Chaque nœud transmet donc le taux de remplissage de sa file d'attente à tous les autres nœuds du NoC RF. Cette transmission se fait sur le canal zéro, utilisé comme canal de service, en même temps que les données utiles sont transférées sur les autres canaux.

Le mécanisme d'allocation dynamique sera présenté plus en détail dans la section 4.4. La sous-section 4.4.3 présentera les principes de l'algorithme alors que la sous-section 4.4.4 présentera son intégration matérielle.

4.2.5 Le broadcast

Le broadcast est intrinsèquement lié à l'architecture et aux mécanismes d'allocation dynamique des ressources de communication. Comme nous le verrons dans la Section 4.4, les canaux de communication sont attribués pour émettre et non pour recevoir. Ainsi, ce n'est qu'après avoir reçu les données et avoir commencé à les décoder qu'un nœud peut savoir si elles lui sont réellement destinées. Il faut donc que toutes les données émises sur le NoC RF soient reçues par tous les nœuds. De plus, le broadcast permet à l'algorithme distribué de ne pas générer plus de trafic qu'un algorithme centralisé puisque les données émises sur le canal de service sont reçues par tous les nœuds du NoC RF.

Il suffit alors de définir une plage d'adresse particulière ou un bit indiquant si le message est un broadcast pour qu'il en soit effectivement un. L'architecture proposée permet ainsi de faire du broadcast sans surcoût par rapport aux communications point à point.

4.3 La hiérarchie de WiNoCoD

Nous avons donc fait le choix d'un réseau hiérarchique afin de profiter au mieux des qualités intrinsèques des différents types de réseaux. La figure 4.1 illustre l'architecture hiérarchique du CMP utilisant le NoC WiNoCoD. Il est composé de grappes qui

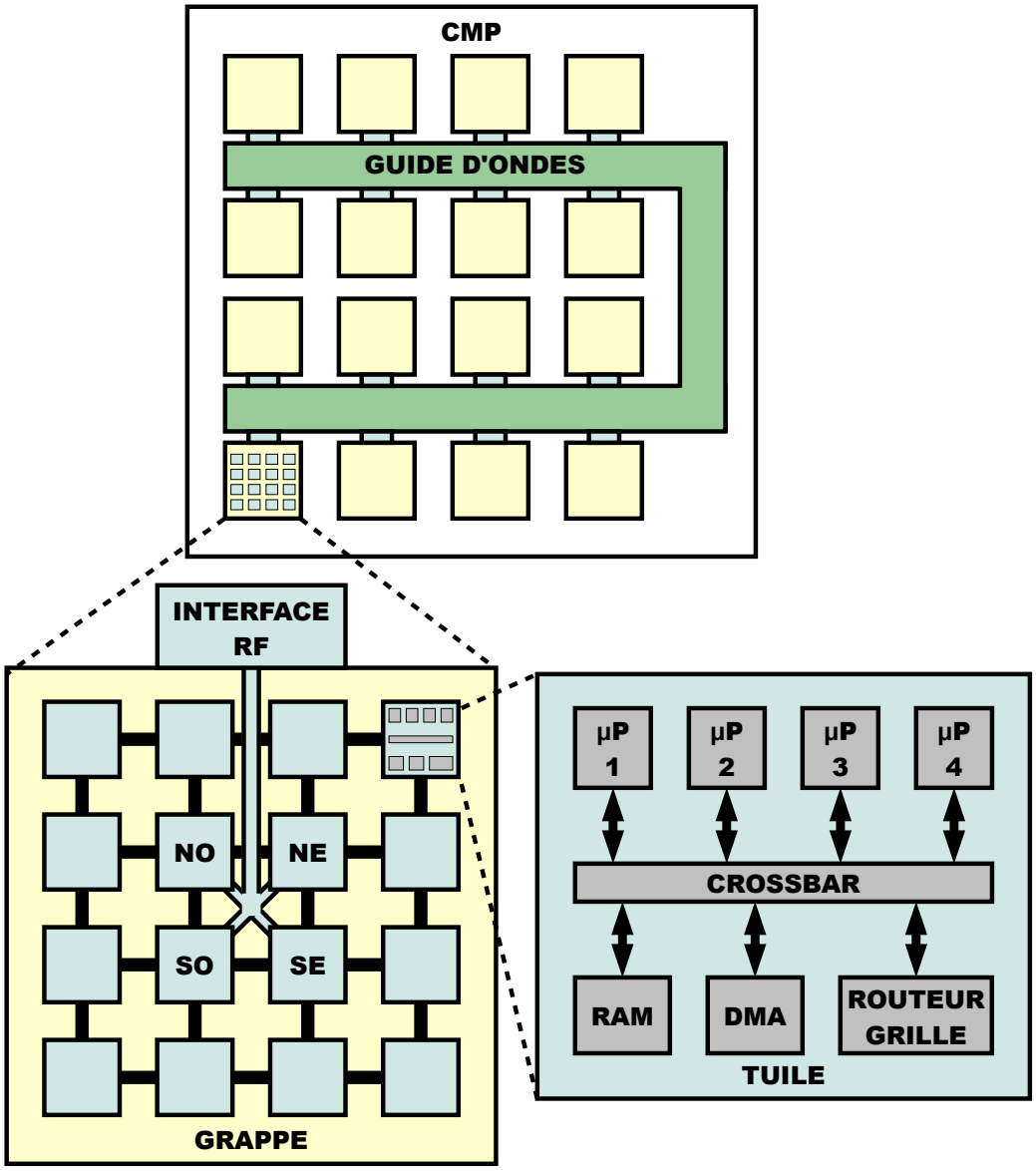

Figure 4.1 – Organisation hiérarchique du CMP et topologie du NoC pour une configuration de 16 grappes de 16 tuiles de 4 cœurs

contiennent des tuiles, elles-mêmes constituées de plusieurs cœurs et d'autres composants. À chaque niveau hiérarchique (tuile, grappe, global) correspond un niveau d'interconnexion donné.

4.3.1 Tuiles

La tuile est le niveau hiérarchique le plus bas. Une tuile contient P processeurs, ayant chacun un cache instructions et un cache données, ainsi qu'une mémoire locale et potentiellement divers périphériques. Tous ces composants sont connectés par un crossbar local comme on peut le voir dans la figure 4.1. L'intérêt d'un crossbar est qu'il permet aux communications de se faire en parallèle entre les différents initiateurs (processeurs, DMA) et cibles (DMA, mémoire, périphériques). Un processeur peut, par exemple, communiquer avec un périphérique pendant que le DMA communique avec la mémoire. Si le crossbar présente cet avantage par rapport à un bus, il partage l'inconvénient de passer difficilement à l'échelle. Alors que les performances d'un bus ne passent intrinsèquement pas à l'échelle, le crossbar est lui limité par l'augmentation quadratique du nombre de fils en fonction du nombre d'éléments à connecter. Pour cette raison, le crossbar de notre tuile est aussi connecté à un routeur qui lui permet de communiquer avec les autres tuiles.

4.3.2 Grappes

Les tuiles sont connectées via leur routeur pour former des grappes de $M \times M$ tuiles. Chaque routeur est ainsi relié aux quatre routeurs les plus proches. On a donc une grille formée par un réseau point à point de routeurs filaires. La grille passe à l'échelle du point de vue de la bande passante puisque celle-ci augmente en même temps que la taille de la grille. Malheureusement la latence va augmenter elle aussi en même temps que la dimension de la grille. Il devient alors de plus en plus coûteux de faire communiquer deux tuiles quelconques de la plateforme. Si la taille d'une tuile est intrinsèquement limitée par les caractéristiques du crossbar, c'est donc l'augmentation de la latence qui limite les dimensions d'une grappe.

4.3.3 CMP

On utilise donc le NoC RF comme troisième niveau d'interconnexion quand sa latence devient plus faible que celle de la grille. Cela permet d'intégrer plus de cœurs sur une même puce tout en maîtrisant l'augmentation de la latence. La section suivante présente ce NoC RF en détail.

4.4 Le NoC RF

Les grappes de tuiles utilisent le NoC RF pour communiquer entre elles. Chaque grappe dispose d'une interface RF reliée d'une part, au réseau filaire, et d'autre part, au guide d'ondes. C'est cette interface RF qui permet à la grappe de transmettre et de recevoir des données via le guide d'ondes. Un nœud du NoC RF est donc formé par une grappe et son interface RF.

4.4.1 L'architecture de l'interface RF

C'est grâce à l'interface RF présentée dans la Figure 4.2 qu'une grappe est capable d'émettre et de recevoir des données via le guide d'ondes. C'est aussi dans cette interface qu'a lieu la reconfiguration dynamique du NoC RF.

Les données arrivant du réseau filaire sont lues par l'arbitre qui en extrait les données utiles afin de créer les *FLow control unIT* (FLIT) qui circuleront sur le NoC RF. Une fois cette opération effectuée, le FLIT est stocké dans la FIFO d'émission de l'arbitre. Comme nous le verrons par la suite, l'état de remplissage ce cette FIFO est l'information de base utilisée par l'algorithme d'allocation dynamique des ressources de communication. Cet état de remplissage fait partie des informations de service qui sont transmises à l'ensemble du réseau.

Les FLIT issus de l'arbitre sont ensuite découpés en groupes de bits dont la taille correspond au codage utilisée (BPSK, QPSK, 16-QAM ou 64-QAM). Les informations de service provenant de l'arbitre sont elles aussi mises sous la forme d'une constellation I/Q par le codeur. Une fois codés par le codeur, les FLIT de données et les informations de service sont transmis au placeur.

Le placeur parallélise les FLIT provenant du codeur pour les placer sur les entrées de la IFFT correspondant aux canaux actuellement alloués au nœud. Il place aussi les informations de service sur la portion du canal de service qui lui est dédié. Les autres entrées sont remplies de zéros pour ne pas perturber l'émission des autres nœuds du NoC RF.

La IFFT génère les symboles OFDM qui sont ensuite transmis à l'émetteur qui se charge de les émettre sur le guide d'ondes. Les caractéristiques de l'émetteur et du guide d'ondes permettent aux données d'être reçues par les récepteurs de tous les nœuds du NoC RF.

Une fois les données reçues par le récepteur, elles suivent le chemin inverse dans la partie réception de l'interface RF, et ce dans tous les nœuds du réseau. Le signal RF analogique est tout d'abord reçu par le récepteur et converti en signal numérique. Il est ensuite traité par la FFT qui décompose le signal reçu sur chaque sous-bande.

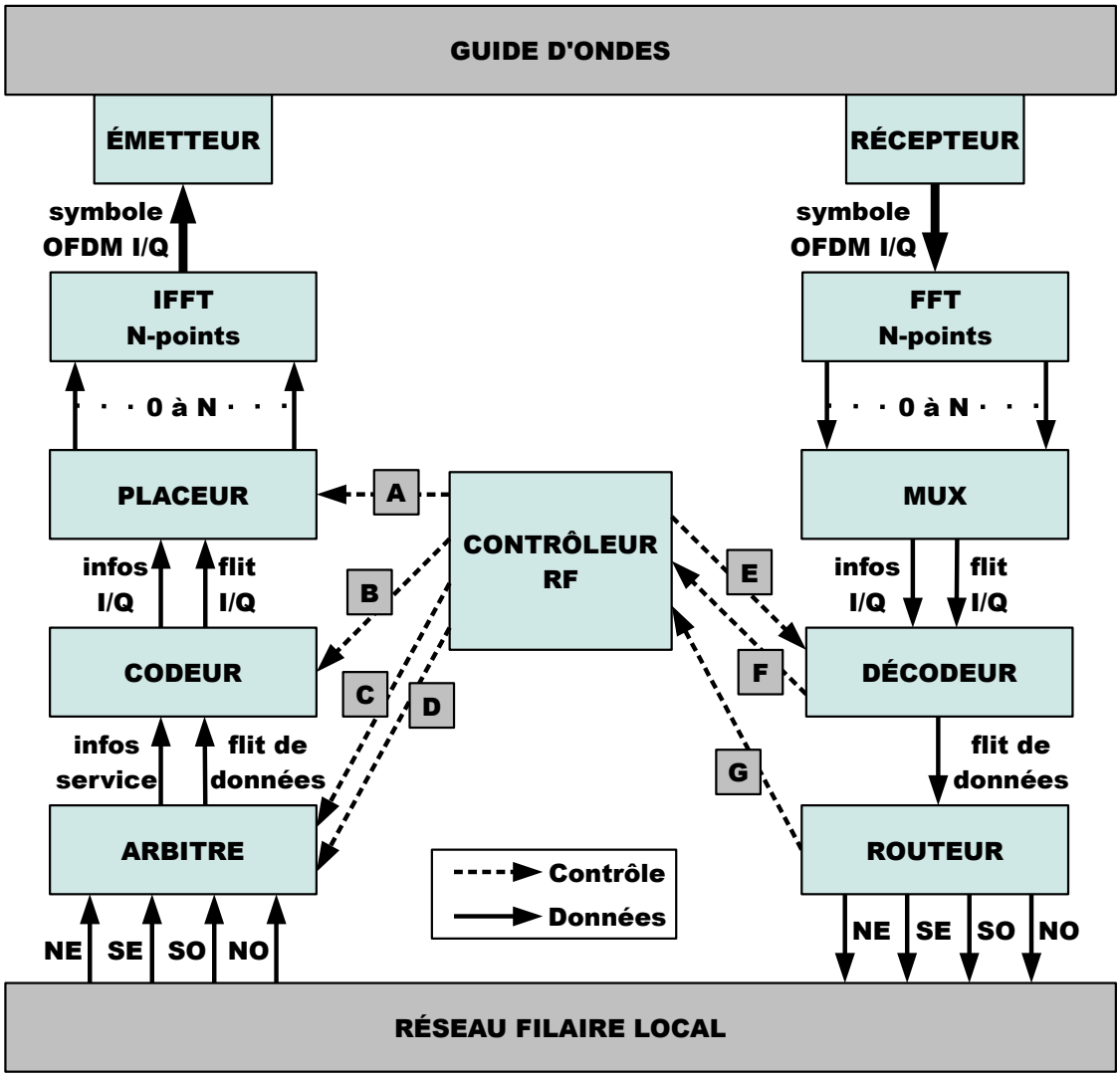

Figure 4.2 – Schéma fonctionnel de l'Interface RF

Ces sous-bandes sont ensuite transmises au multiplexeur, chaque sous bande correspond à un flit. Les FLIT provenant des différents nœuds du réseau sont donc maintenant reconstitués mais toujours sous forme de constellations I/Q.

Les FLIT étant toujours codés sous la forme d'une suite de symboles sur une constellation I/Q, ils sont envoyés au démodulateur. Une fois le décodage terminé, les FLIT sont maintenant sous forme d'une suite de bits correspondant exactement à l'information transmise par l'arbitre source.

Le routeur peut maintenant analyser les FLIT reçus pour savoir s'ils sont ou non destinés au réseau filaire local. Une fois cette information connue, il les place dans une FIFO correspondant au nœud dont provient le FLIT afin de pouvoir reconstituer les paquets. Ce stockage intermédiaire est nécessaire puisque rien ne garantit que deux FLIT consécutifs proviennent de la même source. Une fois tous les FLIT d'un paquet arrivés, celui-ci est placé dans la FIFO de réception correspondant au port de sortie vers lequel le message sera envoyé. Ces FIFO permettent d'absorber le trafic provenant du NoC RF quand le réseau filaire n'est pas disponible.

Comme chaque grappe décode l'intégralité de chaque symbole OFDM, toutes les données circulant sur le NoC RF sont visibles de toutes les grappes. Le broadcast est une caractéristique intrinsèque des communications circulant sur le NoC RF. L'architecture WiNoCoD permet donc de faire du broadcast sans surcoût par rapport aux communications point à point.

Finalement le contrôleur RF est chargé d'allouer dynamiquement les canaux de communication, de choisir la modulation à utiliser et de mettre en place un contrôle de flux. Pour cela, il utilise les informations de service émises par les arbitres de chaque nœud et transmises directement par le décodeur après réception via le signal F de la Figure 4.2.

Le contrôleur RF utilise l'état de remplissage des FIFO d'émission des arbitres de chaque nœud pour allouer les canaux de communication. Cette état de remplissage faisant partie des informations transmises grâce aux canaux de service. Il transmet ensuite le nouveau schéma d'allocation au placeur via le signal A. Son algorithme d'allocation dynamique lui permet aussi de choisir le type de modulation à utiliser. Il indique ensuite ce choix au codeur via le signal B et au décodeur via le signal E.

Le contrôleur RF doit aussi mettre en place le contrôle de flux permettant de prévenir la perte de données en cas de saturation. L'état de remplissage des FIFO de réception des routeurs est transmise du routeur au contrôleur RF via le signal G qui le transmet ensuite à l'arbitre via le signal C. Chaque nœud connait ainsi l'état de remplissage de sa propre FIFO de réception. L'arbitre peut alors transmettre cette information sur le canal de service du nœud. Or comme chaque contrôleur RF reçoit les informations provenant de l'ensemble des canaux de service du système, il connait la liste des nœuds ne pouvant plus recevoir de données. Il peut donc indi-

quer à son arbitre via le signal D l'état global du système ce qui permet à l'arbitre de bloquer l'émission de données vers les nœuds dont la FIFO de réception est pleine.

4.4.2 Description détaillée des composants

Nous allons maintenant présenter les différents composants du NoC RF WiNoCoD. Les détails d'implémentation seront présentés dans le chapitre suivant avec la description du modèle utilisé pour évaluer l'architecture WiNoCoD.

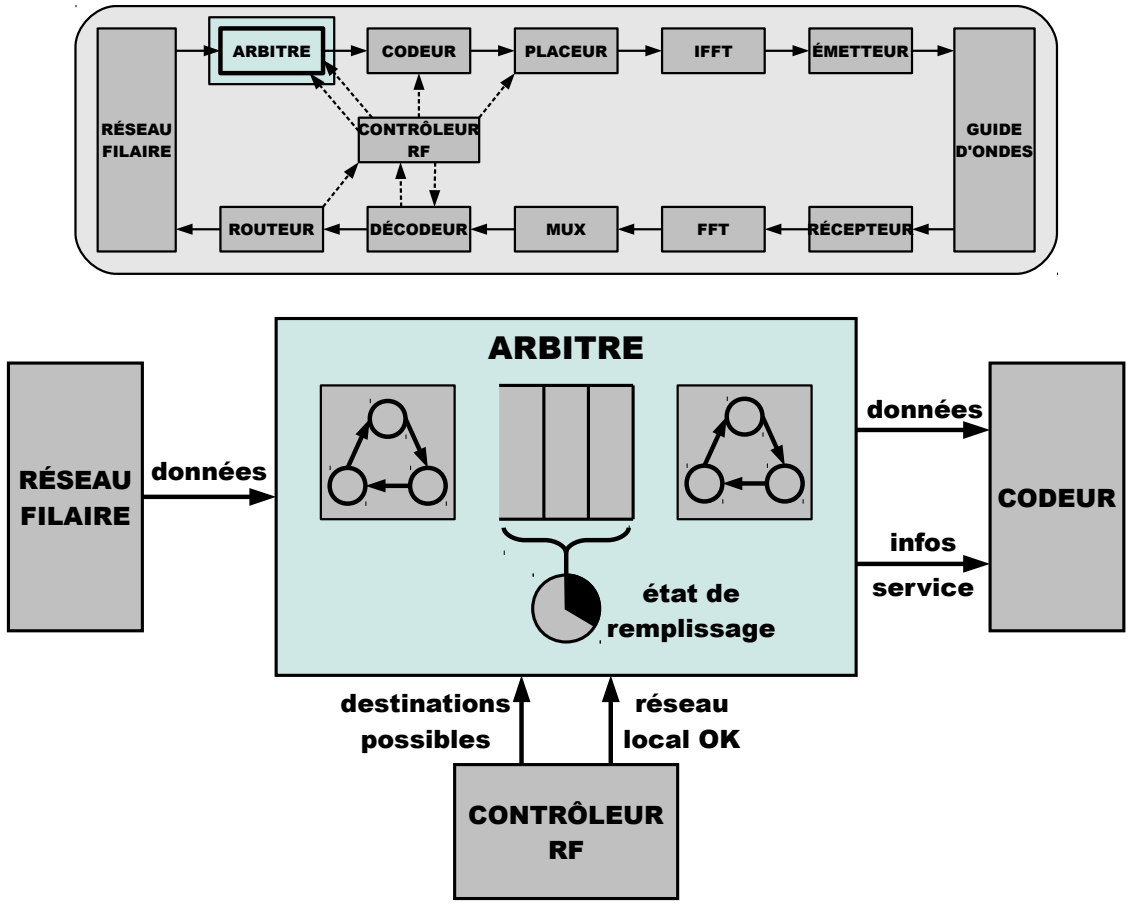

Figure 4.3 – Arbitre

4.4.2.1 L'arbitre

L'arbitre, représenté sur la Figure 4.3, est chargé de traiter les données provenant du réseau filaire. Pour ce faire, son fonctionnement se fait en deux temps.

Dans un premier temps, il doit gérer les données consommées sur le réseau filaire. Celles-ci sont analysées pour créer les FLIT qui circuleront ensuite sur le NoC RF. Grâce au contrôleur RF et aux mécanismes que nous détaillerons dans la description de celui-ci, l'arbitre connait la liste des nœuds en état de recevoir des données.

Il peut donc analyser les paquets provenant du réseau filaire et bloquer momentanément la communication si la destination du paquet n'est pas prête à recevoir des données. Une fois le FLIT RF constitué, il est stocké dans une file d'attente qui permet de gérer la variation du trafic provenant du réseau filaire. Cette file d'attente permet aussi, en analysant son état de remplissage, d'évaluer le trafic provenant du réseau filaire et donc les besoins en ressources de communication.

Dans un deuxième temps, l'arbitre doit gérer le transfert des FLIT de données vers le décodeur, mais aussi le transfert de différentes informations de service. Les FLIT présents dans la FIFO d'émission sont placés sur les ports de sortie de l'arbitre et consommés par le codeur en fonction de sa disponibilité. En plus des données utiles, l'arbitre transmet des données de service utilisées par les différents contrôleurs RF du réseau pour l'allocation dynamique et pour le contrôle de flux. Ces informations de service correspondent à l'état de remplissage de la FIFO d'émission de l'arbitre et à celui de la FIFO de réception du routeur.

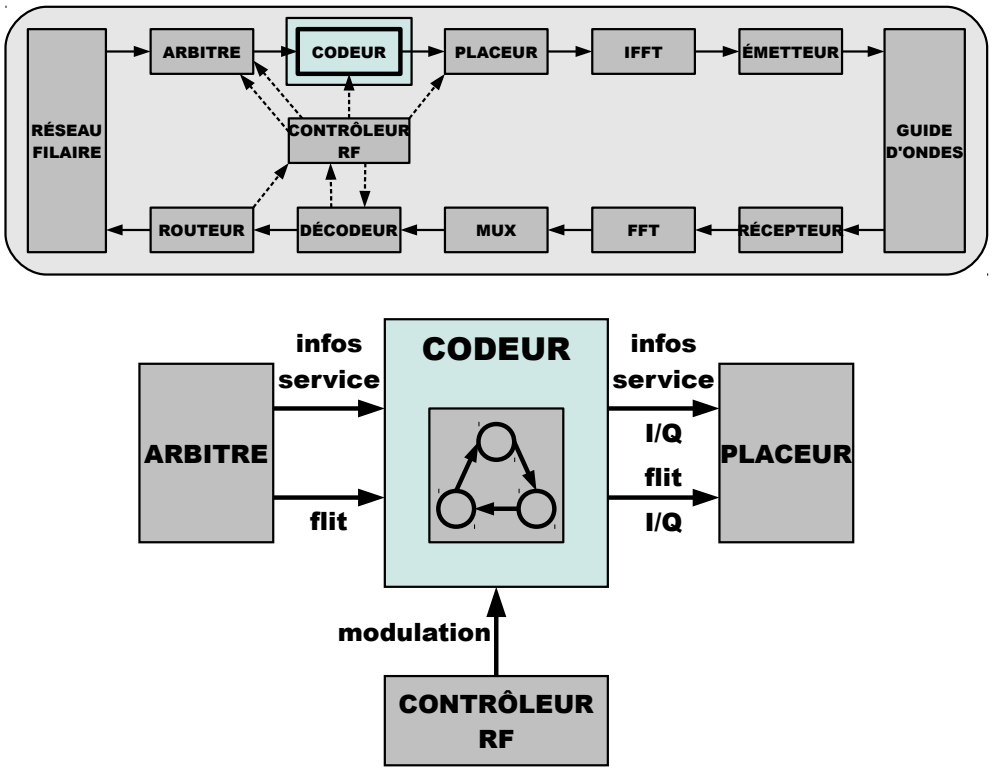

Figure 4.4 – Codeur

4.4.2.2 Le codeur

Le codeur, représenté sur la Figure 4.4, est chargé d'effectuer la modulation numérique des FLIT transmis par l'arbitre, ainsi que des informations de service. Cette modulation numérique à pour but de pouvoir optimiser l'utilisation de la bande passante ainsi que le taux d'erreur du reste de la chaine de transmission.

Les données sont modulées en BPSK, QPSK ou 16-QAM suivant les indications du contrôleur RF. Celui-ci choisit une modulation suivant le besoin total en bande passante en entrée du NoC RF. C'est donc dans le codeur que se fait une partie de l'adaptation du NoC RF aux besoins en communications. Une fois les données modulées, elles sont transmises au placeur.

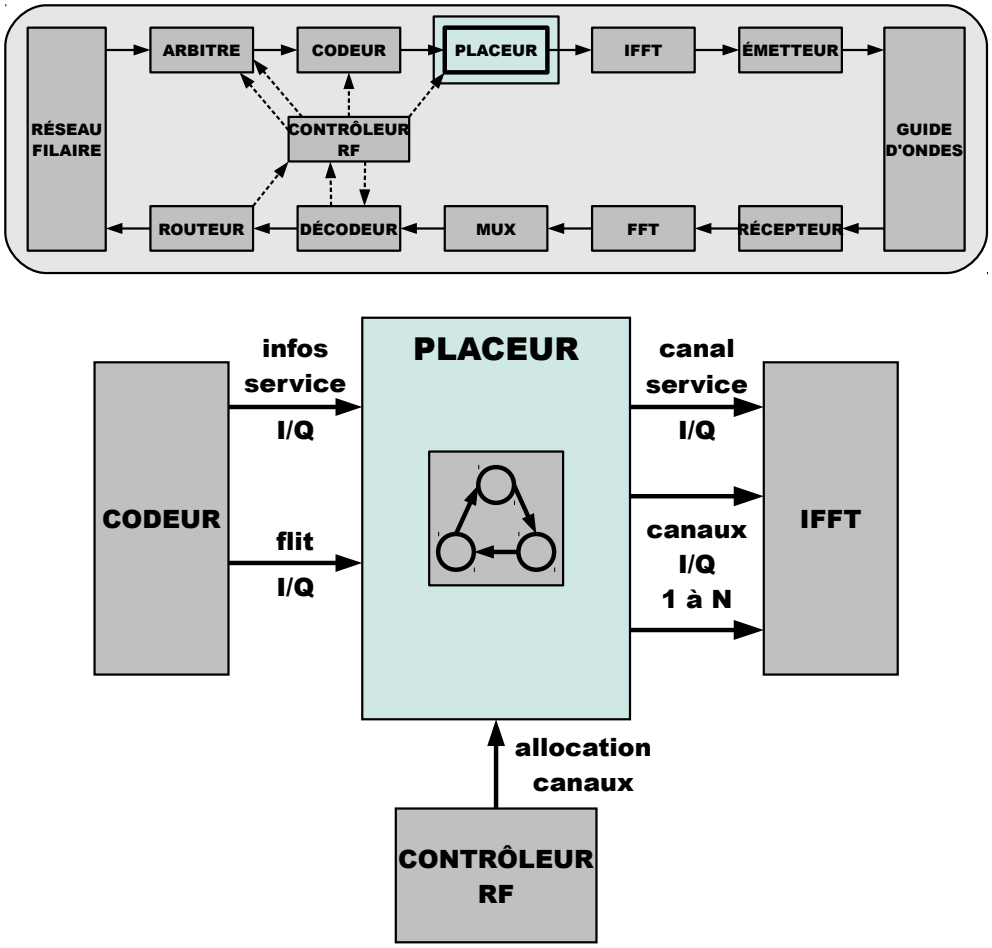

Figure 4.5 – Placeur

4.4.2.3 Le placeur

Le placeur, représenté sur la Figure 4.5, est chargé de placer les FLIT modulés provenant du codeur sur les entrées de la IFFT. Chaque entrée de la IFFT correspond à une sous-bande, elles sont regroupées de manière à former les canaux de communication du réseau. Dans un système comprenant N nœuds, les entrées sont regroupées en N+1 canaux afin de permettre à chaque nœud d'utiliser un canal dans la configuration statique tout en conservant un canal de service.

C'est ici qu'est appliqué le schéma d'allocation provenant du contrôleur RF puisque c'est le placeur qui met effectivement en place l'allocation des canaux de communication et donc de la bande passante au sein du NoC RF. Ce schéma d'allocation est

mis à jour par le contrôleur RF au début de chaque cycle RF. En plus des données utiles, le placeur transmet les informations de service de l'arbitre sur l'entrée de la IFFT correspondant au canal zéro. Ce canal est utilisé comme canal de service et chaque nœud du réseau s'y voit attribué un certain nombre de sous-bandes.

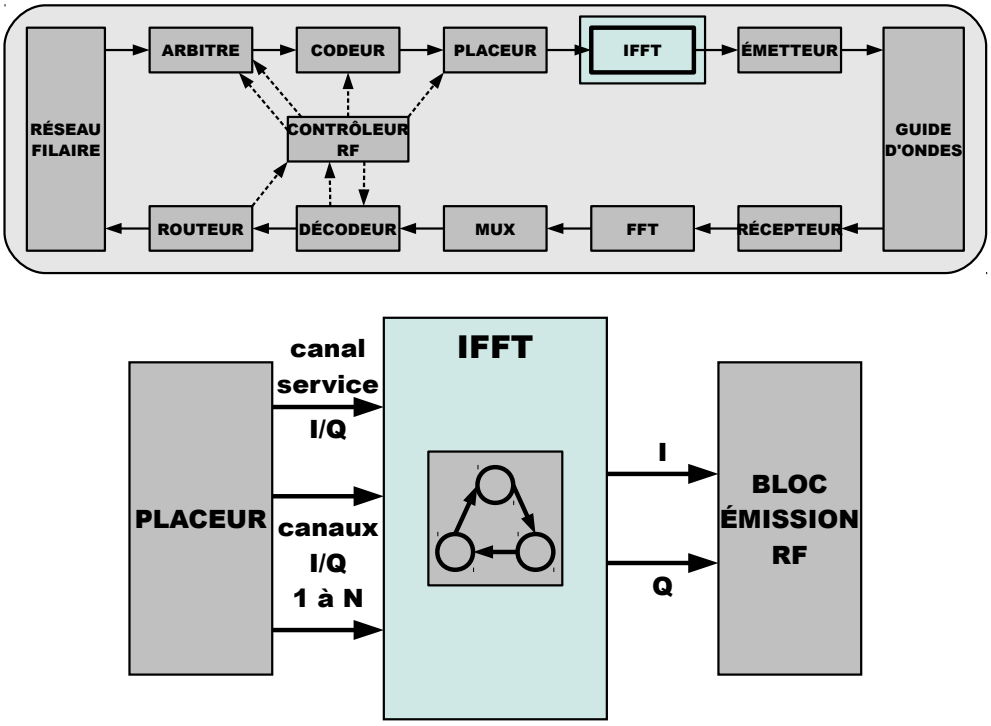

Figure 4.6 – IFFT

4.4.2.4 La IFFT

Chaque symbole créé par le codeur est positionné par le placeur sur l'une des entrées de la IFFT, représentée sur la Figure 4.6. La IFFT permet de diviser la bande totale disponible en sous-bandes orthogonales. L'utilisation de sous-bandes orthogonales permet de minimiser les interférences et tout en optimisant l'utilisation de la bande passante.

La IFFT permet donc d'émettre tous les symboles en parallèle en utilisant les différentes sous-bandes. De plus ces données peuvent être envoyées vers des destinations différentes permettant ainsi un parallélisme plus important dans les communications, notamment dans le cas où plusieurs réseaux seraient multiplexés sur le NoC RF. La IFFT possède deux sorties, une pour la composante réelle du signal et une pour la composante imaginaire.

4.4.2.5 Le bloc de transmission RF

La Figure 4.7 représente le bloc de transmission RF, la première étape réalisée par ce composant se fait par l'intermédiaire de deux CNA, un pour la composante réelle

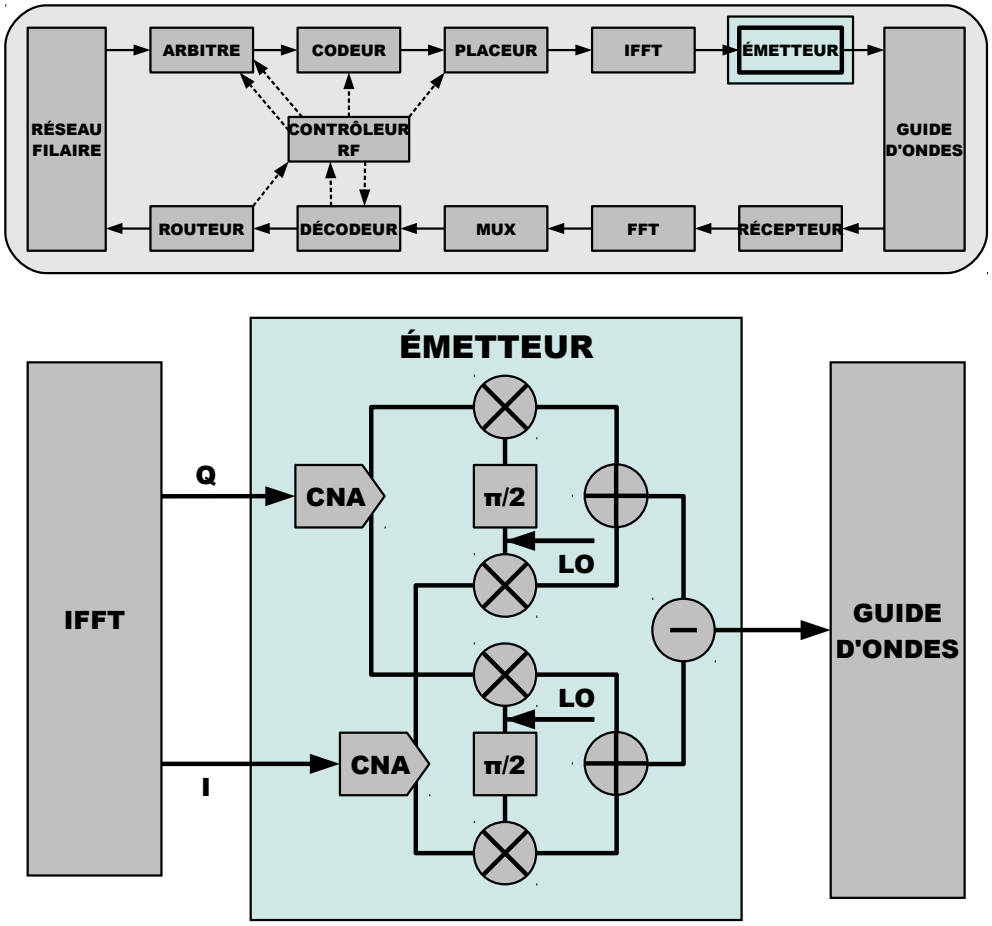

Figure 4.7 – Bloc de transmission RF

du signal et un pour la composante imaginaire. Une fois effectuée la conversion numérique/analogique du signal, celui-ci passe par un mélangeur qui permet de le combiner avec le signal provenant de l'oscillateur local. Les deux signaux ainsi créés sont ensuite combinés pour être transmis sur le guide d'ondes.

4.4.2.6 Le guide d'ondes

Pour chaque interface RF, le bloc de transmission Tx et le bloc de réception Rx sont connectés au guide d'ondes, représenté sur la Figure 4.8. Ce guide d'ondes est réalisé grâce à une ligne micro-ruban et l'accès à cette ligne se fait via un couplage capacitif. Cette technique consiste à laisser un espace entre l'émetteur/récepteur et le guide d'ondes. Le couplage capacitif permet de diminuer les réflexions et donc d'avoir une ligne de transmission plus homogène sur l'ensemble du circuit. Cette technique a été proposée par des partenaires du projet WiNoCoD [Hamieh *et al.*, 2014].

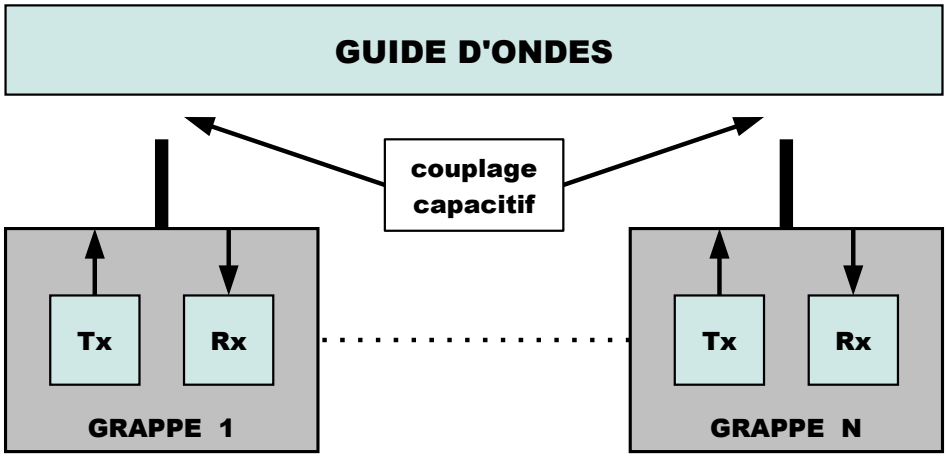

Figure 4.8 – Guide d'ondes

4.4.2.7 Le bloc de réception RF

La première étape réalisée dans le bloc de réception, représenté sur la Figure 4.9, est la séparation des composantes réelles et imaginaires du signal. Le signal est ensuite combiné avec l'oscillateur local et passe à travers un filtre passe bas pour isoler le signal de la bande porteuse. Finalement, le signal analogique est converti en signal numérique de manière à pouvoir être transmis et utilisé par la suite de la chaine de réception du NoC RF.

4.4.2.8 La FFT

Le bloc de réception RF fournit à la FFT, représentée sur la Figure 4.10, les composantes imaginaires et réelles du signal sous forme numérique. Ce signal correspond à l'ensemble de la bande passante. Le rôle de la FFT est de découper ce signal de manière à retrouver les différentes sous-bandes, la FFT présente donc en sortie autant de points de constellation qu'il y a de sous-bandes dans l'architecture. Les sous-bandes sont regroupées par canaux. Le canal zéro qui est le canal de service contient alors l'état des FIFO de chaque nœud du NoC RF alors que les autres canaux contiennent les données utiles.

4.4.2.9 Le multiplexeur

Le multiplexeur, représenté sur la Figure 4.11, dispose en entrée de la totalité des sous-bandes reconstituées par la FFT. Les sous-bandes qui ont été regroupées pour constituer des canaux sont capables de transmettre un FLIT entier en un cycle RF. Le multiplexeur va donc transmettre les données provenant de la FFT suivant cette granularité en commençant par le canal zéro qui est le canal de service. À chaque cycle d'horloge, le multiplexeur transmet les données présentes sur un canal en indiquant s'il s'agit ou non du canal de service.

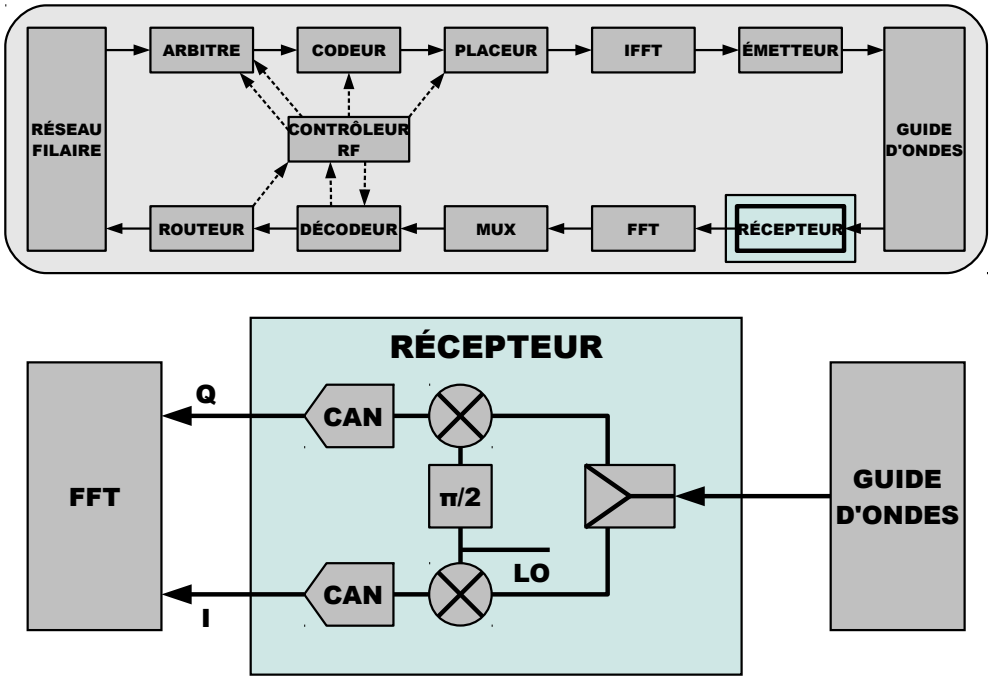

Figure 4.9 – Bloc de réception RF

Le canal de service contient l'état de remplissage des FIFO d'émission et de réception de toutes les grappes du système. Il est important de le traiter en priorité puisque les données qu'il transmet sont utilisées par l'algorithme d'allocation dynamique. Avoir ces données au plus tôt de chaque symbole OFDM permet ainsi d'avoir plus de temps pour exécuter cet algorithme.

4.4.2.10 Le décodeur

Le décodeur, représenté sur la Figure 4.12, traite les données provenant du multiplexeur au rythme d'un FLIT par cycle d'horloge. Il décode les données en fonction de la modulation indiquée par le contrôleur RF. Le choix d'un type de modulation ou d'un autre est fait suivant l'algorithme d'allocation dynamique puisque c'est un des deux paramètres qui permet d'influer sur le débit du NoC RF. Le FLIT ainsi créé est ensuite transmis au contrôleur RF s'il s'agit de données provenant du canal de service, et au routeur dans le cas contraire.

4.4.2.11 Le routeur

Le routeur, représenté sur la Figure 4.13, reçoit les données du décodeur sous forme de FLIT RF. Pour pouvoir les transmettre sur le réseau filaire il doit préalablement les reconvertir sous la forme de FLIT filaire. Ce n'est qu'à partir de ce moment qu'il est possible de savoir si le message arrivant est destiné au réseau filaire local ou non. Un flit est transmis au réseau filaire s'il est adressé à un des éléments qui lui est connecté ou s'il s'agit d'un broadcast.

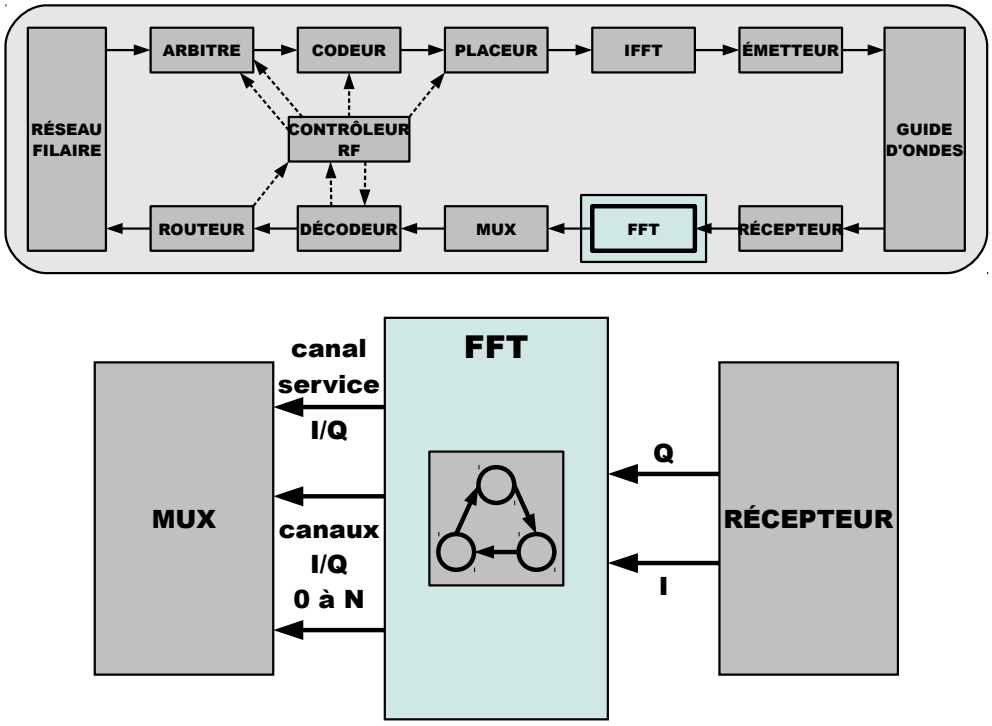

Figure 4.10 – FFT

Les données sont ensuite stockées dans une des FIFO intermédiaires du routeur. Le routeur possède une FIFO intermédiaire par nœud du NoC RF. Il est nécessaire de stocker les données de cette manière car les transactions du réseau filaire se font par paquet. Il est donc tout a fait possible que lors d'un cycle RF, la totalité des FLIT d'un paquet ne soit pas encore arrivée. Il est donc important de reconstituer ces paquets avant de les stocker dans la FIFO principale pour éviter tout mélange entre les FLIT provenant des différents nœuds du NoC RF.

Une fois les paquets placés dans cette FIFO, ils seront envoyés sur le réseau filaire selon la disponibilité de celui-ci. De plus, tout comme la FIFO d'émission de l'arbitre, la FIFO de réception du routeur possède un mécanisme permettant de surveiller son état de remplissage. Cet état de remplissage est utilisé pour mettre en place un mécanisme de contrôle de flux permettant au NoC RF d'arrêter d'envoyer des données vers un nœud du réseau si le réseau filaire de celui-ci sature. Il faut donc fixer un seuil à partir duquel le contrôleur RF utilise le canal de service pour indiquer au reste du réseau que le nœud ne peut plus recevoir de données temporairement. Ce seuil doit être fixé de manière à ce que la FIFO de réception soit capable d'absorber la quantité de données pouvant arriver pendant le temps de prise en compte du message d'alerte par le reste du réseau.

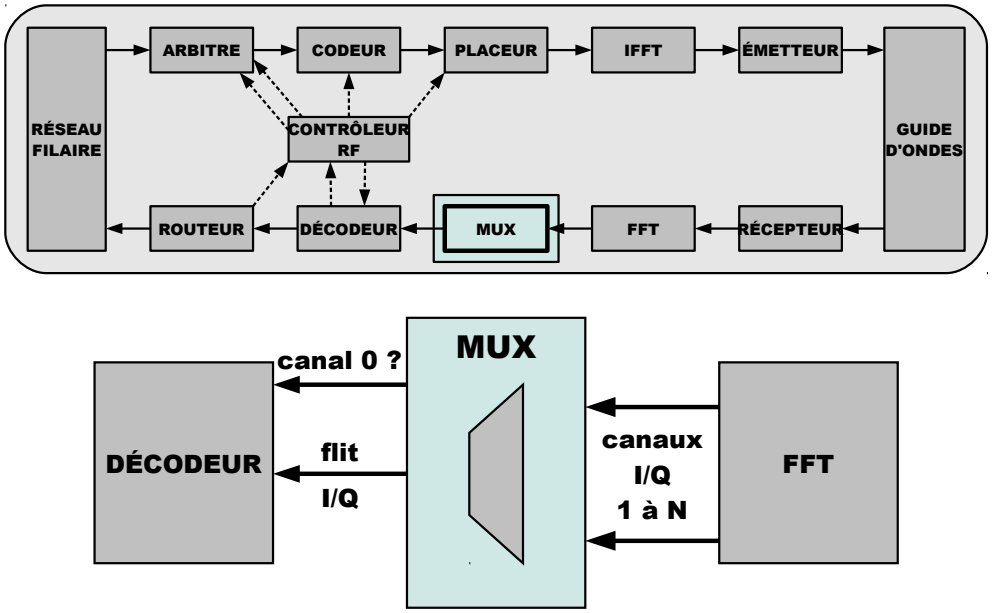

Figure 4.11 – Multiplexeur

4.4.3 L'algorithme d'allocation dynamique

L'objectif du NoC WiNoCoD est de proposer un réseau capable d'allouer ses ressources de communication en fonction des besoins des applications s'exécutant sur l'architecture à un instant donné. Il faut donc développer et mettre en place un algorithme capable d'offrir ce service. Dans cette section, nous allons présenter l'un des algorithmes proposées dans le cadre du projet WiNoCoD.

L'algorithme s'exécute de manière distribuée dans chaque nœud. Si cela présente l'inconvénient de devoir faire le calcul autant de fois qu'il y a de nœuds dans le réseau, et donc de répliquer des composants, cela permet aussi de s'affranchir de l'envoi de messages de reconfiguration sur le NoC RF. Cela évite ainsi de générer du trafic supplémentaire, tout en minimisant le délai nécessaire à la mise en place d'une nouvelle configuration, une fois que celle-ci a été choisie.

L'algorithme proposé alloue les ressources de communication du réseau en fonction des besoins d'émission de chaque nœud du NoC RF. De par l'utilisation d'un algorithme distribué, chaque nœud du réseau doit transmettre ses besoins à tous les autres nœuds avant que l'algorithme ne puisse s'exécuter. Or, grâce à la capacité de broadcast de WiNoCoD, envoyer ces informations à tous les nœuds du NoC RF ne coûte pas plus cher que les envoyer à un unique nœud.

L'algorithme commence par prendre en compte les besoins des différents nœuds du réseau. Ces besoins correspondent au pourcentage de remplissage de la FIFO d'émission de l'arbitre de chaque nœud. On note B_i^t, les besoins d'un nœud i du réseau durant le cycle RF t. Chaque nœud calcule ensuite S^t, la somme des besoins des N nœuds du réseau représentée par l'équation (4.1). Il calcule A_i^t, le nombre de

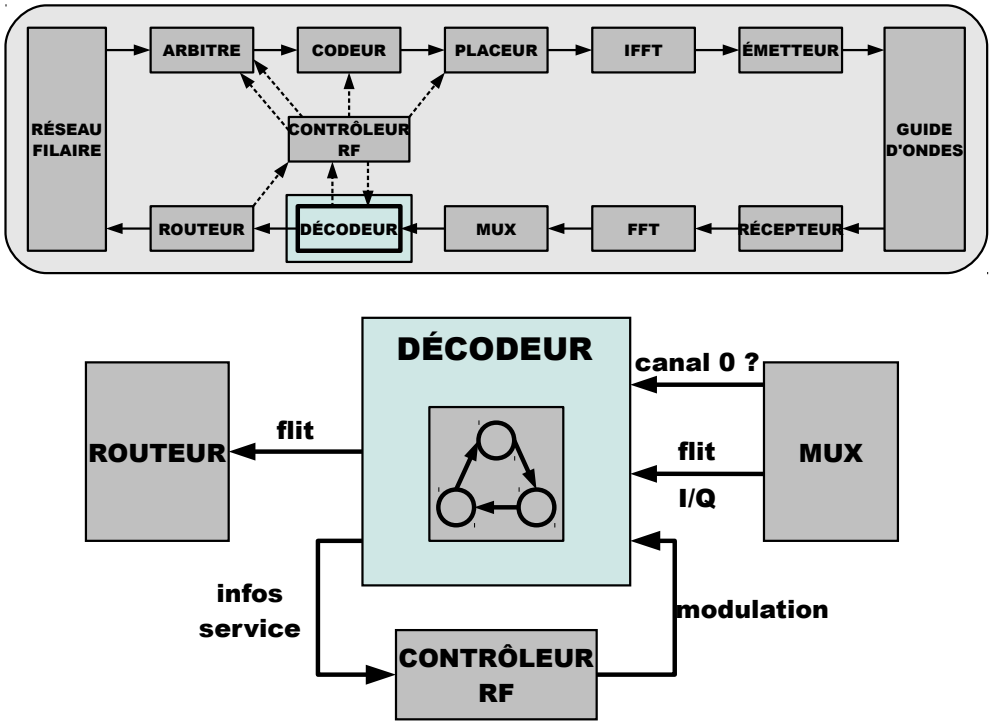

Figure 4.12 – Décodeur

bandes à allouer à un nœud i grâce à l'équation (4.2). Ce calcul prend en compte M, le nombre total de bandes disponibles dans le système. Comme A_i^t est arrondi à la valeur inférieure, il est possible qu'une fois tous les A_i^t calculés, il reste des bandes non allouées. Ces bandes restantes sont alors allouées à un ou plusieurs nœuds suivant la politique du tourniquet. Le calcul du nouveau schéma d'allocation se termine avant la fin du cycle RF de manière à pouvoir le mettre en place dés le cycle RF suivant.

$$S^t = \sum_{j=1}^{N} B_j^t \tag{4.1}$$

$$A_i^{t+1} = M \times \frac{B_i^t}{S^t} \tag{4.2}$$

Il est possible d'améliorer cet algorithme, notamment en pondérant les besoins des clusters [Brière *et al.*, 2015]. D'une part, avec une prévision du trafic reçu entre le moment où ils sont transmis et le moment où l'allocation sera effective. Et d'autre part, avec une estimation des données qui auront été transmises en fonction du précédant schéma d'allocation des bandes.

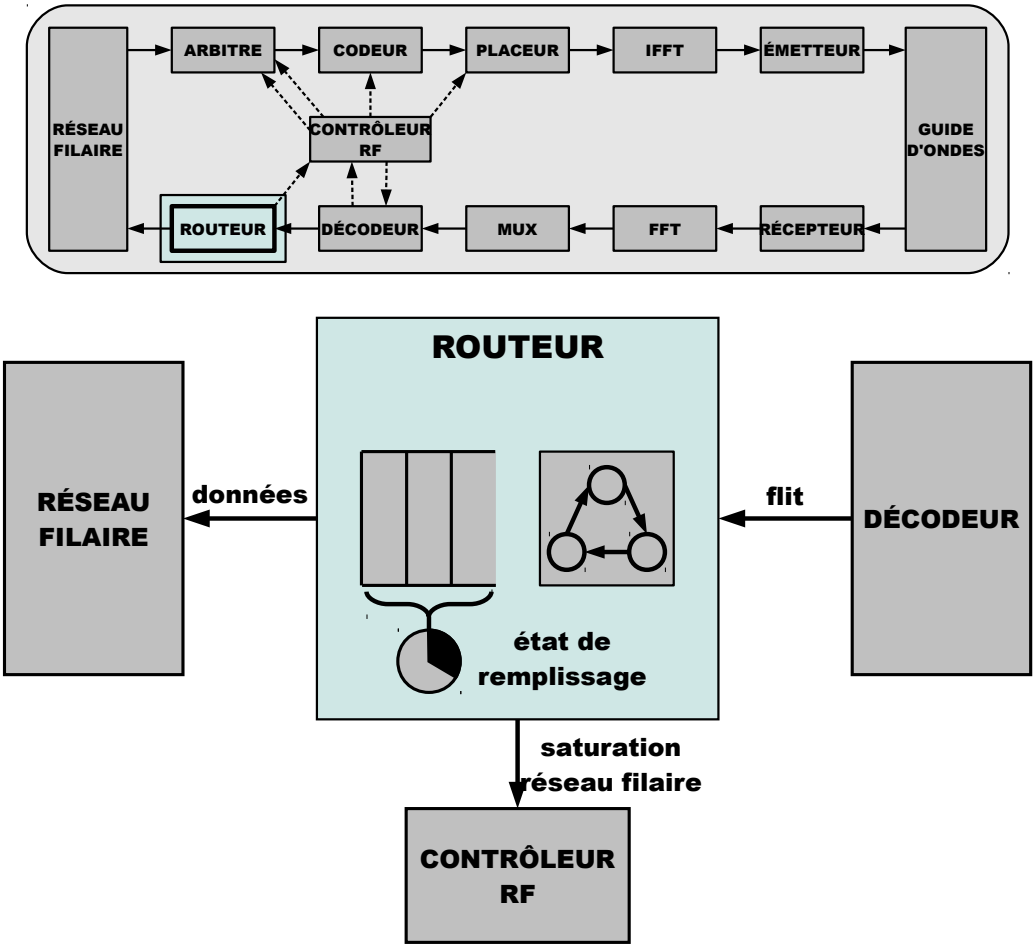

Figure 4.13 – Routeur

4.4.4 Le contrôleur RF : support matériel de l'allocation dynamique

Le contrôleur RF, représenté sur la Figure 4.14, est chargé de mettre en place les mécanismes permettant de fournir les fonctionnalités suivantes :

— Allocation dynamique des canaux de communication
— Adaptation de la modulation
— Contrôle de flux

4.4.4.1 Allocation dynamique des canaux de communication

L'allocation des sous-porteuses est faite avec comme granularité le symbole OFDM. Ainsi au début de chaque symbole OFDM, chaque nœud utilise une portion du canal de service pour émettre l'état de remplissage de la FIFO d'émission de son arbitre et de la FIFO de réception de son routeur. Une nouvelle fois, grâce à la capacité de broadcast de WiNoCoD, les informations de service sont envoyées en même temps à tous les nœuds du réseau. Il n'y a donc pas besoin d'envoyer spécifiquement ces informations à chaque nœud, ainsi le canal de service n'a besoin que d'un minimum de sous-bandes.

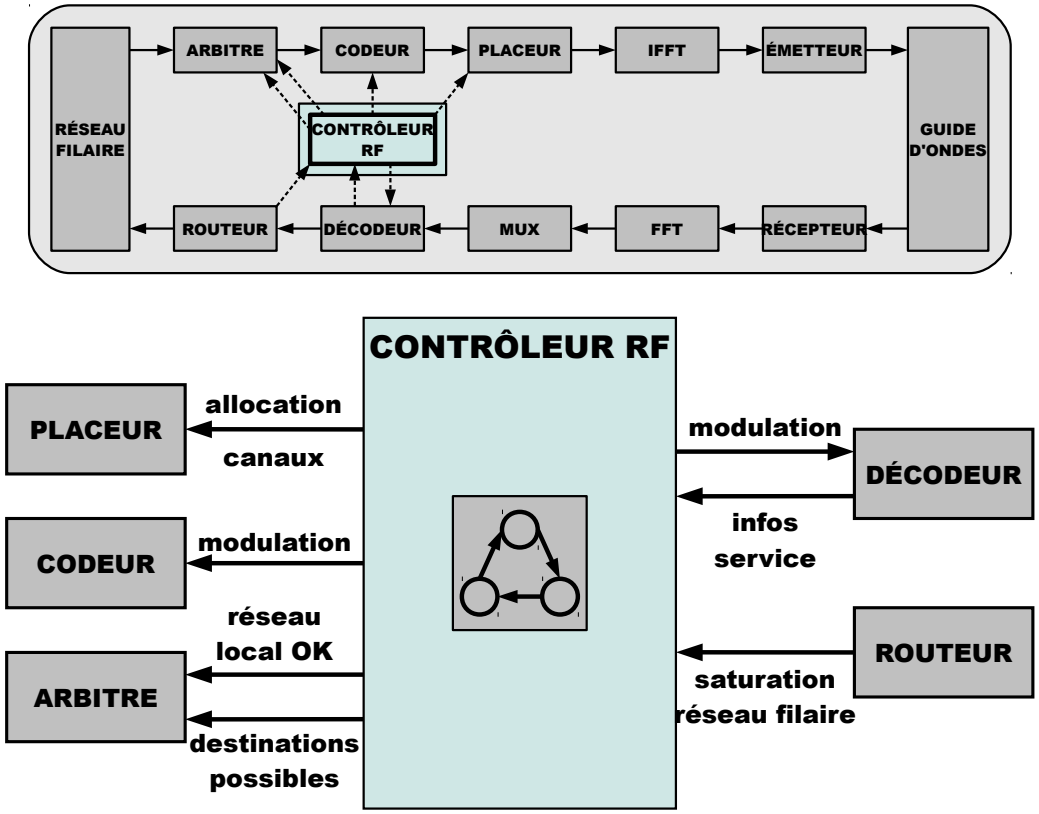

Figure 4.14 – Contrôleur RF

Le chronogramme de la Figure 4.15 représente le déroulement de cet algorithme d'allocation. Lors de la réception de chaque symbole OFDM, le décodeur de chaque interface RF commence par traiter les sous-porteuses constituant le canal de service. Ces informations de service sont ensuite transmises au contrôleur RF du nœud. C'est la réception de ces données qui déclenche l'exécution de l'algorithme d'allocation dynamique réparti dans le contrôleur RF de chaque nœud du NoC RF. Chaque contrôleur RF indique ensuite au placeur de son nœud les sous-porteuses à utiliser. Au cycle suivant, le placeur peut alors répartir les données à émettre sur les entrées de la IFFT correspondant aux sous-porteuses attribuées au nœud dans le nouveau schéma d'allocation.

4.4.4.2 Adaptation de la modulation

Une autre façon d'augmenter la bande passante allouée à un nœud est d'augmenter l'efficacité spectrale en changeant de type de modulation (de BPSK à QPSK par exemple). Ainsi, chaque symbole représentant un nombre plus important de bits, on augmente la quantité de données qu'un nœud peut envoyer sans lui attribuer plus de sous-porteuses. Le contrôleur RF peut donc décider de changer de modulation s'il détecte que toutes les FIFO d'émission se remplissent de plus en plus malgré l'allocation dynamique des canaux de communication. La nouvelle modulation est

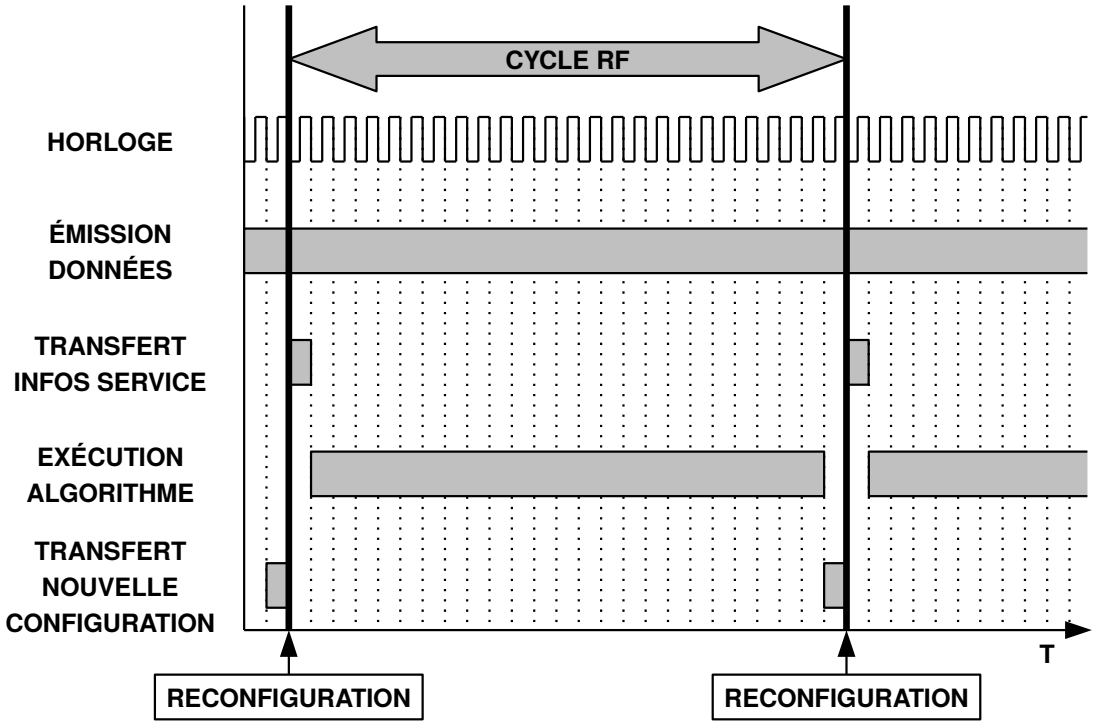

Figure 4.15 – Chronogramme du schéma d'allocation du contrôleur RF

alors transmise au codeur et au décodeur.

4.4.4.3 Contrôle de flux

L'autre fonction du contrôleur RF qui va de pair avec l'allocation dynamique des ressources de communication est de prévenir la perte de paquet. Ainsi, si le réseau filaire relié au routeur de l'interface RF d'un nœud X sature, il faut pouvoir en informer les autres nœuds pour qu'ils n'envoient plus de paquets vers ce nœud X.

Pour cela, le routeur du nœud X indique au contrôleur RF quand sa FIFO de réception dépasse un certain pourcentage de remplissage correspondant au seuil de saturation. Ce seuil est fixé de manière à pouvoir continuer de recevoir des paquets pendant le temps nécessaire à l'arrêt de l'émission de données à destination du nœud X. Une fois que le contrôleur RF reçoit ce signal, il transmet l'information à l'arbitre. Le fait que le nœud X n'est plus en mesure de recevoir de nouvelles données est alors ajouté par l'arbitre aux informations de service. Ces données transmises sur le canal de service permettent donc de transmettre la saturation d'un nœud en réception en plus de ses besoins en émission. Ainsi quand les autres interfaces RF du réseau traitent les informations reçues à travers les canaux de service, ils sont en mesure d'indiquer à l'arbitre d'arrêter d'émettre à destination de tel ou tel nœud. Le message de saturation émis par le nœud X est maintenu jusqu'à ce que la FIFO de réception de son routeur soit repassée en dessous du seuil de saturation. Ce n'est qu'après que les nœuds du réseau pourront recommencer à émettre vers le nœud X.

4.5 Conclusion

Dans ce chapitre nous avons présenté le NoC RF WiNoCoD. Ce réseau sur puce a été proposé dans le but de répondre à la problématique identifiée dans le chapitre 2 et raffinée dans le chapitre 3 :
Comment allouer dynamiquement les ressources de communication dans un NoC de manière cohérente avec l'amélioration de la latence et du débit ainsi que le support du broadcast ?

La mise en place efficace de cette allocation dynamique repose sur deux principes. D'une part, pouvoir diviser la bande passante totale du NoC en sous-bandes répartissables entre les différents nœuds du réseau. D'autre part, disposer d'un mécanisme de reconfiguration permettant d'allouer dynamiquement ces sous-bandes en fonction des besoins des nœuds du réseau. De plus, nous avons vu que le broadcast était une caractéristique intrinsèque de WiNoCoD. Il permet à chaque nœud de communiquer ses besoins au reste du réseau et donc d'utiliser un algorithme de reconfiguration dynamique distribué. Ce n'est cependant pas son seul intérêt puisque les communications de type broadcast sont utilisées dans les protocoles de cohérence de cache.

Dans le chapitre suivant, nous présenterons la modélisation de l'architecture WiNoCoD. Nous décrirons d'abord l'environnement utilisé pour réaliser ce modèle. Nous présenterons ensuite le modèle en lui même ainsi que les choix ayant été faits de manière à pouvoir simuler efficacement l'architecture tout en restant le plus proche possible de la réalité.

Chapitre 5

Modélisation SystemC de l'architecture

Sommaire

5.1 Introduction

Dans le chapitre précédent, nous avons présenté WiNoCoD, un réseau sur puce utilisant la RF et ayant une organisation hiérarchique, les principes de ce NoC RF et les détails de son architecture. Les deux principales caractéristiques de ce réseau sont de pouvoir allouer dynamiquement ses ressources de communication et de pouvoir faire du broadcast sans surcoût par rapport aux communications point à point.

Dans ce chapitre, nous présentons le modèle mis en place afin d'évaluer l'architecture WiNoCoD. La section 5.2 présente l'environnement et les outils utilisés pour modéliser l'architecture proposée. La section 5.3 présente l'implémentation du modèle de l'architecture WiNoCoD.

5.2 Outils de modélisation

Afin de pouvoir étudier les caractéristiques d'une architecture, il est utile d'en établir un modèle. Pour cela, on utilise un *Hardware Description Language* (HDL) permettant de modéliser dans un formalisme donné le fonctionnement de l'architecture. Les HDL les plus couramment utilisés sont VHDL [Shahdad *et al.*, 1985], Verilog [Thomas et Moorby, 1991], SystemVerilog [Rich, 2003] et SystemC [Panda, 2001], même si ce dernier n'est pas un langage à proprement parler.

De plus, les différents HDL permettent de modéliser une architecture avec différents niveaux d'abstraction en fonction des caractéristiques que l'on cherche à évaluer, et de l'avancement dans la conception de l'architecture étudiée. Ainsi, il est possible d'aller du plus haut niveau d'abstraction, avec par exemple une modélisation du système basée uniquement sur ses spécifications, au plus bas niveau avec la description en portes logiques des composants.

Le tableau 5.1 liste les différents niveaux de modélisation en les associant aux HDL dans lesquels ils sont utilisables. On peut ainsi constater que le HDL permettant d'avoir l'abstraction de plus haut niveau et la modélisation la plus globale d'une architecture est SystemC. C'est le HDL que nous avons retenu pour modéliser l'architecture WiNoCoD.

5.2.1 SystemC

À la fin des année 90, la complexification des architectures développées a fait ressentir le besoin d'utiliser des HDL ayant un plus haut niveau d'abstraction. Les modèles comportementaux et RTL utilisés en Verilog et VHDL devenaient trop long à réaliser et à simuler pour être utilisés dés les premières phases de développement d'une architecture. De plus, les concepteurs d'architectures matérielles ont cherché

	VHDL	Verilog	SystemVerilog	SystemC
Modèle système	✗	✗	✗	✓
Modèle fonctionnel	✗	✗	✗	✓
TLM	✗	✗	✓	✓
CABA	✓	✓	✓	✓
RTL	✓	✓	✓	✗
Porte logique	✓	✓	✓	✗

Tableau 5.1 – Niveaux d'abstractions des principaux HDL

à pouvoir co-simuler le matériel et le logiciel. D'une part pour pouvoir développer au plus tôt le logiciel s'exécutant sur la future architecture, et d'autre part, pour pouvoir évaluer les performances de celle-ci. SystemC a donc été développé dans le but de proposer un HDL permettant de modéliser une architecture au niveau système et de pouvoir co-simuler le matériel et le logiciel.

SystemC n'est pas à proprement parler un langage, mais un ensemble de classes C++ et un simulateur. Étant basé sur C++, SystemC permet d'utiliser le même langage et les mêmes outils pour modéliser le matériel et pour écrire le logiciel. SystemC repose sur une représentation structurelle des architectures matérielles, des types de données et une représentation adéquat du temps et de la concurrence.

Comme les autres HDL, SystemC permet de décrire des modules réalisant des fonctions de base et de les assembler pour former des modules de plus en plus complexes. Ces modules communiquent via des signaux pouvant être de plus en plus complexes suivant le niveau d'abstraction. Un signal peut ainsi être un bit, un vecteur de bits ou n'importe quel objet possédant les opérateurs d'affectation et de comparaison. Ces modules et les signaux qui leur permettent de communiquer utilisent des types spécifiques à SystemC en plus des types de C++.

Les types spécifiques à SystemC permettent de représenter le fonctionnement des composants matériels avec différents niveaux de précision. Au plus bas niveau, il est possible d'utiliser un `sc_bit` permettant de modéliser la valeur `0` ou `1` d'un bit ou un `sc_logic` y ajoutant un état "haute impédance" et un état "indéfini". Ces `sc_bit` et `sc_logic` peuvent être utilisés sous forme de vecteur pour simplifier leur manipulation, `sc_bv` et `sc_lv` respectivement. À plus haut niveau, il est possible d'utiliser des types tels que `sc_int` pour représenter les entiers ou encore `sc_fixed` pour représenter les nombres en virgule fixe. Le type `sc_int` et les types qui en dérivent permettent de représenter les entiers, signés ou non, sur un nombre de bits défini par l'utilisateur. Le type `sc_fixed` et les types qui en dérivent permettent de représenter les nombres à virgule fixe, signés ou non, en permettant à l'utilisateur de fixer le nombre de bits de la partie entière et de la partie décimale. De plus, il est possible de définir des types plus complexes permettant par

exemple de représenter l'interface utilisée pour connecter un composant à un bus.

Finalement, SystemC prend en charge la modélisation de la concurrence et du temps nécessaires à la description des architectures matérielles. La modélisation du temps repose sur le type `sc_time` pour représenter le temps lui-même et le type `sc_clock` pour générer des horloges. La modélisation de la concurrence repose sur l'utilisation de processus. Un module pouvant ainsi avoir plusieurs processus représentant différentes tâches s'exécutant en parallèle. Ainsi, les modèles du temps et de la concurrence permettent à SystemC de représenter le fonctionnement du matériel grâce à son simulateur à évènements discrets.

5.2.2 SoClib

SystemC a été développé dans le but de faciliter la modélisation d'une architecture au niveau système. Ce niveau d'abstraction implique de construire les architectures à partir de blocs de base correspondant aux divers composants matériels. On parle alors d'*Intellectual Property core* (IP core). C'est pour répondre à ce besoin qu'a été créée la bibliothèque d'IP core SoClib [SoCLib Consortium, 2003] dans le cadre du projet ANR éponyme.

Tous les IP core de SoClib sont décrits grâce à SystemC. Dans cet environnement de prototypage virtuel, il est possible de décrire des composants avec deux niveaux de précision, CABA (Cycle Accurate Bit Accurate) et TLM (Transaction Level Modeling). Dans un modèle CABA, les composants sont décrits de manière précise, on connait le comportement cycle après cycle des composants, et ce au bit près. Dans un modèle TLM, les communications et le fonctionnement interne des composants sont modélisés de manière séparée. Dans un soucis de précision, le modèle du *Network on Chip* (NoC) RF WiNoCoD a été réalisé en CABA.

De plus, soit les IP core de SoClib respectent le protocole *Virtual Component Interface* (VCI), soit elles sont accompagnées d'une IP core permettant de faire la conversion entre VCI et un protocole tiers. VCI est une sous-partie de la norme *Open Core Protocol* (OCP). Tout comme SystemC, OCP est un standard défini par Accellera. Le protocole de communication VCI est basé sur des transactions entre initiateurs et cibles. Un initiateur émet un paquet de commande à destination d'une cible identifiée par les bits de poids fort de l'adresse. La cible émet ensuite un paquet de réponse à destination de l'initiateur identifié par son index. Ces paquets de commande et de réponse peuvent être composés d'un ou plusieurs *FLow control unIT* (FLIT) en fonction du type de transaction. Un paquet de commande pour une lecture sera par exemple composé d'un seul FLIT alors que le paquet de réponse sera composé d'autant de FLIT que de mots demandés par la requête de lecture. Parmi les différents champs composant un paquet VCI, les champs `val` et `ack` permettent de mettre en place un mécanisme d'acquittement pour s'assurer de la réception des paquets.

Le modèle du NoC RF WiNoCoD offre donc une interface VCI permettant son intégration dans SoClib. Cependant, le protocole VCI n'est utilisé que pour interfacer le NoC RF avec le réseau filaire. Les données sont ensuite transmises entre les différents blocs du NoC RF suivant un protocole qui lui est propre.

Le respect strict du protocole VCI n'est cependant pas obligatoire. Une version de WiNoCoD interfaçable avec le réseau *Distributed Scalable Programmable Interconnection Network* (DSPIN) [Miro-Panades *et al.*, 2006] a ainsi été développée. DSPIN est un réseau de routeurs permettant de créer une grille 2D et reposant sur le protocole du même nom.

Il est donc tout à fait envisageable d'adapter WiNoCoD à d'autres protocoles pour pouvoir l'interfacer avec d'autres bibliothèques d'IP core que SoClib. Des travaux ont d'ailleurs été réalisés dans ce sens pour interfacer le générateur de trafic NoC-Bench [Pekkarinen *et al.*, 2011] avec WiNoCoD, ou tout autre réseau présent dans SoClib [Azeem *et al.*, 2015].

5.3 Modélisation de l'architecture

Nous allons maintenant présenter les détails d'implémentation des différents composants de l'architecture WiNoCoD.

Le développement des différents modèles présentés dans ce chapitre a pour but la simulation de l'ensemble du système. De plus, le but de cette simulation est d'évaluer l'apport de la reconfiguration dynamique. Nous avons donc simplifié le modèle en abstrayant certains composants de l'architecture n'entrant pas directement en jeu dans les mécanismes de reconfiguration dynamique.

Les canaux de communications constituent la ressource allouée par le contrôleur RF et son algorithme d'allocation dynamique. Comme ces canaux de communication sont visibles dés la sortie du placeur et jusqu'à l'entrée du multiplexeur, il n'est pas nécessaire de modéliser la IFFT et la FFT pour évaluer l'apport de la reconfiguration dynamique du NoC RF. Il n'est pas non plus nécessaire de modéliser les transmetteurs et récepteurs RF puisqu'ils n'interviennent pas non plus dans les mécanismes d'allocation dynamique. Comme le montre la Figure 5.1, nous avons donc intégré les émetteurs/récepteurs et IFFT/FFT à un composant que nous avons appelé « guide d'ondes numérique ».

De plus, pour que les grappes de notre architecture hiérarchique puissent être reliées au NoC RF, il a été nécessaire de modifier les routeurs DSPIN. Nous avons ainsi développé une version 3D des routeurs avec un routage adapté à la configuration hiérarchique de WiNoCoD.

Dans la suite de ce chapitre, les différentes valeurs numériques indiquées correspondront à la configuration suivante du NoC RF :

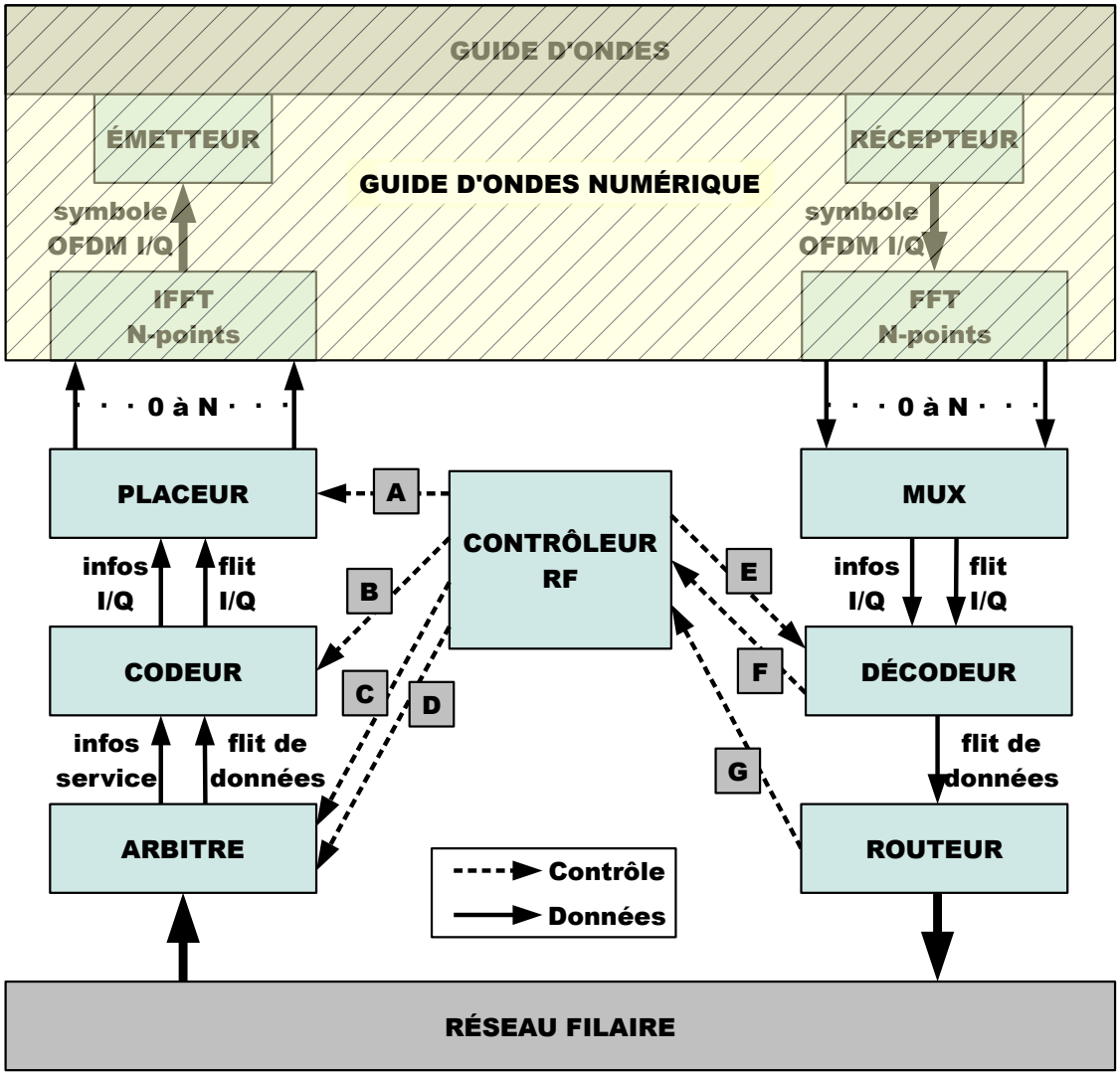

Figure 5.1 – Modèle de l'interface RF

— 16 nœuds
— modulation 16-QAM
— 512 sous-bandes
— 16 canaux de 30 sous-bandes chacun pour les données
— 1 canal de service de 32 sous-bandes

5.3.1 Le routeur 3D

La Figure 5.2 représente le modèle du routeur 3D utilisé pour relier le réseau filaire au réseau RF. Ce routeur est une extension 3D du routeur DSPIN proposé par Miro-Panades *et al.* [2006]. En plus des cinq ports d'origine (Nord, Sud, Est, Ouest et Local), il possède deux ports verticaux (Haut et Bas) lui permettant d'être relié aux couches inférieures et supérieures. Cependant, nous n'avons besoin que d'une connexion supplémentaire pour relier le réseau filaire à la RF. Nous n'utilisons donc

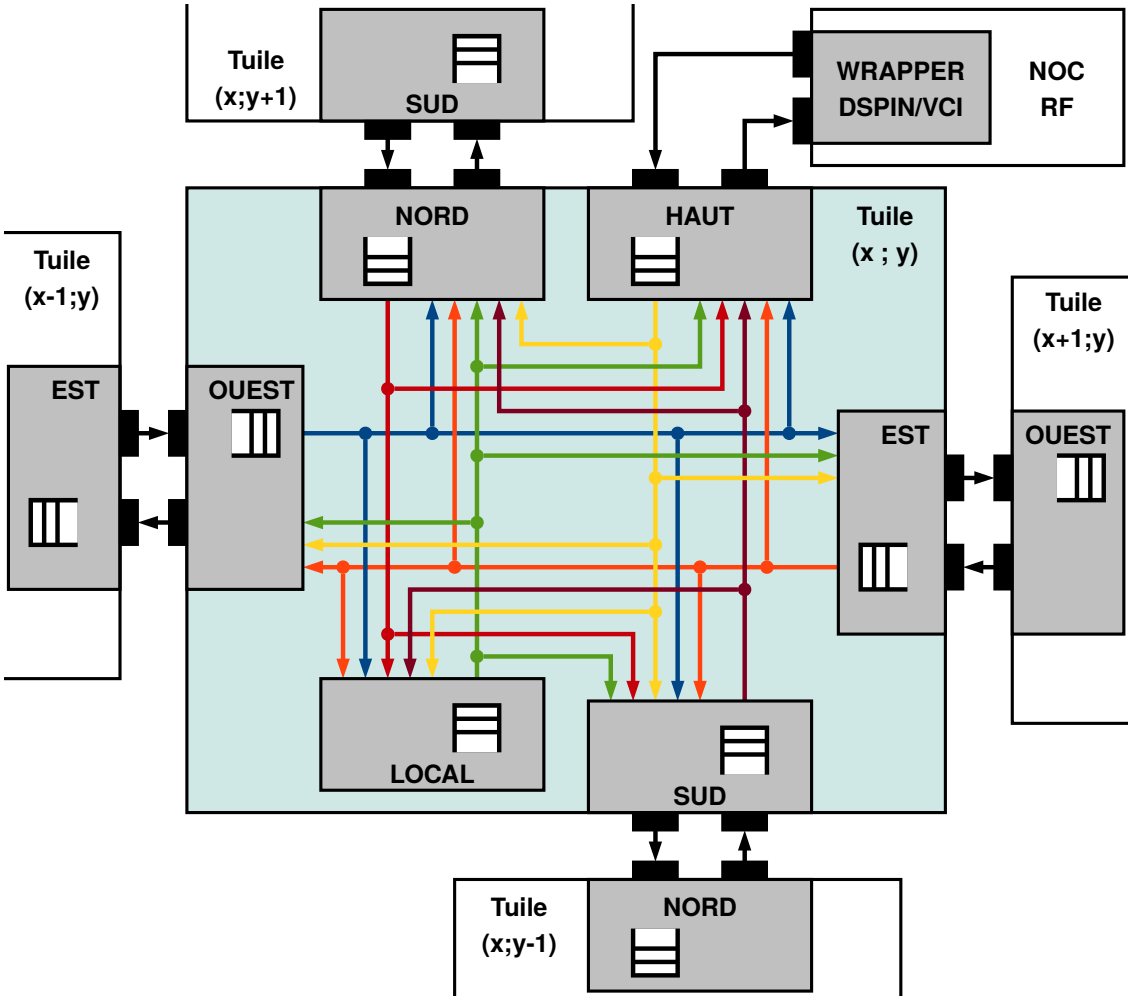

Figure 5.2 – Modèle du routeur 3D

que la connexion vers les haut alors que la connexion vers le bas est désactivée et n'est pas représentée sur la figure pour plus de simplicité.

Le routage des paquets se base sur les bits de poids fort de l'adresse regroupés en trois champs X, Y et Z. Ces champs permettent de déterminer le routeur de destination d'un paquet et ainsi de savoir s'il se trouve dans le même plan. Dans le cadre de WiNoCoD, l'appartenance de la destination à un autre plan correspond à l'utilisation du réseau RF. Le ou les routeurs centraux d'une grappe sont des routeurs 3D alors que les autres sont des routeurs 2D. Ces derniers sont modifiés pour être capables d'analyser le champ Z de l'adresse de manière à router les paquets vers le routeur 3D de la grappe s'ils doivent utiliser le réseau RF.

5.3.2 L'arbitre

La Figure 5.3 représente le modèle de l'arbitre. Le rôle de ce composant est de traiter les données provenant du réseau filaire pour les transmettre sur le NoC RF. L'arbitre est composé de deux *Machines À États* (MAE).

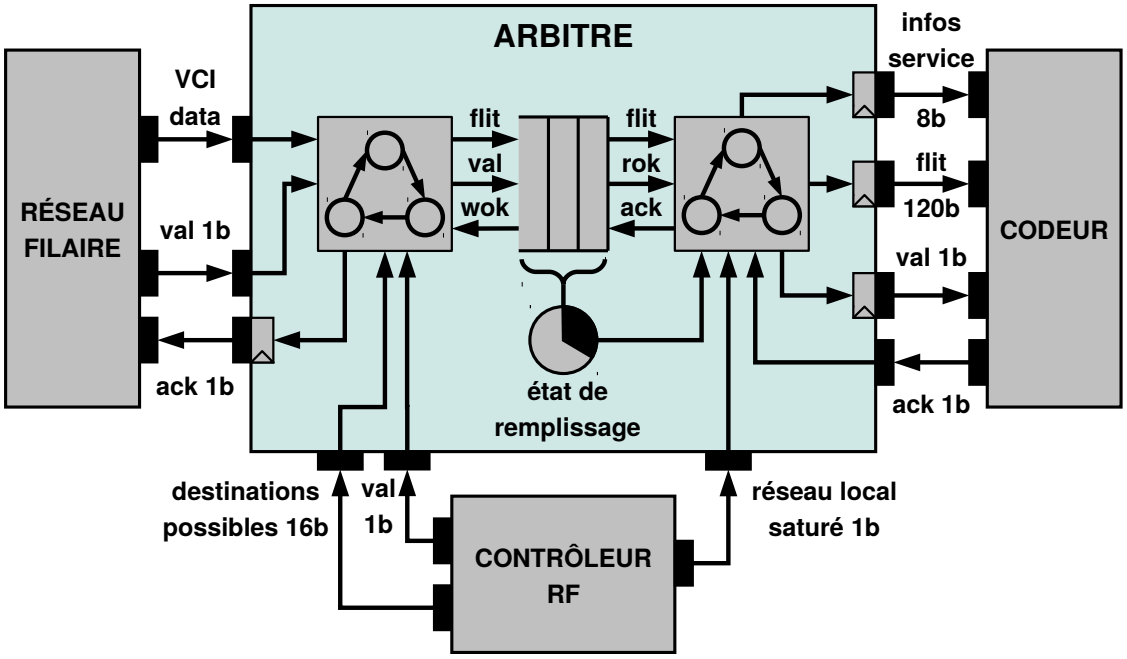

Figure 5.3 – Modèle de l'arbitre

La première MAE a pour but de gérer l'interfaçage avec le réseau filaire. Les données circulant sur le réseau filaire utilisent la norme VCI. L'arbitre doit donc respecter lui aussi cette norme et plus particulièrement le mécanisme d'acquittement utilisé par celle-ci. Parmi les différents champs de données présents dans une interface VCI, ce sont les signaux `val` et `ack` qui permettent de mettre en place ce mécanisme d'acquittement. Le réseau filaire qui émet un paquet positionne son signal `val` à `1` pour indiquer qu'il présente un nouveau paquet valide sur son port de sortie. L'arbitre qui reçoit le paquet peut alors lire les données et positionner son signal `ack` à `1` pour l'indiquer au réseau filaire. Le contrôleur RF transmet à ce premier automate la liste des nœuds du NoC RF en état de recevoir de nouvelles données. Un changement de cette liste est indiqué par le positionnement du signal `val` du contrôleur RF à `1`. L'arbitre analyse les paquets provenant du réseau filaire pour connaitre leur destination. Il peut ainsi bloquer un paquet tant que sa destination n'est pas prête à le recevoir. Ensuite, l'arbitre fusionne les différents champs VCI pour créer un FLIT RF.

Les FLIT RF sont stockés dans la FIFO de l'arbitre. L'écriture dans cette FIFO par la première MAE et la lecture par la deuxième utilisent elles aussi un mécanisme d'acquittement. La FIFO indique à la première MAE quand elle peut recevoir de nouvelles données en mettant son signal `wok` à `1`. La première MAE déclenche alors l'écriture d'un FLIT en positionnant son signal `val` à `1`. Quand des FLIT sont disponibles dans la FIFO, celle-ci l'indique à la deuxième MAE en mettant son signal `rok` à `1`. La deuxième MAE peut alors lire un FLIT et indiquer qu'il a bien été consommé en renvoyant un signal `ack` à la FIFO.

De plus, comme nous l'avons vu dans le chapitre précédant, cette FIFO est une pièce centrale de l'allocation dynamique des ressources de communication. Son état de remplissage est transmis à la deuxième MAE de l'arbitre parmi les informations de service. La deuxième composante de ces informations de service étant la disponibilité du réseau filaire local côté réception. Cette information provient du routeur et est transmise par le contrôleur RF.

La deuxième MAE de l'arbitre doit donc transmettre au codeur les données sous la forme de FLIT ainsi que les informations de service. L'interface entre l'arbitre et le codeur repose elle aussi sur un mécanisme d'acquittement. Ainsi la deuxième MAE de l'arbitre stocke dans les registres correspondant à ses ports de sortie les informations de service, un FLIT ainsi qu'un signal `val` indiquant au codeur la présence de nouvelles données. Le codeur peut alors lire ces données selon sa disponibilité et informer l'arbitre une fois cette lecture terminée en positionnant son signal `ack` à `1`.

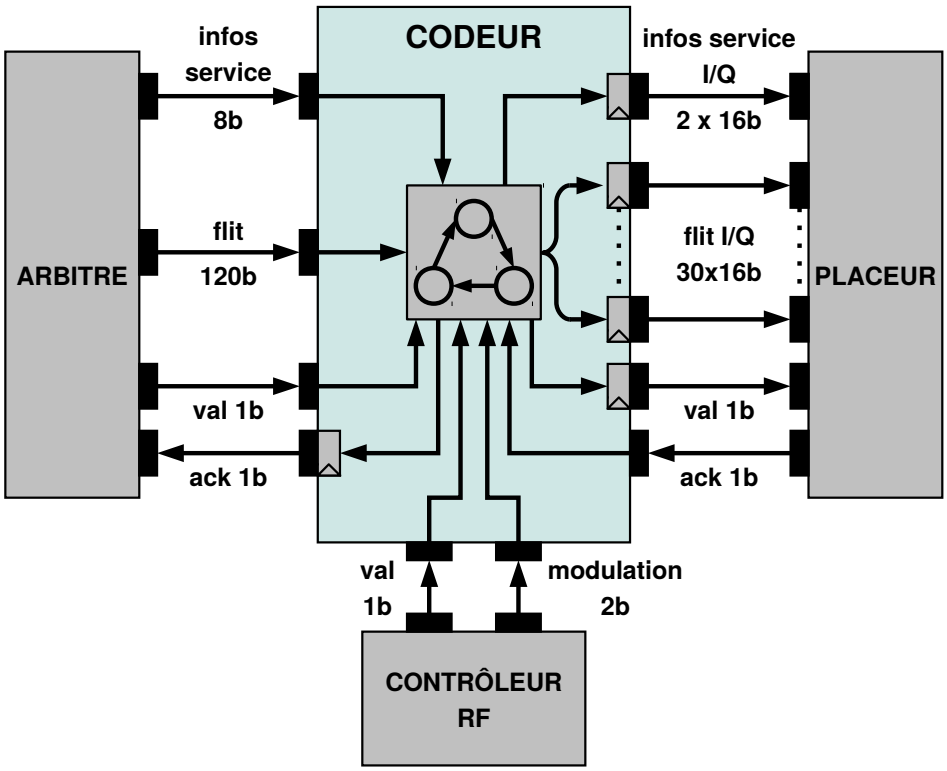

Figure 5.4 – Modèle du codeur

5.3.3 Le codeur

La Figure 5.4 représente le modèle du codeur. Le codeur reçoit de l'arbitre les FLIT de données sur 120 bits et les informations de service sur 8 bits. Les 120 bits correspondent au nombre de bits total des différents champs constituant un FLIT VCI. Les 8 bits sont induits par la modulation utilisée et le nombre de bandes du canal

de service allouées par nœud. Le codeur effectue ensuite la modulation numérique de ces données. La modulation à utiliser est indiquée par le contrôleur RF via le signal `modulation`. Ce signal sur 2 bits permet de choisir entre les modulations BPSK, QPSK et 16-QAM. Une fois modulés, le FLIT et les informations de service sont placés dans le registre correspondant aux ports de sortie du codeur. Dans la configuration retenue, la modulation 16-QAM, un FLIT est modulé sur une seule séquence de 30 symboles I/Q. Il faudrait 2 séquences si le codeur utilisait une modulation QPSK et 4 avec une modulation BPSK. La synchronisation avec le placeur est garantie par le même mécanisme d'acquittement qu'avec l'arbitre utilisant les signaux `val` et `ack`. Ainsi, en fonction de la modulation utilisée, le placeur peut lire les données de l'arbitre à un rythme différent de celui auquel il écrit vers le placeur.

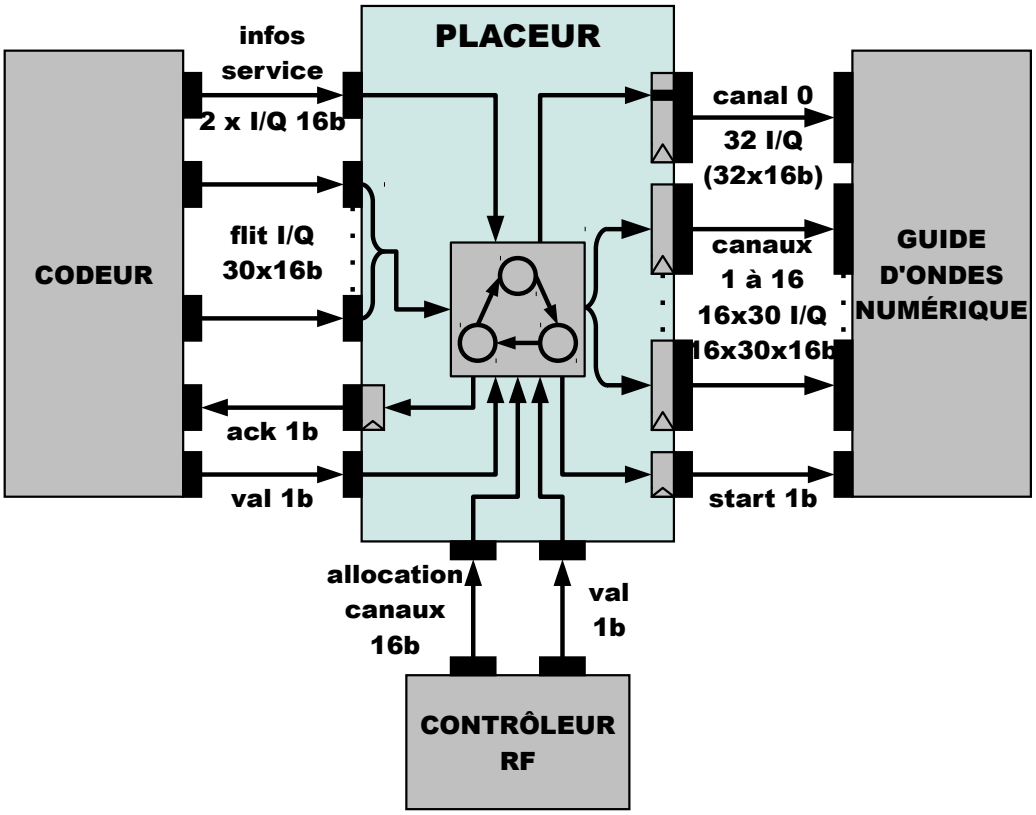

Figure 5.5 – Modèle du placeur

5.3.4 Le placeur

La Figure 5.5 représente le modèle du placeur. Le placeur est chargé de transmettre les FLIT provenant du codeur sur les canaux alloués au nœud RF. C'est le contrôleur RF qui indique au placeur les canaux à utiliser via le signal `allocation canaux`. Ce signal utilise un codage one hot, il a donc besoin d'autant de bits qu'il y a de canaux à allouer, 16 dans la configuration présentée. Il est important de noter qu'en fonction de la modulation utilisée, les canaux peuvent être regroupés afin de trans-

mettre un FLIT RF complet à chaque cycle du NoC RF. Ainsi, les canaux seraient regroupés par deux avec la modulation QPSK et par quatre avec la modulation BPSK. De plus, il place les informations de service sur la portion du canal de service correspondant à l'indice du nœud dont il dépend. Pour simplifier la modélisation et améliorer la vitesse de simulation, le placeur est directement relié au composant « guide d'ondes numérique ». En l'absence d'IFFT pour fixer la période d'un symbole OFDM, c'est donc au placeur de définir cette période. Pour ce faire, il dispose d'un signal `start` lui permettant d'indiquer quand ses sorties sont valides et de copier le fonctionnement périodique de l'IFFT. Ce signal est mis à `1` tout les N cycles d'horloge. Pour mettre en place la configuration retenue, une horloge à 1 GHz et un cycle réseau de 25 ns, il faut donc fixer N à 25.

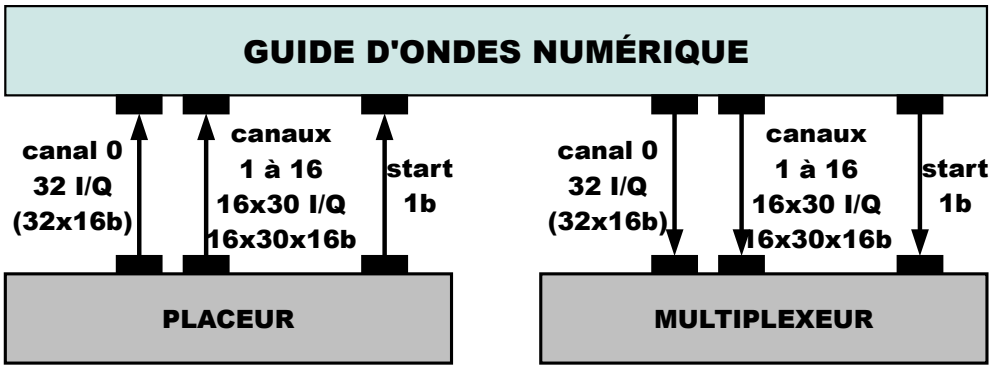

Figure 5.6 – Modèle du guide d'ondes numérique

5.3.5 Le guide d'ondes numérique

La Figure 5.6 représente le modèle du guide d'ondes. L'émetteur et le récepteur RF ainsi que la IFFT et la FFT n'étant pas intégrés au modèle de l'architecture, le guide d'ondes numérique est directement relié au placeur et au multiplexeur de chaque interface RF du réseau.

Pour chaque interface RF du réseau, le guide d'ondes reçoit les différentes sorties du placeur : le canal de service, les 16 canaux transférant les données utiles et le signal `start`. Les informations de service sont placées sur la portion du canal de service attribuée au nœud local. Les données utiles sont placées sur les canaux attribués au nœud local. Toutes les sorties du placeur ne correspondant pas aux canaux qui lui sont attribués sont laissées à zéro. Le guide d'ondes doit donc additionner une à une les sorties de tous les placeurs du réseau pour créer le signal complet. Ce calcul est déclenché par le signal `start` qui est simultanément mis à `1` par tous les placeurs du réseau. Une fois cette opération réalisée, les données sont placées sur les entrées de tous les multiplexeurs du réseau. Le guide d'ondes place alors son propre signal

`start` à `1` pour indiquer la validité de ces données.

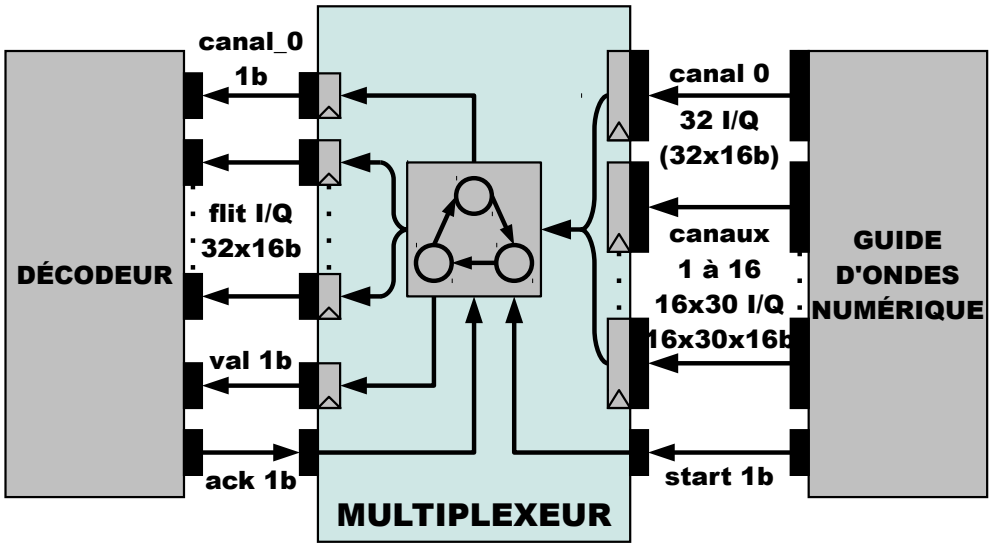

Figure 5.7 – Modèle du multiplexeur

5.3.6 Le multiplexeur

La Figure 5.7 représente le modèle du multiplexeur. Le multiplexeur reçoit donc le canal de service, les canaux transférant les données utiles ainsi qu'un signal `start` lui indiquant l'arrivée de nouvelles données valides. Tout comme pour le placeur, le multiplexeur est directement relié au « guide d'ondes numérique » de manière à simplifier la modélisation et améliorer la vitesse de simulation.

Le multiplexeur transmet le contenu des canaux un par un en se synchronisant avec le décodeur grâce à ses signaux `val` et `ack`. Pour pouvoir lancer l'exécution de l'algorithme d'allocation au plus tôt de chaque cycle RF, le multiplexeur commence par le canal de service. Il utilise alors son signal `canal_0` pour indiquer au décodeur que les données présentées correspondent au canal de service.

5.3.7 Le décodeur

La Figure 5.8 représente le modèle du décodeur. Le décodeur est chargé de réaliser la démodulation numérique des données provenant du guide d'ondes et de reconstituer les FLIT. Le type de modulation à utiliser lui est indiqué via le signal `modulation` par le contrôleur RF.

Le décodeur utilise le signal `canal_0` transmis par le multiplexeur pour déterminer si les données décodées correspondent aux canaux de service ou aux données

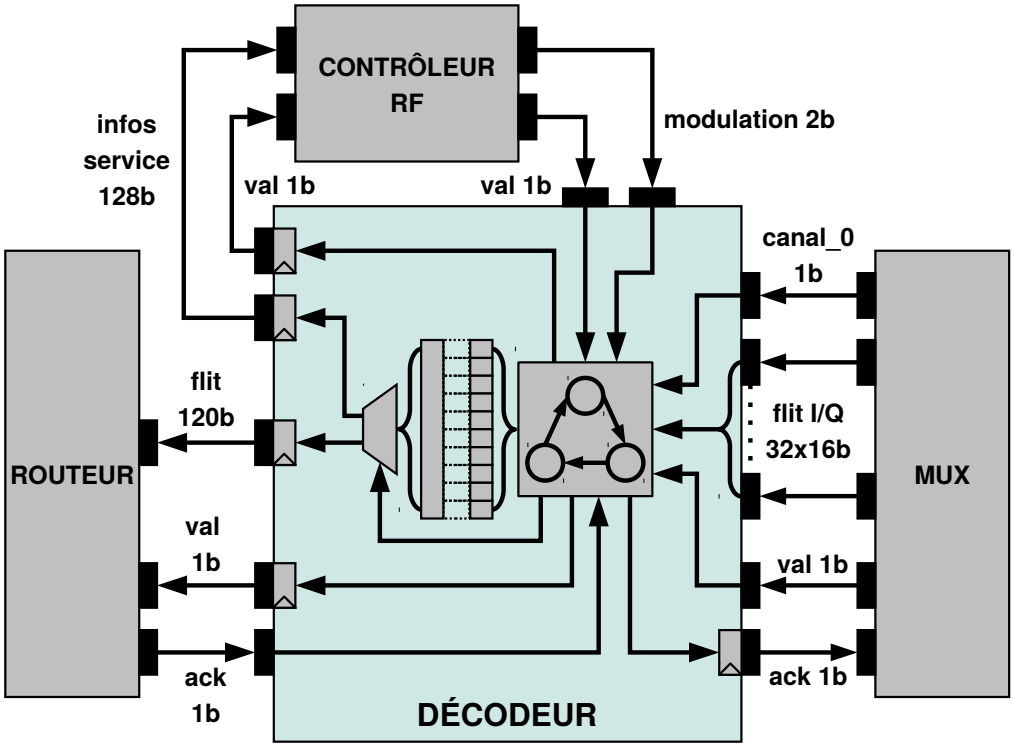

Figure 5.8 – Modèle du décodeur

utiles. Il peut alors les transmettre au contrôleur RF dans le premier cas ou au routeur dans le deuxième. Dans les deux cas, là communication repose là encore sur une synchronisation via les signaux `val` et `ack`.

5.3.8 Le routeur RF

La Figure 5.9 représente le modèle du routeur. Ce composant est chargé de fournir les fonctionnalités suivantes :

— Reconstituer et router les paquets provenant du NoC RF
— Gérer les variations de trafic du NoC RF et la disponibilité du réseau filaire
— Fournir les informations utilisées pour le contrôle de flux amont

5.3.8.1 Reconstitution des paquets

Une fois le FLIT RF reconstitué par le décodeur, il est transmis au routeur. Le routeur peut alors analyser ce FLIT pour savoir s'il doit le transmettre au réseau filaire. C'est le cas des FLIT explicitement adressés à un élément connecté au réseau filaire local et des messages de broadcast. Le routeur commence donc par reconstituer les différents champs d'un FLIT VCI à partir du FLIT RF. Ensuite, si le FLIT n'est pas

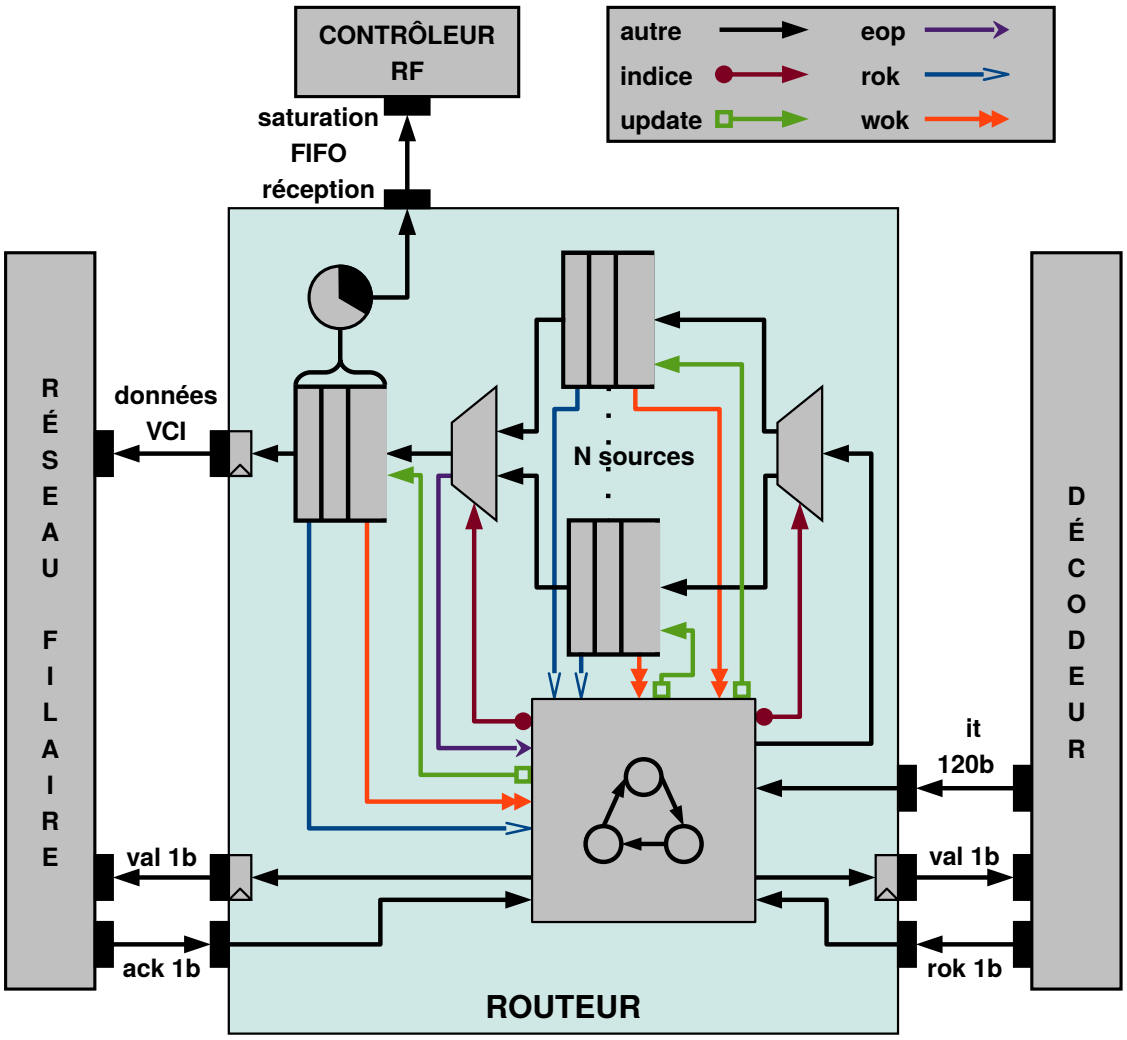

Figure 5.9 – Modèle du routeur

destiné au réseau local, le routeur l'ignore tout simplement et indique au décodeur de transmettre le FLIT RF suivant.

Les communications avec le réseau filaire se font à la granularité d'un paquet VCI. Pour ne pas bloquer inutilement les communications il est donc nécessaire de ne pas commencer la transmission d'un paquet VCI tant que celui-ci n'est pas complet. Pour ce faire, le routeur contient une FIFO par nœud du réseau RF lui permettant de reconstituer les paquets VCI. Chaque FLIT RF contient le numéro du nœud RF dont il provient. Cette information est utilisée pour stocker le FLIT VCI nouvellement créé dans la FIFO correspondant au bon nœud RF. Pour ce faire, la MAE du routeur indique au démultiplexeur via un signal `indice` le numéro du nœud source et donc de la FIFO correspondante. De plus, un FLIT VCI comprend un champ `eop` indiquant si le FLIT est le dernier du paquet ou non. La MAE du routeur peut ainsi savoir quand une FIFO contient au moins un paquet VCI complet. Elle peut donc ainsi selon une politique LRU commencer le transfert d'un paquet VCI d'une des FIFO sources vers la FIFO de réception. La MAE indique au multiplexeur le numéro

de la FIFO source devant transmettre des données dans la FIFO de réception via un deuxième signal `indice`.

5.3.8.2 Lissage du trafic

Toutes ces FIFO transmettent un signal `wok` à la MAE pour indiquer si elles peuvent encore recevoir des données. Elles transmettent aussi un signal `rok`, ce signal n'a pas la même signification pour les FIFO sources et la FIFO de réception. Pour les FIFO sources, ce signal indique si le FLIT en tête de FIFO est le dernier d'un paquet. Pour la FIFO de réception, ce signal indique tout simplement que la FIFO contient au moins un FLIT. La MAE du routeur utilise le signal `rok` de la FIFO de réception pour contrôler le signal `val` transmis au réseau filaire et indiquer la présence de nouvelles données.

5.3.8.3 Contrôle de flux amont

Finalement, tout comme l'état de remplissage de la FIFO d'émission de l'arbitre, l'état de remplissage de la FIFO de réception du routeur est utilisé par le contrôleur RF. Ainsi quand la FIFO de réception du routeur dépasse un certain seuil de remplissage, le routeur indique au contrôleur RF que le réseau filaire est en train de saturer. Cette information est transmise par le contrôleur RF via l'arbitre à l'ensemble du réseau pour que les autres nœuds n'envoient plus de données vers ce nœud.

5.3.9 Le contrôleur RF

La Figure 5.10 représente le modèle du contrôleur RF. Les mécanismes d'allocation dynamique et le contrôle de flux mis en place par le contrôleur RF reposent sur les informations de service. Comme nous l'avons présenté plus tôt, ces informations sont transmises sur le NoC RF par l'arbitre. Elles contiennent l'état de remplissage de la FIFO de l'arbitre ainsi que la saturation de la FIFO de réception du routeur. Dans chaque nœud, cette dernière information est transmise par le routeur au contrôleur RF qui la transmet à son tour à l'arbitre. Dés que ces informations sont reçues par le décodeur, celui-ci les transmet au contrôleur RF et place son signal `val` à `1`, déclenchant ainsi l'exécution du contrôleur RF.

Le contrôleur RF commence par analyser les informations de service indiquant si les différents nœuds peuvent recevoir des données et transmet cette information à l'arbitre. Il exécute ensuite l'algorithme d'allocation dynamique pour attribuer les canaux de communication et choisir la modulation à utiliser. Il indique la modulation à utiliser au codeur et au décodeur via un signal `modulation` sur 2 bits permettant

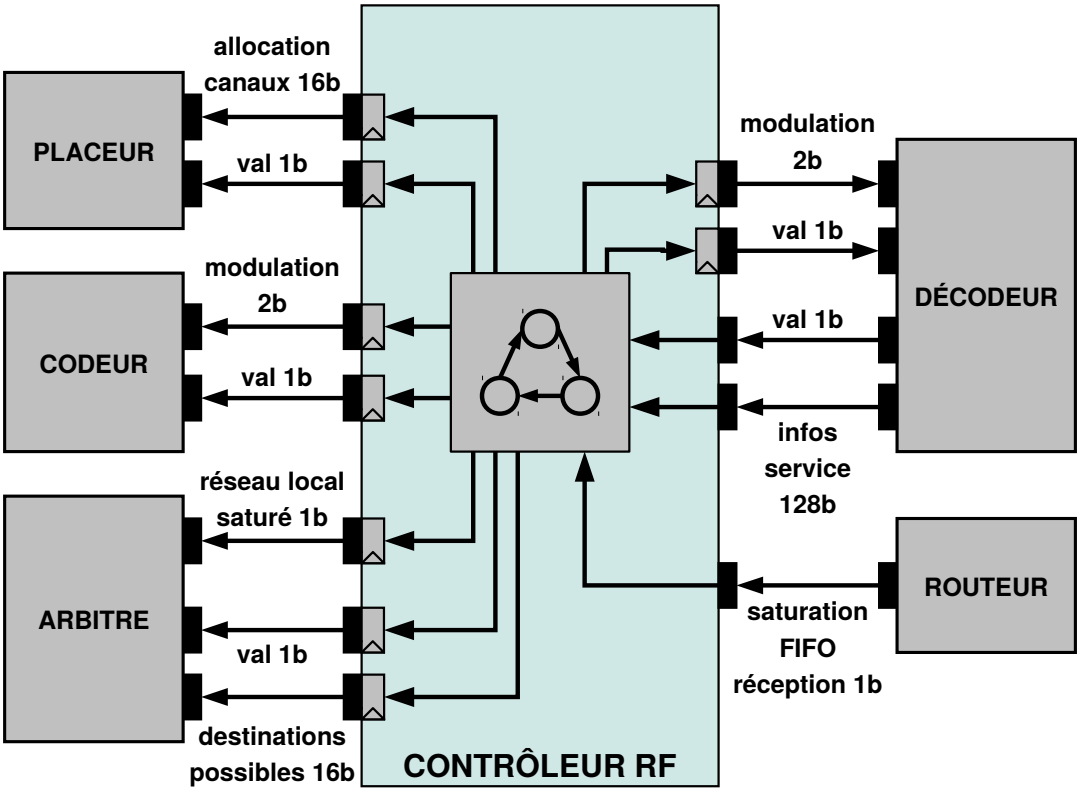

Figure 5.10 – Modèle du contrôleur RF

de choisir parmi les 3 modulations possibles. Il indique la nouvelle configuration des canaux de communication au placeur via un signal `allocation canaux`, ce signal utilise un codage one hot sur 16 bits, soit le nombre maximal de canaux disponibles avec un codage 16 QAM. Ces différents signaux de configuration sont tous accompagnés d'un signal `val` permettant au différents composants de prendre en compte la nouvelle configuration.

5.4 Conclusion

Dans ce chapitre, nous avons présenté l'environnement utilisé pour modélisé l'architecture WiNoCoD ainsi que les détails de l'implémentation de ce modèle. Nous avons utilisé SystemC pour modéliser les composants du NoC RF WiNoCoD. De plus, nous avons respecté les normes utilisées dans la bibliothèque SoClib afin de pouvoir intégrer WiNoCoD dans un *Chip Multiprocessor* (CMP) complet. Nous disposons donc d'un modèle complet de l'architecture capable d'exécuter directement du code et ainsi permettre l'évaluation de la solution proposée.

Dans le chapitre suivant, nous allons présenter les différentes expérimentations réalisées dans le but d'évaluer notre NoC RF.

Chapitre 6

Expérimentations et résultats

Sommaire

6.1 Introduction

Dans le chapitre précédent, nous avons présenté l'environnement utilisé pour modéliser l'architecture *Chip Multiprocessor* (CMP) utilisant le *Network on Chip* (NoC) RF WiNoCoD. Nous avons ensuite détaillé l'implémentation des différents composants du NoC RF et du routeur filaire modifié reliant le réseau filaire au réseau RF. Dans ce chapitre nous allons présenter les différentes expérimentations réalisées afin d'évaluer la faisabilité et les performances du NoC RF WiNoCoD.

La section 6.2 présente une première évaluation des performances de l'architecture d'un point de vue théorique. Ces résultats se basent sur les caractéristiques techniques de l'architecture du NoC RF proposé. Cette section permet d'identifier pour quelles configurations, nombre de nœuds RF et de tuiles, l'usage du NoC RF présente un intérêt.

La section 6.3 présente l'évaluation de la surface et de la consommation de l'architecture du CMP utilisant le NoC RF WiNoCoD. Ces résultats se basent sur l'état de l'art des différents composants utilisés dans le NoC RF et le reste du CMP à travers une étude bibliographique. Cette section permet d'identifier les configurations pour lesquelles la surface et la consommation du NoC RF en font une solution viable.

Les sections suivantes présentent une évaluation plus approfondie des performances de l'architecture et notamment de l'apport le la reconfiguration dynamique. La reconfiguration dynamique se limite cependant à l'allocation dynamique des canaux de communications puisque le type de modulation utilisé ne change pas. Ces résultats se basent sur l'exécution du modèle SystemC simulable de l'architecture WiNoCoD. La section 6.4 présente l'évaluation de l'architecture soumise au trafic généré par l'exécution d'applications sur la plateforme. Les résultats présentés tiennent compte de l'allocation dynamique des ressources de communication. Finalement, la section 6.5 présente l'évaluation de l'architecture soumise à un trafic synthétique.

6.2 Évaluation des performances intrinsèques

Dans cette section, nous présentons l'évaluation des performances intrinsèques de notre NoC RF. Cette évaluation n'est pas basée sur le modèle SystemC, elle utilise uniquement les caractéristiques techniques de l'architecture. Nous évaluons les performances intrinsèques du médium de communication sans tenir compte de l'allocation dynamique des ressources de communication. Cette section permet d'identifier pour quelles configurations, nombre de nœuds RF et de tuiles, l'usage du NoC RF présente un intérêt par rapport à un réseau filaire de type grille 2D.

6.2.1 Protocole expérimental

Pour réaliser cette première évaluation de notre architecture, nous avons tenu compte de la taille du réseau, de sa structure et de ses caractéristiques temporelles. Nous avons comparé les performances de WiNoCoD à celles d'une grille composée de routeurs *Distributed Scalable Programmable Interconnection Network* (DSPIN) [Miro-Panades *et al.*, 2006]. Ces routeurs étant ceux utilisés dans notre modèle SystemC, cela nous permet d'évaluer l'intérêt de superposer notre réseau RF à ce réseau filaire de type grille 2D. Nous avons évalué les performances de ces deux réseaux face à un trafic réparti aléatoirement et ne générant pas de contention.

Pour cette évaluation nous avons retenu une fréquence de fonctionnement de 1 GHz pour les composants numériques. Pour la RF, nous avons utilisé une bande de 20 GHz allant de 20 GHz à 40 GHz étudiée lors de travaux préliminaires réalisés dans le cadre du projet WiNoCoD [Hamieh *et al.*, 2014]. Cette bande est divisée en autant de sous-bandes que nécessaire pour créer un canal de données par nœud du NoC RF et un canal de service partagé par tous. Chaque canal est composé de 30 sous-bandes et chaque nœud RF a besoin de 2 sous-bandes dans le canal de service. Ainsi, un NoC RF de 16 nœuds a besoin de 16×30 sous-bandes pour les données et 16×2 sous-bandes pour le canal de service, soit 512 sous-bandes. Une bande de 20 GHz permet donc de créer 512 sous-bandes de 40 MHz. Si c'est la largeur de la bande utilisée par le NoC RF qui conditionne le débit de celui-ci, c'est le nombre de sous-bandes qui conditionne sa latence. Par exemple, dans la configuration ci-dessus, la durée d'un cycle RF est de 25 ns, soit l'équivalent de 25 cycles pour des composants numériques cadencés à 1 GHz.

Dans la suite de cette section, nous avons cherché à évaluer pour quelle configuration il devient plus favorable d'utiliser le NoC RF que le réseau filaire constitué par la grille. Pour ce faire, nous avons fait varier le nombre de tuiles composant le CMP et le nombre de nœuds composant le NoC RF. Nous avons ensuite évalué les caractéristiques de l'architecture WiNoCoD suivant deux approches.

Dans la première, nous avons fait varier le nombre de sous-bandes créées à partir de la bande initiale de 20 GHz en fonction du nombre de nœuds RF du réseau. Cette configuration permet d'attribuer un canal par nœud RF dans le schéma d'allocation statique. Ainsi plus le NoC RF comporte de nœuds, plus il y a de sous-bandes et plus la durée du cycle RF augmente.

Dans le deuxième cas, nous avons fixé le nombre de canaux créés à 16 et uniquement fait varier le nombre de nœuds RF du réseau. Dans cette approche, chaque nœud RF peut disposer de plusieurs canaux en configuration statique si on diminue le nombre de nœuds. Si au contraire, le nombre de nœuds RF dépasse le nombre de canaux, les nœuds ne pourront pas émettre à tous les cycles, interdisant par là même l'allocation statique des ressources de communication.

L'évaluation a été faite en étudiant deux schémas de communication : Une communication point à point, où une tuile choisie au hasard communique avec une autre tuile elle aussi choisie au hasard. Une communication de type broadcast, où une tuile choisie au hasard envoie un même message à toutes les autres tuiles du CMP.

6.2.2 Résultats

6.2.2.1 Communication point à point

La latence l_{RG} d'un routeur de la grille est de 3 cycles, un pour la lecture de la donnée en entrée, un pour le choix du port de sortie et un pour l'écriture de la donnée sur la sortie choisie. L'équation 6.1 donne D_{CMP}, le nombre moyen de routeurs traversés pour relier deux tuiles quelconques, routeur de la tuile d'origine compris, dans le cas d'un CMP formé par une grille de $N \times N$ tuiles. On peut donc obtenir L_{grille} la latence moyenne d'une grille grâce à l'équation 6.2, avec l_{RG}, la latence d'un routeur de la grille.

$$D_{CMP} = \frac{2}{3}N + 1 \tag{6.1}$$

$$L_{grille} = D_{CMP} \times l_{RG} \tag{6.2}$$

La latence l_{RF} d'un NoC RF de 16 nœuds est de 25 ns. En supposant une fréquence de fonctionnement de 1 GHz pour la partie numérique, on peut donc émettre un symbole tous les 25 cycles. Il faut ajouter à la latence du NoC RF le nombre moyen de cycles nécessaires pour que la donnée aille de la tuile source au routeur RF en émission et du routeur RF à la tuile destination en réception. Pour un CMP contenant des grappes de $M \times M$ tuiles, la distance moyenne D_{grappe} entre une tuile et le routeur RF est donnée par l'équation 6.3 si M est pair et par l'équation 6.4 si M est impair. Soit l_{RG}, la latence d'un routeur de la grille, et l_{RF}, la latence du NoC RF. L'équation 6.5 donne L_{NoC_RF} la latence moyenne du NoC RF. Nous pouvons alors comparer le délai moyen de réception d'un *FLow control unIT* (FLIT) avec le NoC RF à celui d'un FLIT avec une grille reliant toutes les tuiles du CMP entre elles.

$$D_{grappe} = \frac{M}{2} \tag{6.3}$$

$$D_{grappe} = \frac{M^2 + 2M - 1}{2M} \tag{6.4}$$

$$L_{NoC\_RF} = l_{RG} \times D_{grappe} + l_{RF} \tag{6.5}$$

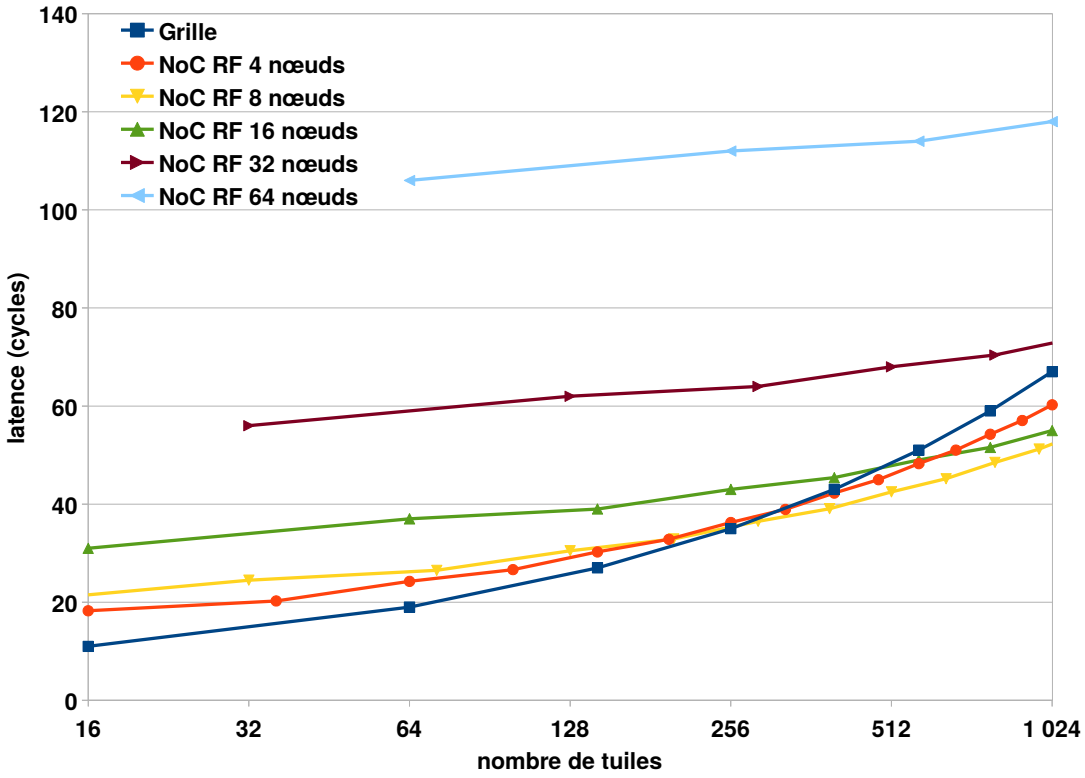

Figure 6.1 – Latence moyenne du transfert point à point d'un FLIT sur une grille et sur un NoC RF ayant autant de canaux que de nœuds

Nombre de sous-bandes proportionnel au nombre de nœuds RF La Figure 6.1 montre la latence moyenne des communications point à point en fonction du nombre de tuiles composant le CMP, pour un réseau filaire de type grille et pour plusieurs configurations du NoC RF. Nous avons ici découpé la bande totale de manière à disposer d'un canal par nœud RF.

On constate ainsi que c'est la taille du CMP qui fixe le nombre de nœuds du NoC RF. Le NoC RF doit être composé de 4 nœuds pour un CMP d'au plus 128 tuiles. Il doit posséder 8 nœuds pour un CMP de 128 à 1 024 tuiles. Alors que le besoin de disposer d'un NoC RF de 16 nœuds n'arrive que pour des CMP de 1 024 tuiles et plus. Ainsi pour cette taille de CMP, l'utilisation d'un NoC RF de 16 nœuds à la place d'une grille permet une diminution de la latence moyenne des communications point à point de 18 %. Finalement pour la configuration retenue ici, une bande de 20 GHz divisée en autant de canaux que le NoC RF contient de nœuds, il ne semble jamais intéressant d'utiliser un NoC RF de 32 ou 64 nœuds.

Les résultats précédents ne s'intéressent qu'aux différentes configurations possibles de NoC RF. Si on ajoute à cela les performances du réseau filaire, on constate qu'il ne devient intéressant d'utiliser le NoC RF que pour des CMP d'au moins 256 tuiles.

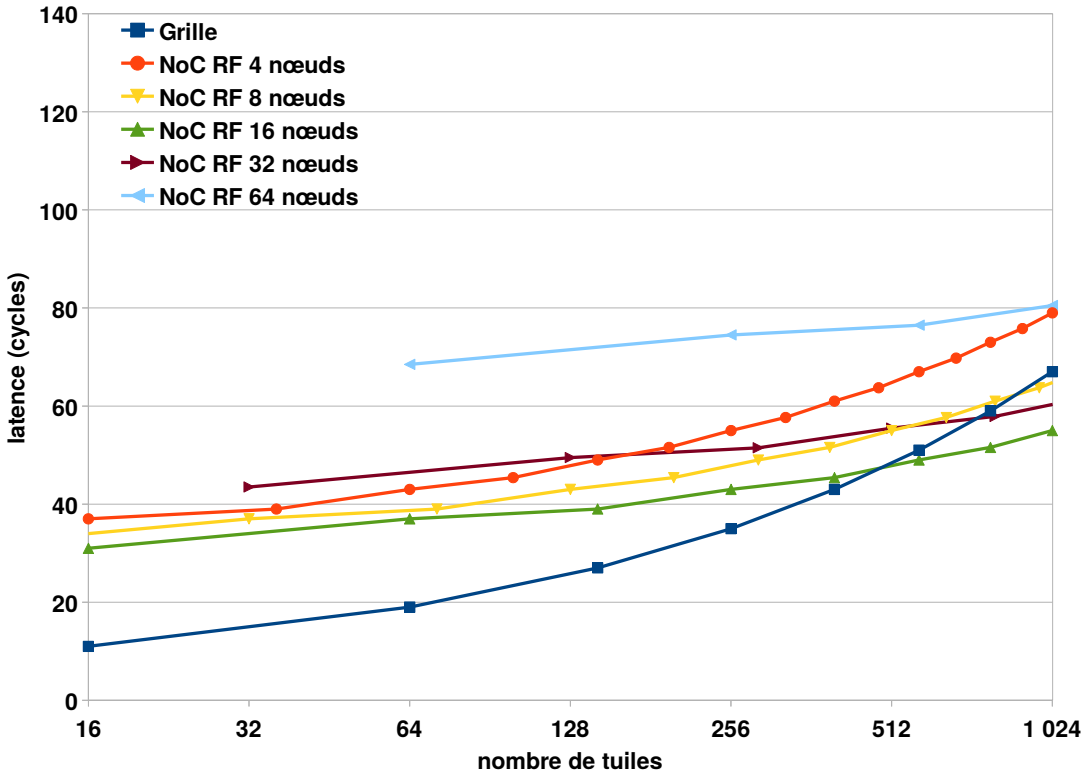

Figure 6.2 – Latence moyenne du transfert point à point d'un FLIT sur une grille et sur un NoC RF ayant un nombre fixe de 16 canaux de communication

Nombre de sous-bandes fixe La Figure 6.2 montre la latence moyenne des communications point à point en fonction du nombre de tuiles composant le CMP pour un réseau filaire de type grille et plusieurs configurations du NoC RF. Nous avons ici découpé la bande totale de manière à créer 16 canaux de communication, et ce quelque soit le nombre de nœuds constituant le NoC RF.

Si on diminue le nombre de nœuds RF et qu'il y a moins de nœuds que de canaux, la latence moyenne augmente. Cette augmentation est due au fait que la diminution du nombre de nœuds RF équivaut à une augmentation de la taille des grappes et donc à une augmentation de la distance moyenne entre une tuile et le nœuds RF dont elle dépend. La portion de la latence due au NoC RF est fixe alors que celle due au réseau filaire augmente. Si on augmente le nombre de nœuds RF et qu'il y a plus de nœuds que de canaux, la latence augmente. Cette augmentation est due au fait qu'en augmentant le nombre de nœuds RF, tous les nœuds ne peuvent pas émettre à chaque cycle RF. On constate donc que pour un nombre de canaux donné, la configuration optimale est celle où il y a autant de nœuds RF que de canaux.

6.2.2.2 Broadcast

Pour effectuer un broadcast dans une grille, les messages doivent traverser au maximum $2N - 1$ routeurs, avec N la largeur de la grille. Pour effectuer un broadcast dans l'architecture WiNoCoD, il faut traverser le NoC RF et au maximum $2M$ routeurs, avec M la largeur d'une grappe. On a donc une latence qui varie selon la taille du CMP pour la grille alors qu'elle varie selon la taille des grappes pour le NoC RF.

De plus, tout message circulant sur le NoC RF est intrinsèquement un broadcast. Cette propriété s'explique par l'allocation des canaux aux émetteurs et non aux récepteurs. Chaque récepteur reçoit la totalité des messages circulant sur le NoC RF et décide ensuite de ne garder que ceux qui lui sont destinés. En plus d'avoir une plus faible latence, le NoC RF permet donc de faire circuler plusieurs broadcasts en même temps puisque chaque message circulant sur le NoC RF peut être un broadcast.

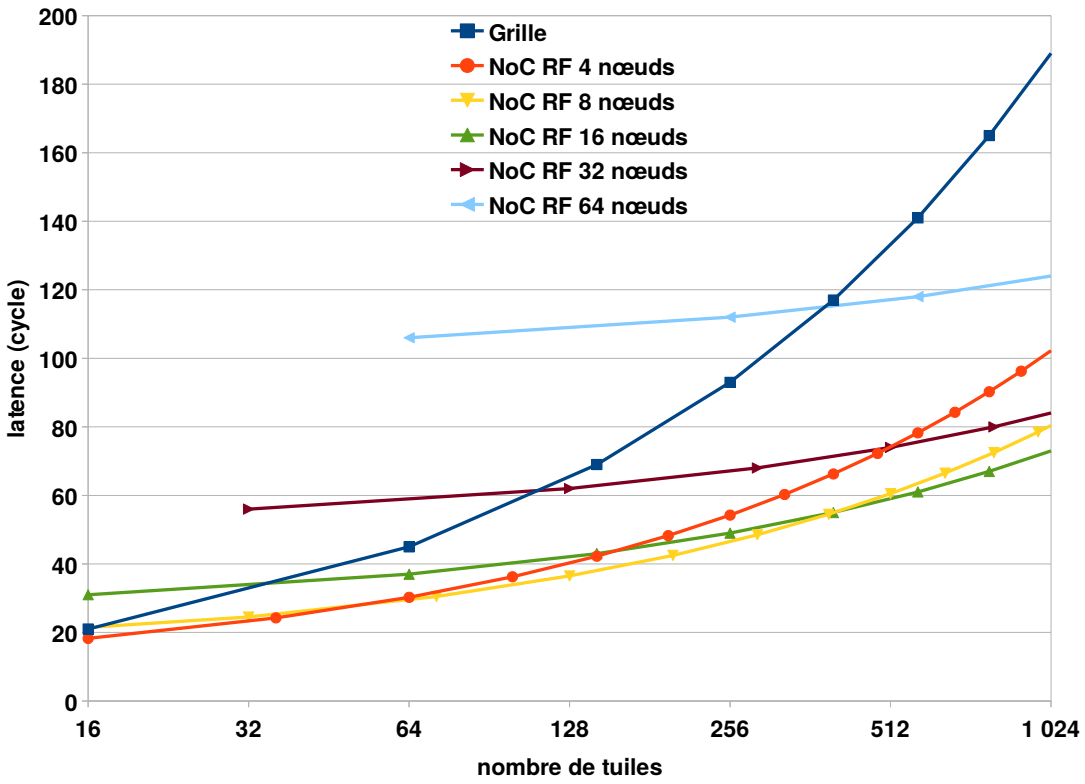

Figure 6.3 – Latence maximale du broadcast d'un FLIT sur une grille et sur un NoC RF ayant autant de canaux que de nœuds

Nombre de sous-bandes proportionnel au nombre de nœuds RF La Figure 6.3 montre la latence maximale d'un broadcast en fonction du nombre de tuiles composant le CMP pour un réseau filaire de type grille et plusieurs configurations du NoC RF. Nous avons ici découpé la bande totale de manière à disposer d'un canal par nœud RF.

Les courbes montrent que les NoC RF à 4 et 8 nœuds sont plus performants que le réseau filaire, et ce, quelque soit la taille du CMP. Alors que pour configurations du NoC RF ayant 16, 32 et 64 nœuds, le CMP doit être composé respectivement d'au moins 64, 128 et 512 tuiles.

Ces courbes permettent aussi d'évaluer les performances des configurations du NoC RF les unes par rapport aux autres. Le NoC RF à 4 nœuds est la meilleure configuration pour les CMP de 16 à 64 tuiles. Le NoC RF à 8 nœuds est la meilleure configuration pour des CMP de plus de 64 tuiles et jusqu'à 512. Le NoC RF à 16 nœuds est la meilleure configuration pour les CMP de plus de 512 tuiles. Finalement, les NoC RF à 32 et 64 nœuds ne sont jamais intéressants pour les tailles de CMP présentées sur la Figure 6.3.

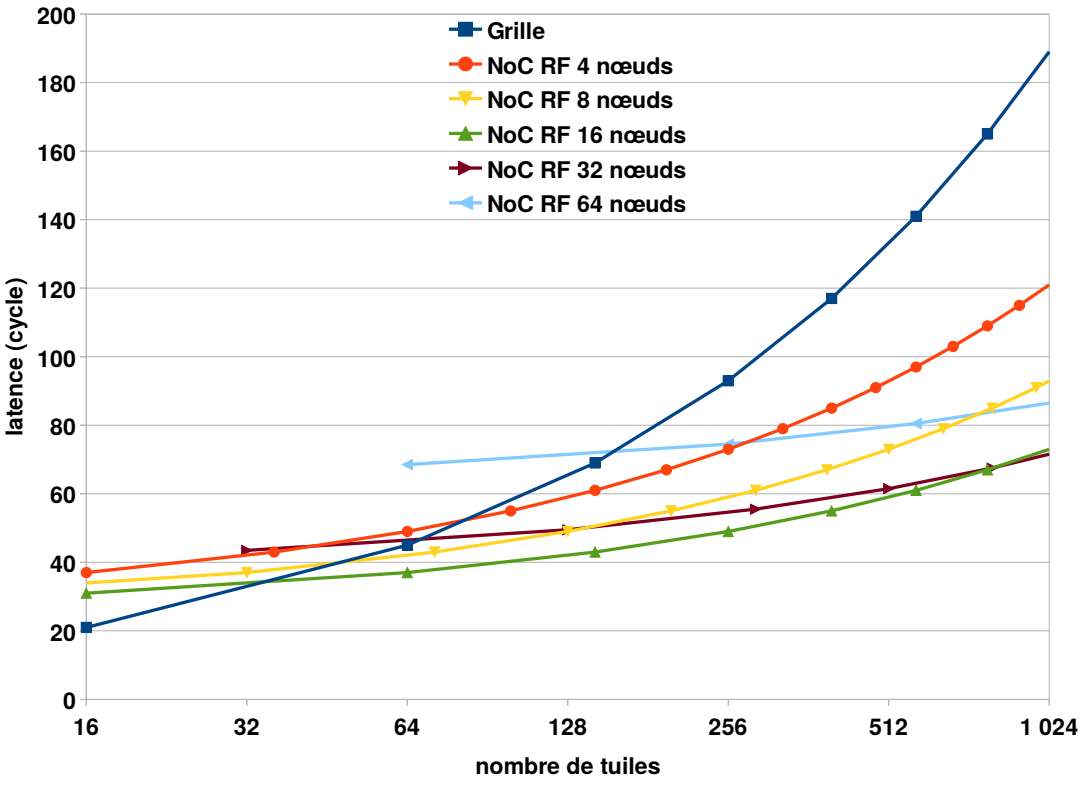

Figure 6.4 – Latence maximale du broadcast d'un FLIT sur une grille et sur un NoC RF à un nombre fixe de 16 canaux de communication

Nombre de sous-bandes fixe La Figure 6.4 montre la latence moyenne des communications point à point en fonction du nombre de tuiles composant le CMP pour un réseau filaire de type grille et plusieurs configurations du NoC RF. Nous avons ici découpé la bande totale de manière à créer 16 canaux de communication, et ce quelque soit le nombre de nœuds constituant le NoC RF.

Ces courbes montrent que la meilleure configuration pour le NoC RF est celle à 16 nœuds. Dans cette configuration, il y a autant de nœuds RF que de canaux. De plus,

Nombre de tuiles	16	32	64	128	256	512	1 024
Broadcast	=	✓	✓	✓	✓	✓	✓
Nombre de nœuds RF	4	4	8	8	8	16	16
Point à point	✗	✗	✗	✗	=	✓	✓
Nombre de nœuds RF	4	4	4	4	8	8	8

Tableau 6.1 – Configurations optimales en fonction du nombre de tuiles du CMP

elle offre de meilleures performances que le réseau filaire pour les CMP de 64 tuiles et plus. Les NoC RF à 4 et 8 nœuds sont toujours moins performants que le NoC RF à 16 nœuds quelque soit la taille du CMP. Le NoC RF à 32 nœuds égale et même dépasse les performances du NoC RF à 16 nœuds pour des CMP de 1 024 tuiles et plus. Finalement, le NoC RF à 64 nœuds dépassera les performances des autres configurations pour des nombres de tuiles supérieurs à ceux retenus dans le cadre de cette étude.

Ces résultats sont influencés par deux facteurs. D'une part, la latence du réseau RF étant fixe, plus la taille du CMP augmente, plus la part de la latence due au réseau filaire est importante. Cette latence du réseau filaire dépend de la taille des grappes qui dépend elle même de la taille du CMP et du nombre de nœuds RF. L'augmentation du nombre de nœuds RF permet donc de limiter l'impact de l'augmentation de la taille du CMP sur la latence. D'autre part, pour les configurations du NoC RF comprenant plus de nœuds que de canaux, un nœud RF n'a pas en permanence un canal qui lui est alloué et doit donc parfois attendre un cycle RF pour pouvoir émettre. C'est donc en fonction de la taille du CMP que l'un ou l'autre des facteurs devient prédominant.

6.2.3 Analyse

Dans cette section nous avons évalué les performances intrinsèques du NoC sans tenir compte de ses capacités de reconfiguration dynamique. Nous avons analysé la latence des communications en fonction de la taille du CMP et du nombre de nœuds composant le NoC RF. Les résultats nous montrent que le nombre de nœuds du NoC RF peut être fixé de manière optimale en fonction de la taille du CMP et de la latence du NoC RF. Cette latence dépend directement de la technologie CMOS utilisée puisque, pour un nombre de sous-bandes donné, c'est la largeur de bande qui fixe la durée du cycle RF. La technologie CMOS utilisé et la largeur de bande ont aussi une forte influence sur la consommation et la surface. Ces caractéristiques physiques seront évaluées dans la section suivante.

Pour les communications point à point, la configuration optimale est celle où la bande est divisée de manière à créer autant de canaux de communication qu'il y

a de nœuds dans le NoC RF. Il en est de même pour les communications de type broadcast jusqu'à 1 024 tuiles. La configuration avec deux fois plus de nœuds RF que de canaux de communication atteint les performances de celle avec un canal par nœud RF pour les CMP de plus de 1 024 tuiles.

Cette section nous a donc permis d'établir la liste des meilleures configurations du NoC RF en fonction de la taille du CMP. De plus, nous avons pu établir pour quelle taille de CMP le NoC RF devenait plus performant que le réseau filaire selon deux schémas de communications. Le Tableau 6.1 résume ces résultats. Ainsi, pour les communications point à point, le NoC RF surpasse les réseaux filaires de type grille 2D pour les CMP de 256 tuiles et plus . Pour les communications de type broadcast, le NoC RF offre de meilleures performances quelque soit la taille du CMP.

6.3 Évaluation technologique

Dans cette section, nous présentons l'évaluation des caractéristiques physiques de l'architecture WiNoCoD. Ces résultats se basent sur l'état de l'art des différents composants utilisés dans le NoC RF à travers une étude bibliographique. Cette section permet de déterminer quelles configurations sont réalisables physiquement parmi celles identifiées dans la section précédente en fonction de leur surface et de leur consommation.

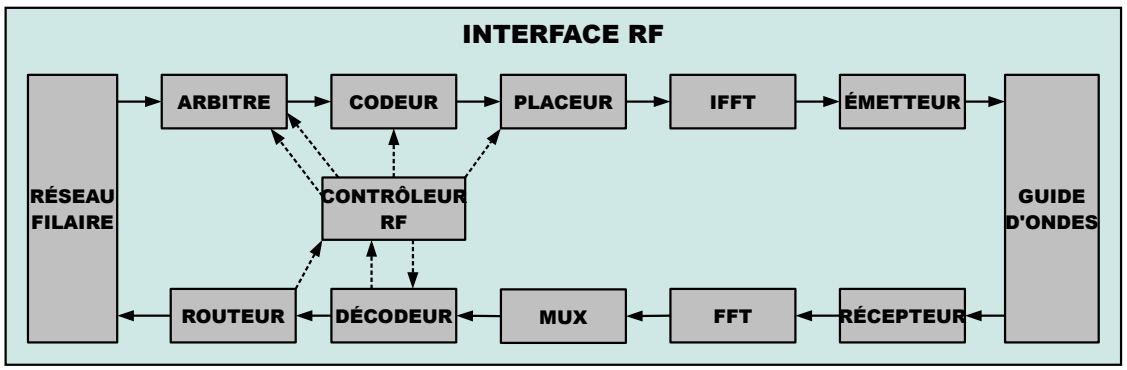

Figure 6.5 – Schéma bloc de l'Interface RF

6.3.1 Protocole expérimental

Afin d'évaluer la faisabilité de l'architecture WiNoCoD, nous avons procédé à l'estimation de la surface et de la consommation de la réalisation physique d'une telle puce. Pour évaluer la part des différents composants dans la surface et dans la consommation de la manière la plus fiable possible, nous avons effectué une mise à l'échelle de ces différents composants. Nous avons retenu une finesse de gravure de 22 nm comme technologie cible. Cette technologie est actuellement utilisée par

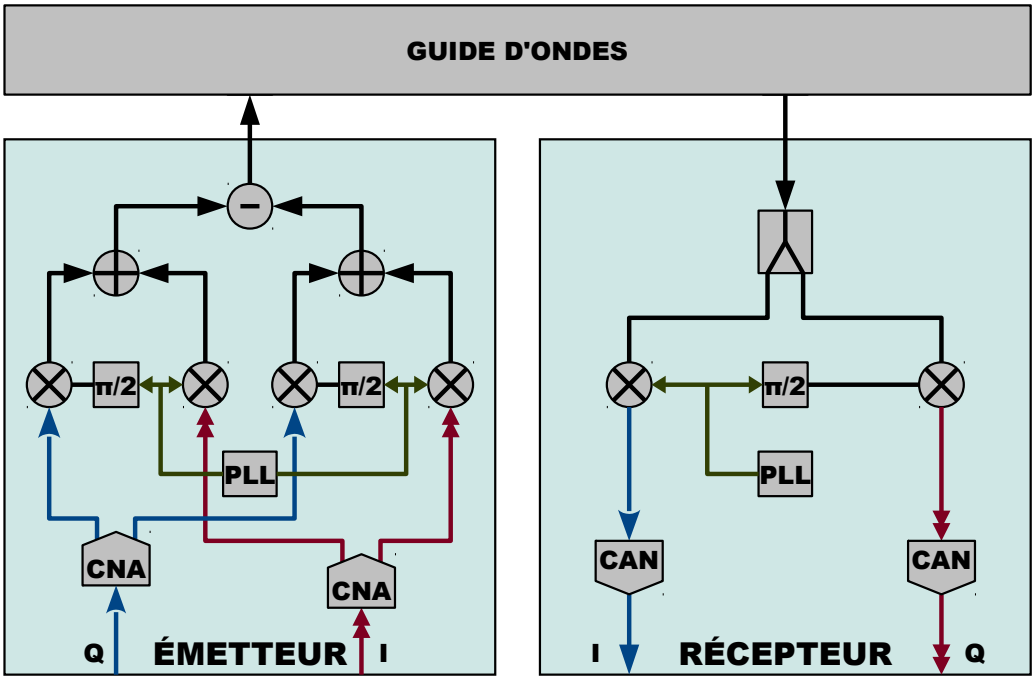

Figure 6.6 – Architecture des émetteurs et récepteurs de l'interface RF

des processeurs comme ceux de la gamme Haswell d'Intel [Kurd *et al.*, 2014] ou le POWER8 d'IBM [Fluhr *et al.*, 2015]

Nous avons évalué plusieurs versions de notre architecture en faisant varier le nombre de tuiles par grappe. Le nombre de cœurs par tuile est fixé à 4. Le nombre de grappes est choisi en fonction de la taille totale du CMP en accord avec les résultats de la section 6.2.

La brique de base permettant de construire le NoC RF WiNoCoD est son interface RF dont nous rappelons l'architecture dans la Figure 6.5. Nous sommes partis de l'état de l'art de ses différents composants en choisissant ceux se rapprochant au mieux des caractéristiques de notre NoC RF. Soit une bande de 20 GHz allant de 20 GHz à 40 GHz étudiée lors de travaux préliminaires réalisés par des partenaires du projet [Hamieh *et al.*, 2014]. Les composants occupant le plus de surface ou ayant la plus grosse consommation sont les FFT/IFFT [Milder *et al.*, 2012], les émetteurs/-récepteurs [Jongsun *et al.*, 2011] et la ligne de transmission [Hu *et al.*, 2013].

La Figure 6.6 rappelle l'architecture des émetteurs et récepteurs utilisés dans WiNoCoD [Drillet *et al.*, 2014]. L'émetteur est composé de deux Convertisseurs Numériques Analogiques (CNA), deux filtres passe-bas, une PLL, quatre mélangeurs, deux décaleurs de phase, deux additionneurs et un amplificateur différentiel. Le récepteur est quant à lui composé de deux Convertisseurs Analogiques Numériques (CAN), une PLL, deux mélangeurs, un décaleur de phase et un diviseur de fréquence. Pour le reste du CMP, nous nous sommes basés sur une fréquence de fonctionnement de 1 GHz et avons pris en compte les RAM [Gong *et al.*, 2008], les routeurs du réseau filaire [Miro-Panades *et al.*, 2008] et les cœurs, des Cortex-A5 [Lee

	Intel Xeon 7150	SUN SPARC T4	Intel Xeon E5	IBM POWER8	NVIDIA P100
Technologie CMOS	65 nm	40 nm	22 nm	22 nm	16 nm
Fréquence	3,5 GHz	3 GHz	3 GHz	3 GHz	1,5 GHz
Nb Cœurs	2	8	18	12	3 584
Nb Threads	4	64	36	96	2 048
Surface	435 mm^2	403 mm^2	662 mm^2	649 mm^2	610 mm^2
Consommation	150 W	240 W	160 W	190 W	300 W
Surface Normalisée	50 mm^2	122 mm^2	662 mm^2	649 mm^2	1 153 mm^2
Consommation Normalisée	41 W	117 W	160 W	190 W	441 W

Tableau 6.2 – Surface (mm^2) et consommation (W) de puces existantes

et al., 2014]. Ce cœur a été choisi car c'est une architecture assez répandue qui présente l'avantage d'être basse consommation et d'occuper une faible surface. Ces deux derniers critères étant importants dans l'optique d'une architecture many-cœurs à plusieurs centaines, voire milliers de cœurs.

Nous avons ensuite comparé ces données avec celles de puces existantes présentées dans le Tableau 6.2. Comme nous cherchons à fixer les limites de l'acceptable en termes de surface et de consommation, toutes les puces retenues correspondent au haut de gamme au moment de leur sortie. Nous avons retenu les processeurs Xeon 7150N [Rusu *et al.*, 2007] et E5-2699 v3 [Bowhill *et al.*, 2016] d'Intel, POWER8 [Fluhr *et al.*, 2015] d'IBM et SPARC T4 [Shah *et al.*, 2012] de SUN. Ces quatre processeurs sont destinés aux stations de travail et aux serveurs. De plus nous avons retenu un GPU NVIDIA, le Tesla P100 [NVIDIA, 2016]. Le Xeon 7150N possède 2 cœurs et utilise un bus pour accéder au cache L3 partagé et aux communications externes. Le Xeon E5 possède 18 cœurs et utilise un réseau de type anneau. Le POWER8 possède 12 cœurs et utilise un réseau composé de 8 bus fonctionnant en parallèles [Starke *et al.*, 2015]. Le SPARC T4 possède 8 cœurs et utilise un crossbar. Finalement, le Tesla P100 est un GPU conçu pour être utilisé comme accélérateur matériel dans le cadre du calcul flottant.

Le Tableau 6.2 nous montre que la surface et la consommation réelle des puces reste toujours dans le même ordre de grandeur malgré l'amélioration de la finesse de gravure. Ainsi les surfaces et consommations normalisées dans la technologie cible de 22 nm ne sont données qu'à titre indicatif, mais c'est bien la consommation et la surface réelle de ces puces qui nous intéresse ici. La surface varie entre 403 et 662 mm^2

alors que la consommation varie entre 150 et 300 W. Bien que l'amélioration des technologies aurait pu être utilisée pour faire baisser la surface et la consommation, l'analyse des surfaces et des consommations montre qu'un autre choix a été fait. Même si on exclut le Tesla P100, tous les constructeurs ont fait le choix de produire des puces de plus en plus parallèles. Le Xeon d'Intel est ainsi passé de 2 cœurs en 2007 à 18 cœurs en 2016.

6.3.2 Résultats

6.3.2.1 Surface

Dans ce qui suit, nous avons utilisé l'équation 6.6 pour obtenir S_N la surface normalisée dans une technologie N, ici 22 nm, en fonction de S_Q la surface dans la technologie d'origine Q.

$$S_N = S_Q \times (N/Q)^2 \tag{6.6}$$

Pour les composants analogiques la mise à l'échelle n'est pas aussi simple, leur surface ne dépend pas uniquement de la finesse de gravure utilisée. Nous avons donc utilisé ici les surfaces de l'état de l'art de ces composants comme borne supérieure.

La surface occupée par notre NoC RF est principalement due aux FFT/IFFT [Milder *et al.*, 2012], aux émetteurs/récepteurs [Drillet *et al.*, 2014] et à la ligne de transmission [Hu *et al.*, 2013]. Nous allons tout d'abord détailler l'estimation de la surface des FFT/IFFT, puis celle des émetteurs/récepteurs et finalement celle de la ligne de transmission. Nous donnerons ensuite l'estimation de la surface totale du CMP incluant notre NoC RF.

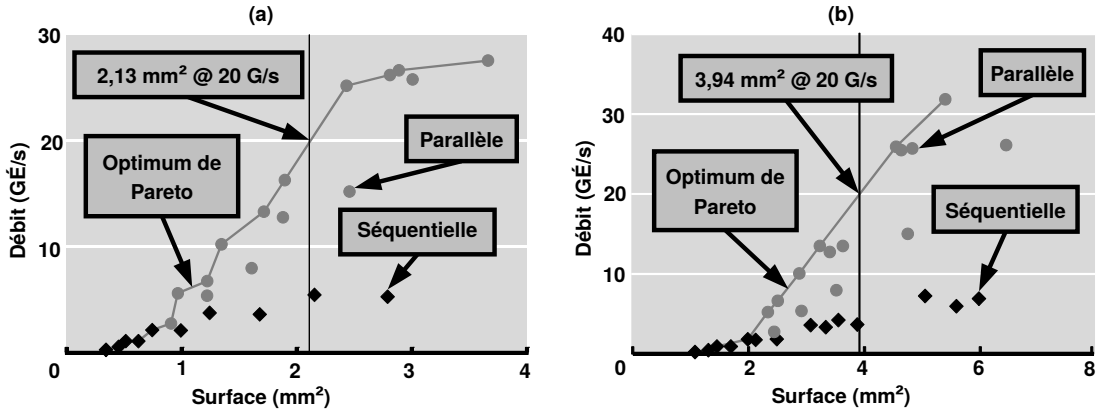

Figure 6.7 – Surface (mm^2) en fonction du taux d'échantillonnage (Giga Échantillons par seconde) de FFT 1024 (a) et 4096 (b) points pour une implémentation ASIC 65 nm ***(extrait de [Milder* et al.*, 2012])***

La Figure 6.7 est tirée des travaux de Milder *et al.* [2012]. Elle présente l'évolution de la surface de deux FFT de 1 024 et 4 096 points en fonction du taux d'échantillonnage recherché. Il est intéressant de constater que l'utilisation d'une architecture parallèle permet jusqu'à une amélioration du débit d'un facteur cinq par rapport à une architecture séquentielle à surface égale.

Nombre de points	128	256	512	1 024	4 096
Surface (65 nm)	0,85	1,15	1,57	2,13	3,97
Surface (22 nm)	0,10	0,13	0,18	0,24	0,45

Tableau 6.3 – Surface (mm^2) optimale pour différentes tailles de FFT en 22 et 65 nm

Le Tableau 6.3 présente les surfaces de différentes tailles de FFT à architecture parallèle dans deux technologies CMOS. La deuxième ligne présente les surfaces dans la technologie 65 nm utilisée dans les travaux de Milder *et al.* [2012]. Les valeurs pour les FFT de 1 024 et 4 096 sont directement extraites de ces travaux alors que les valeurs pour les FFT de 128, 256 et 512 points sont extrapolées. La troisième ligne présente les surfaces des différentes tailles de FFT mis à l'échelle en 22 nm en utilisant la formule 6.6.

Le Tableau 6.4 présente les résultats de l'estimation détaillée de la surface de l'émetteur décrit dans la Figure 6.6. On y constate que la part la plus importante de la surface d'un émetteur est due aux quatre mélangeurs.

Composant	Surface Individuelle	Nombre Élément	Surface Totale
Additionneur [Drillet *et al.*, 2014]	0,09	2	0,18
Amplificateur [Drillet *et al.*, 2014]	0,10	1	0,10
CNA [Nazemi *et al.*, 2015]	0,05	2	0,10
Décaleur [Biglarbegian *et al.*, 2009]	0,08	2	0,16
Mélangeur [Drillet *et al.*, 2014]	0,17	4	0,68
PLL [Ferriss *et al.*, 2014]	0,02	1	0,02
Surface totale émetteur	–	–	**1,24**

Tableau 6.4 – Estimation de la surface (mm^2) d'un émetteur

Le Tableau 6.5 présente les résultats de l'estimation détaillée de la surface du récepteur décrit dans la Figure 6.6. Tout comme pour l'émetteur, on y constate que la part la plus importante de la surface d'un récepteur est de nouveau due aux mélangeurs.

Pour ce qui est du reste de la puce, la majeure partie de la surface est occupée par les cœurs et leurs caches ainsi que les mémoires locales. Un Cortex-A5 d'ARM doté de deux caches de 16 Kio, occupe une surface de 0,53 mm^2 en utilisant une finesse

Composant	Surface Individuelle	Nombre Élément	Surface Totale
CAN [Kull *et al.*, 2014]	0,05	2	0,10
Décaleur [Biglarbegian *et al.*, 2009]	0,08	1	0,08
Diviseur [Ercoli *et al.*, 2012]	0,01	1	0,01
Filtre passe-bas [Drillet *et al.*, 2014]	0,04	2	0,08
Mélangeur [Drillet *et al.*, 2014]	0,17	2	0,34
PLL [Ferriss *et al.*, 2014]	0,02	1	0,02
Surface totale récepteur	–	–	**0,63**

Tableau 6.5 – Estimation de la surface (mm^2) d'un récepteur

de gravure de 40 nm [Lee *et al.*, 2014]. Une mémoire telle que la RAM utilisée dans nos tuiles occupe une surface de 1,25 mm^2 [Gong *et al.*, 2008]. Nous avons ainsi pu établir les résultats présentés dans le tableau 6.6.

Configuration Architecture							
Nombre de cœurs	64	128	256	512	1 024	2 048	4 096
Nombre de tuiles	16	32	64	128	256	512	1 024
Nombre de grappes	4	4	4	8	8	8	16
Surface							
IFFT/FFT	0,78	0,78	0,78	2,11	2,11	2,11	5,76
Émetteur	4,96	4,96	4,96	9,92	9,92	9,92	19,84
Récepteur	2,52	2,52	2,52	5,04	5,04	5,04	10,08
Guide d'onde	0,48	0,68	0,96	1,92	2,72	3,84	7,69
Total NoC RF	**8,74**	**8,94**	**9,22**	**18,99**	**19,79**	**20,91**	**43,36**
Cœurs	10,26	20,52	41,04	82,09	164,17	328,35	656,69
Mémoires locales	0,56	1,12	2,25	4,50	8,99	17,99	35,98
Routeurs	0,09	0,17	0,34	0,69	1,37	2,74	5,48
CMP hors NoC RF	**10,91**	**21,82**	**43,63**	**87,27**	**174,54**	**349,08**	**698,15**
Total CMP	**19,65**	**30,76**	**52,85**	**106,26**	**194,32**	**369,99**	**741,52**
Part NoC RF	**44,5 %**	**29,1 %**	**17,4 %**	**17,9 %**	**10,2 %**	**5,7 %**	**5,8 %**

Tableau 6.6 – Estimation de la surface (mm^2) de la puce

6.3.2.2 Consommation

Les deux scénarios d'évolution des transistors les plus repris sont ceux d'Intel [Borkar et Chien, 2011] et de l'ITRS [ITRS, 2013]. Ainsi pour une diminution d'un facteur

0,7 de la taille des transistors, la diminution de la consommation est comprise entre 0,53 et 0,65 selon l'ITRS et entre 0,65 et 0,74 selon Intel. On peut donc retenir pour le passage d'une technologie à une autre un facteur de 0,7 pour la taille des transistors et de 0,65 pour la consommation. L'équation 6.7 donne W_N la consommation normalisée dans une technologie N en fonction de W_Q la consommation dans la technologie d'origine Q. Nous avons ainsi pu effectuer la mise à l'échelle des composants numériques de notre architecture. Tout comme pour la surface, nous avons utilisé ici la consommation de l'état de l'art des composants analogiques comme borne supérieure.

$$W_N = W_Q \times 0.65^{\log_{0.7}(N/Q)} \tag{6.7}$$

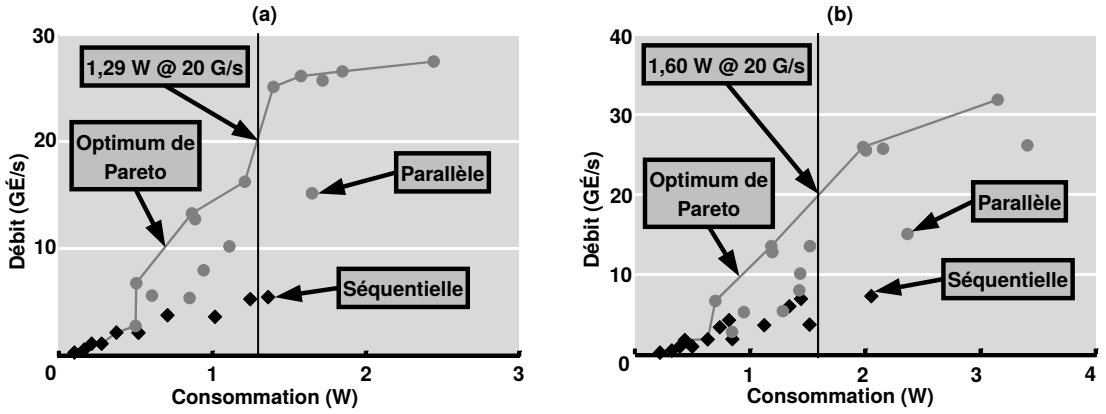

Figure 6.8 – Consommation (W) en fonction du taux d'échantillonnage (Giga Échantillons par seconde) de FFT 1024 (a) et 4096 (b) points pour une implémentation ASIC 65 nm ***(extrait de [Milder* et al.*, 2012])***

Tout comme pour la surface, la Figure 6.8 est tirée des travaux de Milder *et al.* [2012]. Elle présente l'évolution de la consommation de deux FFT de 1 024 et 4 096 points en fonction du taux d'échantillonnage recherché. Alors que l'utilisation d'une architecture parallèle permet d'augmenter le débit sans augmenter significativement la surface, il n'en est pas de même pour la consommation. L'architecture parallèle consomme deux fois plus que l'architecture séquentielle. Cette constatation est assez logique puisque le but de toute architecture pipelinée est d'augmenter le taux d'utilisation de ses différents composants.

Nombre de points	128	256	512	1 024	4 096
Consommation (65 nm)	0,94	1,05	1,16	1,29	1,60
Consommation (22 nm)	0,25	0,28	0,31	0,35	0,43

Tableau 6.7 – Consommation (W) optimale pour différentes tailles de FFT en 22 et 65 nm

Le Tableau 6.7 présente la consommation de différentes tailles de FFT à architecture parallèle dans deux technologies CMOS. La deuxième ligne présente les valeurs pour la technologie 65 nm utilisée dans les travaux de Milder *et al.* [2012]. Les valeurs pour les FFT de 1 024 et 4 096 sont directement extraites de ces travaux alors que les valeurs pour les FFT de 128, 256 et 512 points sont extrapolées. La troisième ligne présente la consommation des différentes tailles de FFT mises à l'échelle en 22 nm selon la formule 6.7.

Composant	Consommation Individuelle	Nombre Élément	Consommation Totale
Additionneur [Drillet *et al.*, 2014]	3,2	2	6,4
Amplificateur [Drillet *et al.*, 2014]	6,4	1	6,4
CNA [Nazemi *et al.*, 2015]	84	2	168
Décaleur [Biglarbegian *et al.*, 2009]	passif	2	passif
Mélangeur [Drillet *et al.*, 2014]	passif	4	passif
PLL [Ferriss *et al.*, 2014]	31	1	31
Consommation totale émetteur	–	–	**212**

Tableau 6.8 – Estimation de la consommation (mW) d'un émetteur

Le Tableau 6.8 présente les résultats de l'estimation détaillée de la consommation de l'émetteur. On y constate que la part la plus importante de la consommation d'un émetteur est due aux deux CNA. Le Tableau 6.9 présente les résultats de l'estimation détaillée de la consommation du récepteur. On y constate que la part la plus importante de la consommation d'un récepteur est due au CAN.

Composant	Consommation Individuelle	Nombre Élément	Consommation Totale
CAN [Kull *et al.*, 2014]	86	2	172
Décaleur [Biglarbegian *et al.*, 2009]	passif	1	passif
Diviseur [Ercoli *et al.*, 2012]	passif	1	passif
Filtre passe-bas [Drillet *et al.*, 2014]	passif	2	passif
Mélangeur [Drillet *et al.*, 2014]	passif	2	passif
PLL [Ferriss *et al.*, 2014]	31	1	31
Consommation totale récepteur	–	–	**203**

Tableau 6.9 – Estimation de la consommation (mW) d'un récepteur

En ce qui concerne le reste de la puce, la majeure partie de la consommation est due aux cœurs. Nous avons ainsi pu établir les résultats présentés dans le tableau 6.10. Ce tableau présente l'estimation de la consommation pour les configurations où

l'utilisation de WiNoCoD pourrait améliorer les performances aux vue des résultats de la section 6.2.

Configuration Architecture							
Nombre de cœurs	64	128	256	512	1 024	2 048	4 096
Nombre de tuiles	16	32	64	128	256	512	1 024
Nombre de grappes	4	4	4	8	8	8	16
Consommation							
IFFT/FFT	2,03	2,03	2,03	4,54	4,54	4,54	10,03
Émetteur	0,85	0,85	0,85	1,70	1,70	1,70	3,39
Récepteur	0,81	0,81	0,81	1,62	1,62	1,62	3,25
Total NoC RF	**3,69**	**3,69**	**3,69**	**7,86**	**7,86**	**7,86**	**16,67**
Cœurs	2,49	4,97	9,95	19,90	39,79	79,59	159,17
Mémoires locales	0,18	0,36	0,72	1,45	2,89	5,78	11,56
Routeurs	0,09	0,19	0,37	0,75	1,50	3,00	5,99
CMP hors NoC RF	**2,76**	**5,52**	**11,05**	**22,09**	**44,18**	**88,36**	**176,73**
Total CMP	**6,45**	**9,21**	**14,74**	**29,95**	**52,04**	**96,22**	**193,40**
Part NoC RF	**57,2 %**	**40,1 %**	**25,1 %**	**26,2 %**	**15,1 %**	**8,2 %**	**8,6 %**

Tableau 6.10 – Estimation de la consommation de la puce (W)

6.3.3 Analyse

6.3.3.1 Surface

Nous avons analysé l'estimation de la surface suivant deux approches. Dans un premier temps, nous avons comparé la surface totale de notre CMP à celle de CMP existants. Les plus grosses puces actuelles ont une surface de l'ordre de 600 mm^2. Or toutes les configurations présentées, exceptée celle à 4 096 cœurs, ont une surface inférieure à cette valeur. Il en ressort donc que du point de vue de la taille totale de la puce, toutes les configurations exceptée la dernière sont réalisables avec les technologies actuelles. Le CMP à 4 096 cœurs n'est pas raisonnablement réalisable en 22 nm mais pourra l'être avec les finesses de gravure suivantes. Dans un second temps, nous avons comparé le ratio entre la surface du NoC et la surface totale de la puce de notre CMP à celui de CMP existants. Ce ratio est par exemple de 5 % pour le NoC proposé par Salihundam *et al.* [2011] et de 10 % pour celui proposé par Daya *et al.* [2014]. Les configurations à 1 024, 2 048 et 4 096 cœurs présentent un ratio de cet ordre de grandeur. Il ressort de la mise en parallèle de ces deux approches que les configurations à 1 024 et 2 048 cœurs ont des caractéristiques réalistes en terme de surface avec une finesse de gravure de 22 nm.

6.3.3.2 Consommation

L'analyse du Tableau 6.10 met en avant que les coûts en termes de consommation ne varient pas de la même façon que les coûts en terme de surface. Cela s'explique du fait de la part croissante du guide d'onde dans la surface du NoC RF alors que la variation de sa longueur n'est pas prise en compte pour le calcul de la consommation. Les composants des émetteurs et des récepteurs ayant été dimensionnés pour la longueur maximale du guide d'onde. Ainsi, pour un nombre de nœuds RF donné, la consommation estimée du NoC RF sera la même quelque, et ce, soit le nombre de cœurs du CMP.

Nous avons analysé les résultats de l'estimation de la consommation suivant deux approches. Dans un premier temps, nous avons comparé la consommation totale de notre CMP à celle de CMP existants. Les plus grosses puces actuelles ont une consommation de l'ordre de 200 W. Or toutes les configurations de CMP utilisant WiNoCoD présentées ont une consommation inférieure à cette valeur. Il en ressort donc que du point de vue de la consommation totale de la puce, toutes les configurations sont réalisables. Dans un second temps, nous avons comparé le ratio entre la consommation du NoC et la consommation totale de la puce de notre CMP à celui de CMP existants. Ce ratio est par exemple de 10 % pour le NoC proposé par Salihundam *et al.* [2011] et de 20 % pour celui proposé par Daya *et al.* [2014]. Les configurations à 1 024, 2 048 et 4 096 cœurs présentent un ratio de cet ordre de grandeur. Il ressort de la mise en parallèle de ces deux approches que les configurations à 1 024, 2 048 et 4 096 cœurs ont des caractéristiques réalistes en terme de consommation avec une finesse de gravure de 22 nm.

6.3.3.3 Synthèse

Dans cette section nous avons évalué la surface et la consommation de l'implémentation physique de notre architecture. On peut espérer une diminution de la consommation avec la baisse de la tension d'alimentation permise par l'amélioration des technologies. D'autre part on peut espérer que l'augmentation de la fréquence utilisée pour la bande RF permise par les nouvelles technologies permettra aussi une diminution de la surface des composants analogiques [Sansen, 2015].

6.4 Évaluation via un trafic réel

Pour évaluer notre architecture, nous avons développé le modèle SystemC CABA présenté dans le Chapitre 5. Dans la section 6.2 nous avons réalisé l'évaluation théorique des apports de WiNoCoD. Dans cette section, nous cherchons à confirmer les

résultats établis dans la section 6.2. Pour ce faire, nous allons évaluer le comportement de WiNoCoD, y compris l'allocation dynamique des canaux de communication, quand l'architecture est soumise à un trafic réel.

6.4.1 Protocole expérimental

Le modèle utilisé comprend l'intégralité de l'architecture, y compris les composants internes des tuiles. Nous disposons ainsi d'un modèle comprenant des processeurs, des mémoires et différents périphériques capable d'exécuter des applications. Cette section se base donc sur l'exécution d'applications directement sur le modèle de notre architecture.

L'architecture utilisée repose sur une mémoire partagée distribuée à cohérence matérielle développée dans le cadre du projet ANR *Tera-Scale ARchitecture* (TSAR) [Greiner, 2009]. Pour ce faire, elle utilise cinq réseaux différents :

— Commandes
— Réponses
— Commandes de cohérence
— Réponses de cohérence
— Acquittement de cohérence

Ces réseaux peuvent être des réseaux physiques différents ou des réseaux virtuels multiplexés sur un unique réseau physique. Dans le cadre de notre étude, nous utilisons cinq réseaux physiques DSPIN et un unique NoC RF sur lequel sont multiplexés les cinq réseaux filaires.

Si cette méthode d'évaluation est la plus précise, c'est aussi la plus longue à mettre en place et à expérimenter. Dans le cadre de cette étude, nous n'avons ainsi pu tester que deux applications et nous n'avons pu évaluer que des CMP composés d'au plus 8×8 tuiles. Cette limitations s'expliquent pour plusieurs raisons. Tout d'abord, plus la taille de la plateforme augmente, plus le temps de simulation augmente. Ensuite, le placement des applications et la gestion du micro système d'exploitation utilisé devient de plus en plus complexe. Pour finir, il s'est avéré que les composants matériels de SoClib n'étaient pas tous utilisables en l'état pour des plateformes de plus grande taille.

La configuration de la plateforme utilisée dans cette section est la suivante :

— modulation 16-QAM
— allocation dynamique des canaux de communication
— nombre de nœuds RF variable
— 64 tuiles de 4 processeurs chacune
— 8 canaux de 30 sous-bandes chacun pour les données
— 1 canal de service de 16 sous-bandes

6.4.1.1 Transpose

La première application utilisée réalise la transposition d'une image. C'est une application multi-thread simple fonctionnant en quatre étapes :

1. Chargement de l'image depuis le disque
2. Transposition de l'image
3. Affichage de l'image transposée
4. Enregistrement de l'image transposée dans le disque

Durant l'étape 1, chaque tuile charge dans sa mémoire locale une portion de l'image à transposer depuis le disque. Durant l'étape 2, chaque tuile copie sa portion de l'image source vers la mémoire locale de la tuile de destination en accord avec la transposition. Durant l'étape 3, chaque tuile s'occupe de l'affichage de la portion de l'image transposée stockée dans sa mémoire locale. Durant l'étape 4, chaque tuile recopie l'image transposée depuis sa mémoire locale vers le disque. L'application génère des communications inter-tuile à chacune de ses étapes. Dans notre architecture, les accès au disque ainsi qu'au périphérique d'affichage se font via une tuile localisée dans un coin de la grille. Les étapes 1, 3 et 4 génèrent des communications de type hot-spot alors que l'étape 2 génère des communications plus homogènes.

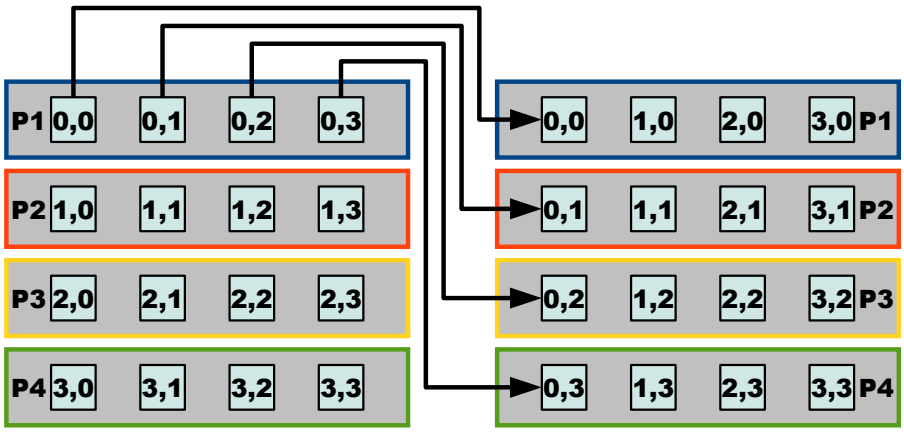

Figure 6.9 – Transposition d'une matrice 4×4 stockée dans la mémoire locale de 4 processeurs

6.4.1.2 FFT

Le calcul de FFT est un programme issu du benchmark SPLASH-2 [Woo *et al.*, 1995]. Comme nous l'avons vu dans le chapitre 2, SPLASH-2 a été développé dans le but d'étudier les architectures à mémoire partagée, qu'elle soit distribuée ou non.

Ce programme est basé sur l'algorithme radix-$\sqrt{n}$ de calcul de FFT proposé par Bailey [1989]. La transformée est effectuée sur un jeu de n points complexes organisés sous forme d'une matrice $\sqrt{n} \times \sqrt{n}$. Dans une configuration à t tuiles, chacune

d'elles se voit associer $\frac{\sqrt{n}}{t}$ lignes de cette matrice à stocker dans sa mémoire locale. Cette algorithme est découpé en six étapes :

1. Transposition de la matrice
2. Transformées locales
3. Application de la racine de l'unité
4. Transposition de la matrice
5. Transformées locales
6. Transposition de la matrice

Les étapes 1, 4 et 6 réalisent une transposition de la matrice. Lors de cette opération, chaque tuile communique avec l'ensemble des autres tuiles suivant le schéma illustré par la Figure 6.9.

L'étape 3 applique la racine de l'unité à la matrice, chaque racine est stockée dans la mémoire locale de la tuile qui en a besoin. Cette étape ne génère donc pas de communication en dehors de la tuile.

Les étapes 2 et 5 réalisent la FFT uniquement par ligne. Le calcul classique d'une FFT comprend la partie (a) par ligne et la partie (b) par colonne illustré par la Figure 6.10. L'utilisation d'une transposition lors de l'étape 4 permet de répéter la partie (a) à la place de la partie (b) de la FFT. Il est ainsi possible de tirer parti du stockage en ligne de la matrice dans les mémoires locales. Les étapes 2 et 5 ne génèrent donc pas de communication externes.

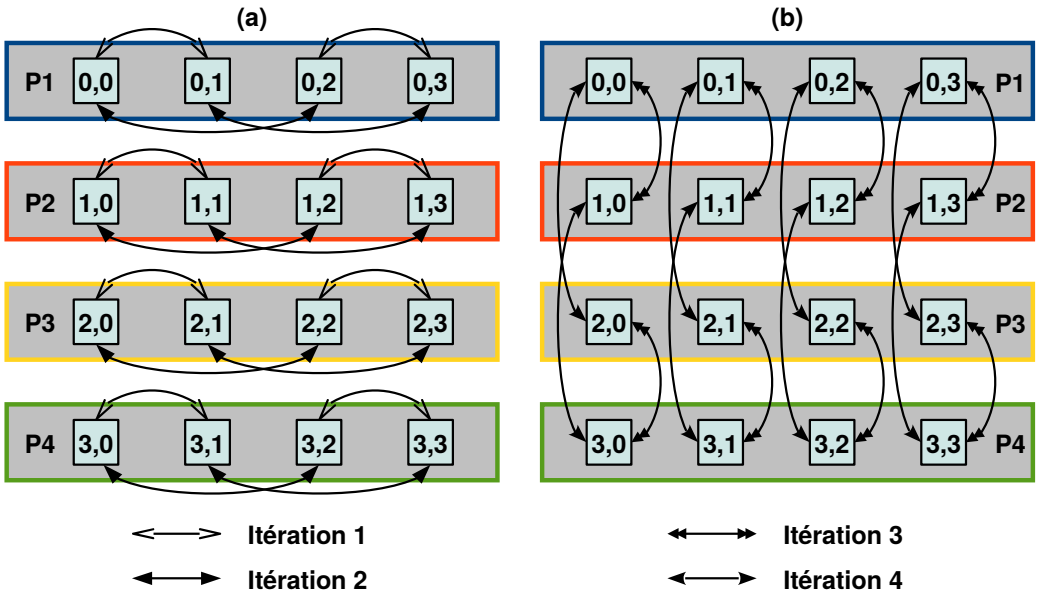

Figure 6.10 – Combinaisons réalisées lors du calcul de la FFT d'une matrice 4×4 stockée par lignes dans la mémoire locale de 4 processeurs

6.4.2 Résultats

La Figure 6.11 présente le temps d'exécution complet des applications FFT et transpose pour quatre configurations d'un CMP de 8 × 8 tuiles. La première utilise un réseau filaire de type grille alors que les trois autres utilisent l'architecture WiNoCoD. Les trois configurations utilisant WiNoCoD diffèrent par le nombre de nœuds RF utilisés et donc le nombre de tuiles qui en dépendent.

Nous constatons que pour cette taille de CMP, les meilleurs résultats sont obtenus avec l'architecture utilisant uniquement DSPIN. De plus, plus le nombre de nœuds constituant le NoC RF augmente, plus le temps d'exécution augmente. Ainsi, pour les NoC RF avec un nombre de nœuds N égal à 2, 4 ou 8, les temps d'exécution par rapport à DSPIN augmentent respectivement de 20 %, 35 % et 49 %. Ces résultats sont en accord avec ceux de la section 6.2 puisque pour les NoC RF de 4 et 8 nœuds, la latence par rapport a une grille augmentait respectivement de 26 % et 42 %.

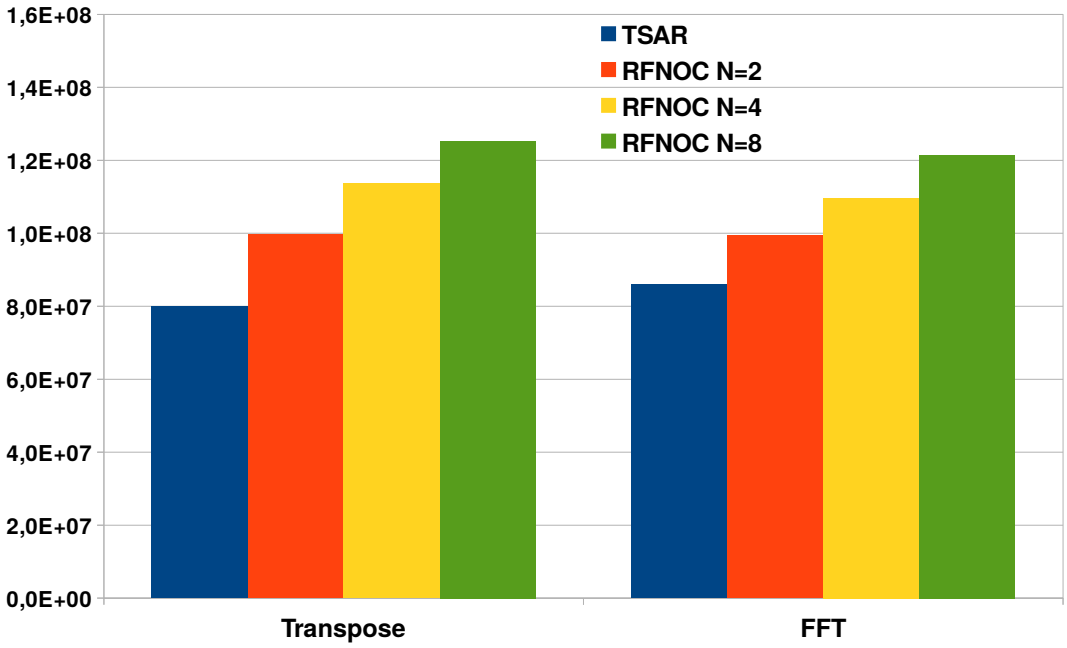

Figure 6.11 – Temps d'exécution total sur un CMP de 8 × 8 tuiles

6.4.3 Analyse

Dans cette section nous avons cherché à évaluer l'architecture WiNoCoD de la manière la plus réaliste possible en exécutant une application directement sur le modèle SystemC de l'architecture. Il apparait que pour un CMP de 8 × 8 tuiles, l'utilisation de WiNoCoD dans sa configuration actuelle n'apporte pas de gain. Ces résultats confirment ceux de la section 6.2 qui montraient déjà que les performances de Wi-

NoCoD ne devenaient meilleurs que celles du réseau filaire que pour les CMP de 16×16 tuiles et plus.

6.5 Évaluation via un trafic synthétique

Tout comme dans la section précédente, les expérimentations présentées dans cette section se basent sur la simulation du modèle SystemC CABA de l'architecture. Cependant, contrairement aux tuiles utilisées dans la section précédente, celles utilisées ici ne comprennent que des générateurs de trafic et des cibles. Nous pouvons ainsi évaluer de plus grosses architectures que dans la section précédente. De plus, nous avons tout particulièrement cherché à évaluer les apports de l'allocation dynamique des ressources de communication. Pour ce faire, nous avons testé trois variantes de l'algorithme d'allocation dynamique.

6.5.1 Protocole expérimental

Pour réaliser les expérimentations de cette section, dans chaque tuile les composants classiques ont été remplacés par deux composants. Un générateur de commande d'une part et une cible synthétique d'autre part. Nous avons ensuite testé plusieurs schémas de communications et fait varier la probabilité d'émission de chaque grappe de manière à atteindre un taux d'émission donné. Ce taux d'émission est exprimé en nombre de commandes émises par grappe par cycle RF. En fonction du schéma de communication testé, les commandes peuvent ou non générer de la part de la cible une réponse qui devra alors circuler à son tour sur le réseau.

Nous avons comparé les performances de cinq configurations de NoC pour un CMP de 32×32 tuiles :

- DSPIN : utilisation exclusive du mesh ;
- WiNoCoD statique : NoC sans reconfiguration dynamique ;
- WiNoCoD dynamique A : le besoin d'un cluster est directement proportionnel au pourcentage de remplissage de sa FIFO d'émission ;
- WiNoCoD dynamique B : le besoin d'un cluster correspond au nombre de FLIT en attente dans sa FIFO d'émission ;
- WiNoCoD dynamique C : le besoin d'un cluster correspond au nombre de FLIT en attente dans sa FIFO d'émission avec une valeur minimale de 1.

La version A de l'algorithme d'allocation dynamique correspond à l'implémentation la plus simple. Chaque nœud RF transmet le pourcentage de remplissage de la FIFO d'émission de son arbitre. Une FIFO ne peut avoir qu'un certain nombre d'états de remplissage correspondant au nombre de bits utilisés pour transmettre

cette information sur le canal de service. Ainsi, deux FIFO contenant respectivement un et deux paquets peuvent avoir le même état de remplissage. Si ce fonctionnement n'est pas gênant quand les FIFO commencent à se remplir, cela peut diminuer la réactivité de la reconfiguration dynamique pour de faibles taux de remplissage.

La version B de l'algorithme d'allocation dynamique remplace donc le pourcentage de remplissage par le nombre exact de paquets contenus dans chaque FIFO. Cependant, le nombre de bits utilisés pour transmettre cette information ne permet pas forcément de coder toutes les valeurs. L'arbitre transmet donc soit le nombre exact de paquets en attente, soit la valeur `max` indiquant qu'il y au minimum N paquets en attente d'émission dans sa FIFO. Cependant cette version, tout comme la A, peut allouer trop de canaux à un nœud RF dans certains cas. Par exemple, un nœud RF dont la FIFO ne contient qu'un paquet peut se voir attribuer tous les canaux si toutes les autres FIFO sont vides.

La version C de l'algorithme d'allocation dynamique ajoute à la version B un état de remplissage minimum égal à `1`. Cette modification permet de continuer à allouer des canaux aux nœuds RF ayant une FIFO d'émission vide tant que la demande globale est faible. Nous augmentons ainsi la probabilité qu'un nœud RF puisse émettre un paquet sur le NoC RF dés qu'il le reçoit du réseau filaire.

De plus, les configurations utilisant l'architecture WiNoCoD présentent les caractéristiques communes suivantes :

— modulation 16-QAM
— 16 nœuds RF
— 16 canaux de 30 sous-bandes chacun pour les données
— 1 canal de service de 32 sous-bandes

6.5.2 Résultats

6.5.2.1 Trafic homogène

La figure 6.12 présente la latence moyenne d'une transaction dans le cas d'un trafic homogène. Chaque tuile émet une requête vers une autre tuile choisie aléatoirement. Pour un faible taux d'émission, la latence moyenne d'une transaction est de 116 cycles pour les configurations utilisant WiNoCoD et de 140 cycles avec le mesh, soit un gain de 17 %. Quand le taux d'émission augmente, les configurations utilisant le NoC RF saturent aux alentours de 0,5 transaction/grappe/cycle-RF alors que le mesh garde une latence constante. La saturation du mesh n'est pas représentée sur la courbe, ses premiers effets arrivent aux alentours de 4 transactions/grappe/cycle-RF. Le NoC RF a la capacité de transmettre l'équivalent d'un FLIT par grappe par

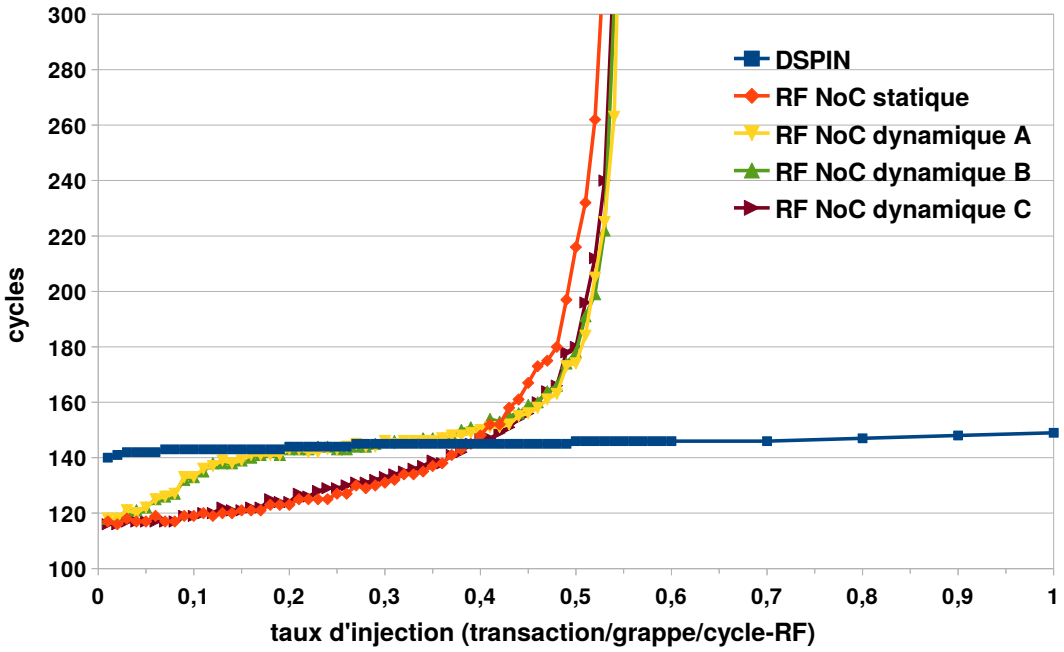

Figure 6.12 – Latence moyenne des communications 1 vers 1 pour un CMP 32 × 32

cycle RF, or une transaction correspond à un couple commande/réponse générant ainsi 2 FLIT. Il est donc normal qu'il commence à saturer à partir de 0,5.

En regardant plus en détail les différentes configurations de WiNoCoD, nous pouvons constater deux points. D'une part, l'allocation dynamique des canaux de communication permet de repousser légèrement le seuil de saturation du réseau. D'autre part, pour un taux d'injection de moins de 0,4 transaction/grappe/cycle-RF, on s'aperçoit que les versions A et B de l'algorithme d'allocation dynamique provoquent une dégradation des performances pour un trafic homogène par rapport à la version statique. Cet inconvénient est éliminé par la version C de l'algorithme d'allocation dynamique qui permet d'avoir les mêmes performances que la version statique.

6.5.2.2 Trafic hétérogène

La figure 6.13 présente la latence moyenne des communications dans le cas d'un trafic hétérogène. Toutes les tuiles du CMP initient une transaction à destination d'une seule et même tuile, générant ainsi un point de contention, on parle alors de hot-spot. Dans notre cas, la tuile de destination se trouve dans un coin du CMP.

Pour un faible taux d'émission, la latence moyenne d'une transaction est de 136 cycles pour les différentes configurations de WiNoCoD et de 199 cycles pour le mesh, soit un gain de 32 %. Lorsque le taux d'injection augmente, le NoC RF statique sature à partir d'un taux d'injection de 0,06 alors que les autres configurations, y compris le mesh, saturent à partir d'un taux d'injection de 0,4. La proximité des

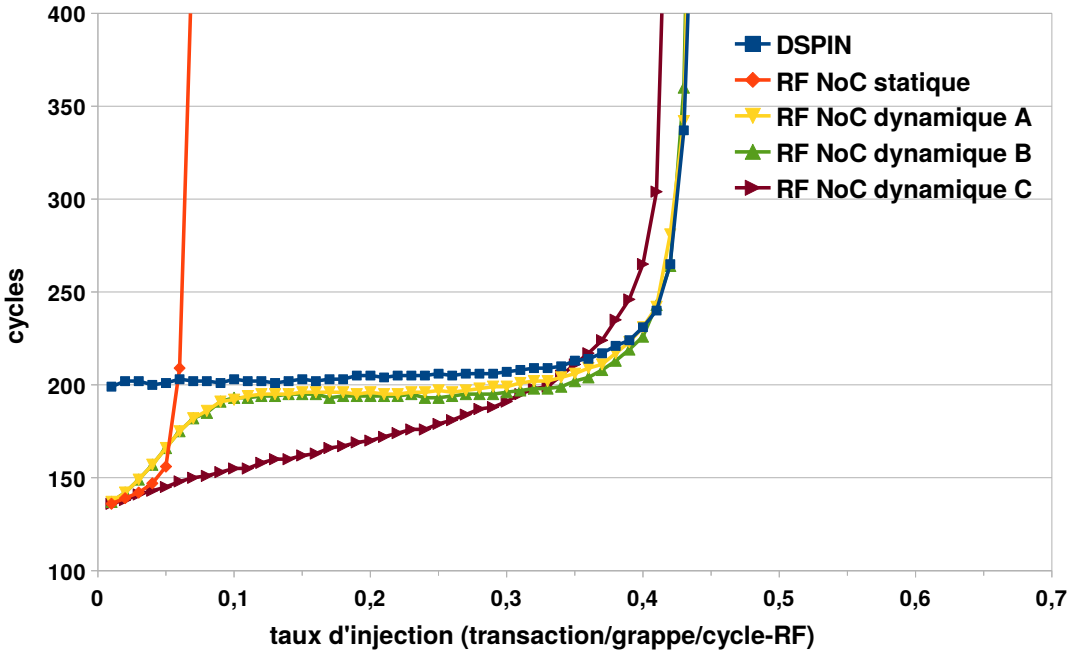

Figure 6.13 – Latence moyenne des communications N vers 1 pour un CMP 32 × 32

résultats pour les différentes versions du NoC RF et de DSPIN met en évidence que cette saturation vient de la tuile hot-spot et du routeur du mesh qui lui est associé.

Concernant les différentes versions de l'algorithme d'allocation dynamique, nous constatons que les versions A et B offrent des performances similaires. La version C présente l'inconvénient de saturer environ 0,02 transaction/grappe/cycle-RF plus tôt que les versions A et B mais elle est plus performante tant que le réseau n'est pas saturé. Ainsi, alors que la latence des versions A et B devient équivalente à celle du mesh à partir de 0,1 transaction/grappe/cycle-RF, la latence de la version C ne devient équivalente à celle du mesh qu'à partir d'un taux d'injection de 0,3.

6.5.2.3 Broadcast avec réponse

La figure 6.14 présente la latence moyenne des communications quand elles sont toutes de type broadcast. Toutes les tuiles du CMP peuvent initier un broadcast. Une transaction de broadcast est terminée quand la tuile émettrice du broadcast a reçu une réponse de chacune des autres tuiles de l'architecture.

Pour ce type de communications, les différentes configurations du NoC RF commencent à saturer à partir de 0,001 transaction/grappe/cycle-RF alors que DSPIN ne commence à saturer qu'aux alentours de 0,010 transaction/grappe/cycle-RF. De plus, même pour un très faible taux d'injection, la latence du NoC RF est plus grande que celle de DSPIN. Finalement, de part l'émission homogène des réponses d'acquittement, la version dynamique du NoC RF ne permet pas d'améliorer les performances par rapport à la version statique.

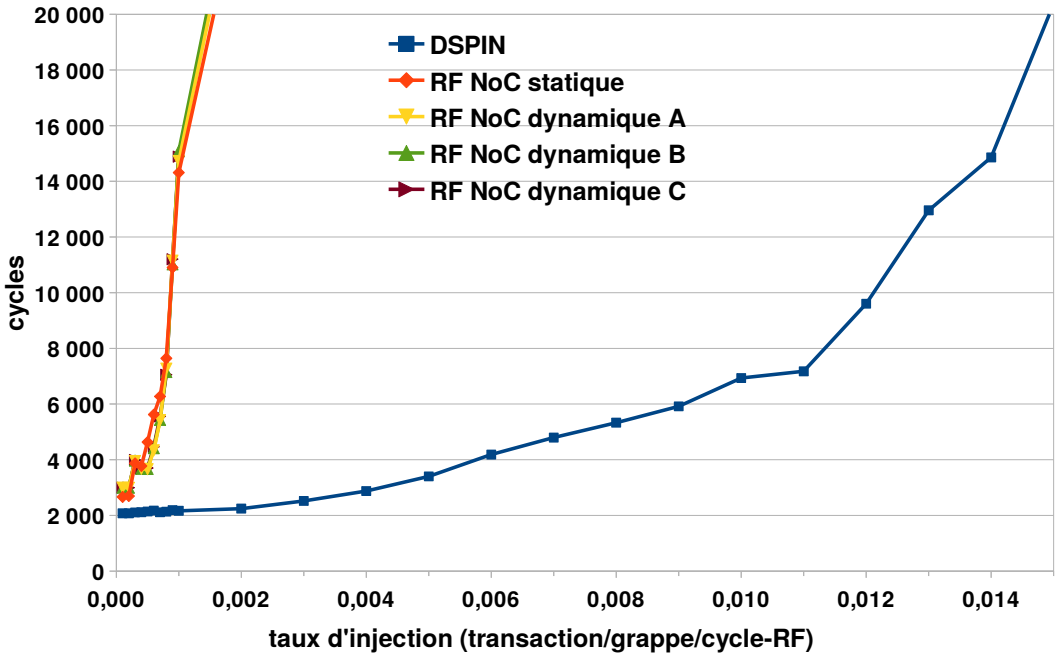

Figure 6.14 – Latence des communications de type broadcast avec réponse d'acquittement pour un CMP 32 × 32

6.5.2.4 Broadcast sans réponse

La Figure 6.14 présente elle aussi la latence moyenne des communications quand elles sont toutes de type broadcast. Cependant, à la différence de la Figure 6.15, le broadcast est ici à sens unique, il ne génère pas de réponse de la part des autres tuiles de l'architecture.

Pour ce type de communication, nous constatons que les performances du NoC RF et de DSPIN sont proches jusqu'à un taux d'injection de 0,05. Cependant alors que le NoC RF sature pour un taux d'injection compris entre 0,10 et 0,13, DSPIN sature pour un taux d'injection compris entre 0,20 et 0,25.

6.5.2.5 Trafic mixte

Finalement, la Figure 6.16 présente la latence moyenne des communications pour un profil mixte. Ce profil rassemble différents schémas de communication dans le but de se rapprocher du trafic que génèrerait l'exécution d'une vraie application sur l'architecture. Nous avons retenu la répartition des communications suivante :

— 5 % de broadcast [Jerger *et al.*, 2008]
— 10 % à destination du hot-spot [Chiu, 2000]
— 85 % de trafic homogène

Pour ce schéma de communications, la latence de DSPIN est de l'ordre de 150 cycles et il ne commence à saturer qu'à partir d'un taux d'injection de 2 transactions par

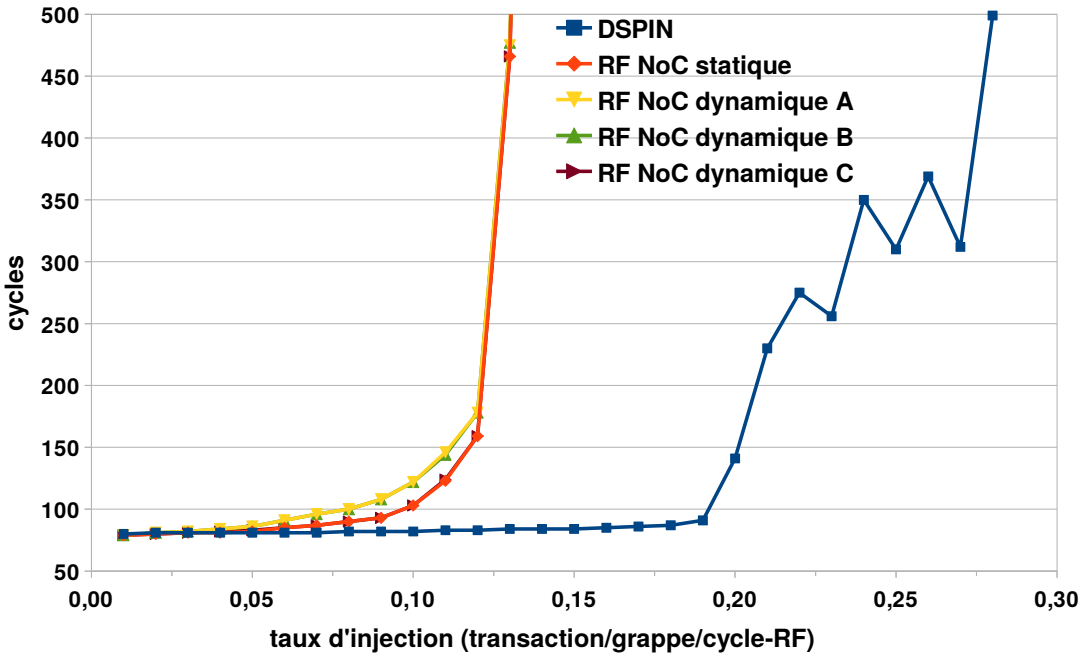

Figure 6.15 – Latence des communications de type broadcast sans réponse pour un CMP 32 × 32

grappe par cycle-RF. Le NoC RF offre une plus faible latence, de l'ordre de 130 cycles, pour un faible taux d'injection permettant ainsi une diminution de la latence de 13 %. Cependant, les différentes versions du NoC RF réagissent différemment face à l'augmentation du taux d'injection. Ainsi la version statique offre une plus faible latence que DSPIN jusqu'à un taux d'injection de 0,2 et sature aux alentours de 0,3 transaction par grappe par cycle-RF. Les versions dynamiques A et B présentent une augmentation rapide de la latence quand le taux d'injection passe de 0,01 à 0,10 puis, la latence reste constante jusqu'à un taux d'injection de 0,5 où les deux réseaux finissent par saturer. Finalement, la version C de l'algorithme d'allocation dynamique permet d'avoir les mêmes performances que la version statique quand le taux d'injection est en dessous de 0,3. Elle permet aussi de maintenir cette augmentation lente de la latence jusqu'au taux d'injection de 0,5 où le NoC RF sature quelque soit la version de l'algorithme d'allocation dynamique utilisée.

6.5.3 Analyse

Dans cette section, nous avons soumis les différentes configurations de NoC à plusieurs profils de communication en utilisant des générateurs de trafic. Le but de cette expérimentation était d'évaluer le gain permis par la reconfigurabilité de notre réseau tout en le comparant à un réseau filaire classique. Pour ce faire nous avons comparé une version statique de notre NoC RF à plusieurs versions de notre algorithme de reconfiguration dynamique ainsi qu'au réseau DSPIN.

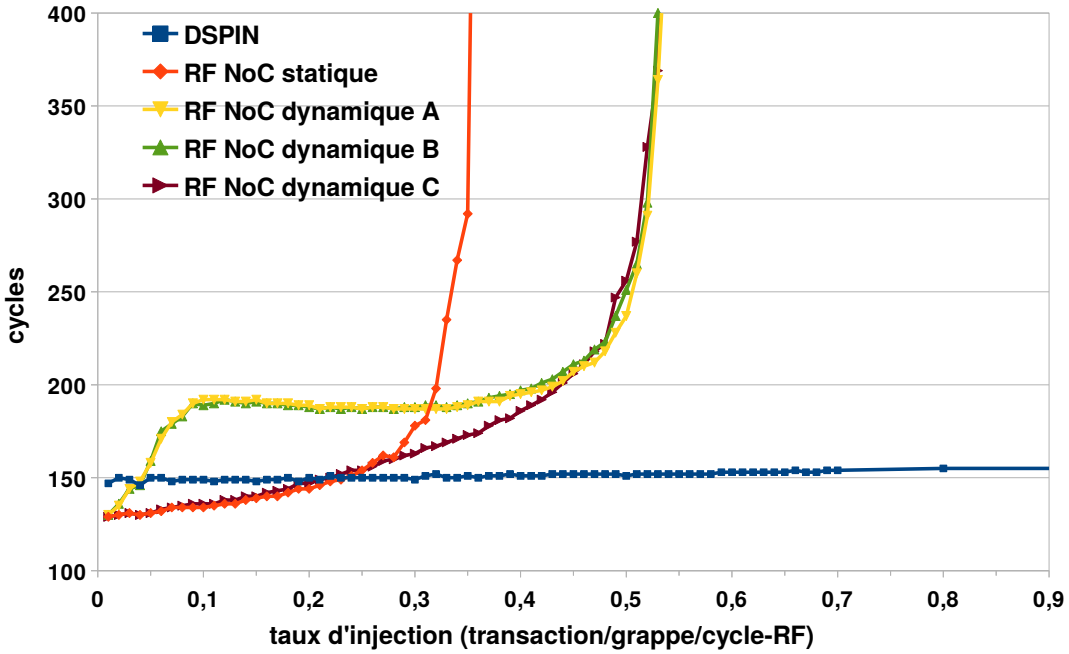

Figure 6.16 – Latence moyenne des communications pour un profil mixte dans un CMP 32×32

L'étude du schéma de communication homogène confirme que l'utilisation du NoC RF permet une baisse de 17 % de la latence pour les taux d'injections minimums testés. Ces résultats viennent ainsi confirmer ceux de la section 6.2 qui montraient que l'utilisation du NoC RF permettait une diminution de la latence de 18 % pour ce type de communication. Cependant, la bande passante totale du NoC RF étant inférieure à celle du réseau filaire, il présente l'inconvénient de saturer plus vite dans le cadre d'un trafic homogène.

L'étude du schéma de communication hétérogène nous a permis de mettre en évidence l'apport de l'algorithme de reconfiguration dynamique. Le NoC RF reconfigurable sature pour un taux d'injection $10\times$ supérieur au seuil de saturation de la version statique. De plus, la meilleure version de l'algorithme d'allocation dynamique permet d'obtenir une latence inférieure à celle du réseau filaire jusqu'à 85 % du seuil de saturation.

L'étude du broadcast nous a montré que si le NoC RF facilite la diffusion d'un message de broadcast d'un point de vue théorique, la version actuelle de WiNoCoD présente certaines limitations. Tout d'abord, si le message de broadcast génère un message d'acquittement de la part des destinataires, le trafic généré est alors très important et la bande passante du NoC RF se révèle alors très insuffisante par rapport à celle du réseau filaire. De plus, même pour un message de broadcast ne générant pas d'acquittement, le NoC RF offre les mêmes performances que le réseau filaire pour un faible taux d'injection mais sature beaucoup plus vite. Si la dynamicité de la diffusion d'un message de broadcast n'a pas été prise en compte lors de

l'évaluation théorique, son caractère fortement parallèle avantage le réseau filaire.

Finalement, nous avons étudié un scénario mêlant communications homogènes, hétérogènes et broadcast. Ce scénario nous montre que le NoC RF permet une amélioration de 13 % de la latence des communications par rapport au réseau filaire tant que le taux d'injection est inférieur à 0,25 transaction par grappe et par cycle RF. Au delà de ce taux d'injection, le NoC RF commence à saturer et le réseau filaire prend l'avantage grâce à sa plus grande bande passante.

L'étude de ces différents scénarios de communication montre que le NoC RF permet de diminuer la latence des communications dans un CMP. Elle met aussi en évidence l'importance du choix de l'algorithme d'allocation dynamique. Ainsi les algorithmes A et B permettent de repousser le seuil de saturation du réseau mais dégradent les performances pour les faibles taux d'injections. Quant à l'algorithme C, il permet de tirer le meilleur parti du NoC RF quelque soit le taux d'injection. Pour finir, les performances du réseau filaire lorsque le taux d'injection augmente montrent l'intérêt de disposer d'une grande bande passante.

6.6 Conclusion

Dans ce chapitre nous avons évalué l'architecture WiNoCoD selon différents critères de manière à déterminer sa validité par rapport à notre problématique :
Comment allouer dynamiquement les ressources de communication dans un NoC de manière cohérente avec l'amélioration de la latence et du débit ainsi que le support du broadcast ?

Nous avons tout d'abord identifié les configurations de NoC RF pouvant améliorer les performances par rapport à un réseau filaire classique pour différentes tailles de CMP. Puis, nous avons évalué le coup matériel des configurations retenues en termes de surface et de consommation. Finalement, nous avons utilisé le modèle SystemC de l'architecture pour évaluer ses performances par rapport à un CMP utilisant uniquement un réseau filaire de type grille. Pour cette dernière étape, nous avons tout d'abord utilisé des générateurs de trafic synthétiques, puis exécuté des applications directement sur le modèle de l'architecture.

Dans la section 6.2, nous avons évalué les performances intrinsèques de WiNoCoD. Le but de cette section était d'établir les configurations optimales du NoC RF en fonction de la taille du CMP sans tenir compte des mécanismes de reconfiguration dynamique. Il en ressort que, pour les communications point à point, le NoC RF offre de meilleures performances que le réseau filaire pour des CMP de 256 tuiles et plus. Pour l'émission des messages de broadcast, le NoC RF offre des performances équivalentes ou meilleures quelque soit la taille du CMP. De plus, nous avons pu

établir que la configuration optimale pour le NoC RF était de disposer d'autant de canaux que de points d'accès.

Dans la section 6.3, nous avons évalué la faisabilité des configurations retenues dans la section 6.2 en terme de consommation et de surface. Du point de vue de la surface, les configurations jusqu'à 2 048 cœurs sont réalisables avec une finesse de gravure de 22 nm. La configuration à 4 096 cœurs nécessite quant à elle une finesse de gravure de 14 nm. Cependant, le ratio entre la surface du NoC et celle du CMP est aussi à prendre en compte et un ratio de 10 % parait raisonnable selon l'état de l'art. Ainsi, si toutes les configurations sont réalisables, seules les configurations à 1 024, 2 048 et 4 096 ont un ratio inférieur à 10 %. Du point de vue de la consommation, toutes les configurations sont réalisables avec une finesse de gravure de 22 nm. Le ratio entre la consommation du NoC et celle du CMP retenu dans l'état de l'art est de 20 %. Seules les configurations à 1 024, 2 048 et 4 096 ont un ratio inférieur à 20 %. Il en ressort donc que les configurations à 1 024 et 2 048 cœurs sont réalisables en 22 nm. Finalement, la configuration à 4 096 cœurs est réalisable avec la finesse de gravure en cours de déploiement, le 14 nm [Nalamalpu *et al.*, 2015].

Dans la section 6.4, nous avons évalué les performances de WiNoCoD en exécutant des applications directement sur le modèle de l'architecture. Si cette méthode présente l'avantage d'être celle qui se rapproche le plus d'une utilisation réelle, c'est aussi la plus longue à mettre en place. Ainsi, en raison de la complexité de la répartition de l'application sur le modèle et du temps de simulation, nous avons dû limiter cette approche à un CMP de 8×8 tuiles. Or, comme le montraient les résultats de la section 6.2, le NoC purement filaire est plus performant que WiNoCoD pour cette taille de CMP. Nous avons donc dû mettre en place la méthode d'évaluation présentée dans la section 6.5 afin d'évaluer de plus grosses architectures.

Dans la section 6.5, nous avons évalué les performances d'un CMP utilisant WiNoCoD par rapport au même CMP utilisant uniquement une grille 2D filaire en utilisant différents scénarios de communication. Le CMP évalué comprend 1 024 tuiles (32×32) ce qui correspond à la plus grande configuration étudiée dans les sections précédentes. Il en ressort que le NoC RF permet de diminuer la latence des communications tant que le taux d'injection ne dépasse pas un certain seuil dépendant du profil de communication. Cette diminution est de 17 % pour le profil de communications homogène et de 13 % pour le profil de communication mixte. Au-delà de ce seuil, le NoC RF sature alors que la grande bande passante du réseau filaire lui permet de ne saturer que beaucoup plus tard. Il est cependant possible de retarder ce seuil de saturation du NoC RF grâce à l'allocation dynamique des canaux de communications. Il serait possible de repousser encore plus ce seuil grâce au changement dynamique de modulation, nous n'avons cependant pas évalué cette possibilité ici puisque les plateformes utilisent toutes une modulation 16-QAM. Finalement, nous avons testé différentes variantes du NoC RF. Nous avons ainsi pu

mettre en évidence les apports de la reconfigurabilité du NoC RF par rapport à une version statique. Cela nous a aussi permis de mettre en évidence l'importance du choix de l'algorithme d'allocation dynamique. Celui-ci doit à la fois permettre au réseau de s'adapter aux variations de trafic et ne pas dégrader les performances par rapport à une version statique lorsque le réseau est peu sollicité.

Ces résultats montrent que la solution proposée répond à notre problématique mais que cette réponse pourrait être améliorée. Ainsi le NoC RF permet la mise en place d'un mécanisme d'allocation dynamique efficace et supporte intrinsèquement le broadcast. De plus, il permet de diminuer la latence des communications longue distance et d'augmenter la bande passante totale de la puce. Cependant, son interfaçage avec le réseau filaire n'exploite pas pleinement ces caractéristiques pour tous les profils de communication. Deux pistes pourraient être explorées pour exploiter pleinement le NoC RF proposé. Une première piste serait de rendre le choix entre le réseau filaire et le NoC RF plus flexible. Nous pourrions par exemple faire ce choix en fonction de la distance et permettre à deux tuiles appartenant à deux grappes différentes de communiquer via le réseau filaire si elles sont suffisamment proches. Il serait aussi possible de conditionner ce choix par le taux d'utilisation du NoC RF. Une deuxième piste serait de mettre en place une hiérarchie mémoire mieux adaptée à la hiérarchie du réseau. L'introduction d'un cache L3 partagé par toutes les tuiles d'un même nœuds RF pourrait ainsi permettre de diminuer le nombre de messages circulant sur le NoC RF.

Chapitre 7

Conclusion et perspectives

Sommaire

7.1 Conclusion

Durant cette thèse, nous avons tenté de répondre à la problématique suivante : Comment allouer dynamiquement les ressources de communication dans un *Network on Chip* (NoC) de manière cohérente avec l'amélioration de la latence et du débit ainsi que le support du broadcast ?

Ainsi nous avons tout d'abord présenté les raisons ayant entrainées le développement des architectures *Chip Multiprocessor* (CMP). L'avènement de ces architectures parallèles a pu avoir lieu pour deux raisons. La première raison provient du logiciel : les applications exécutées présentent du parallélisme de tache à divers degré. La deuxième raison provient du matériel lui même et de son développement : il est de plus en plus difficile d'améliorer les performances séquentielles. Ces constations ont entrainée le développement des premières générations de CMP que nous avons choisi d'appeler multi-cœur. Il s'agit de processeurs utilisant des médiums d'interconnexion classiques tels que les bus, crossbar ou anneaux, limitant de ce fait le processeur à un nombre de cœurs de l'ordre de la dizaine. C'est pour pouvoir continuer à augmenter le nombre de cœurs que les premiers NoC ont été proposés, ces médiums d'interconnexion visant à intégrer plusieurs centaines ou milliers de cœurs sur une même puce, on parle alors de many-coeurs.

Après avoir présenté les premiers NoC constitués de réseau de routeurs filaires 2D, nous avons identifié leurs différentes limitations. Il s'agit du passage à l'échelle de la bande passante et de la latence lorsque le nombre de cœurs augmente, du support des messages de type multicast et broadcast et pour finir de la possibilité d'adapter dynamiquement le réseau aux besoins de communications. Nous avons ensuite présenté différentes solutions de l'état de l'art adressant une ou plusieurs de ces limitations. La première catégorie de solution consiste à conserver la technologies CMOS 2D et à baser les améliorations sur des modifications de l'architecture du réseau. Cependant ces solutions sont elles aussi limitées en terme de passage à l'échelle, soit en terme de performances, soit en terme de coût matériel. Viennent ensuite les solutions basées sur les technologies d'intégration 3D. Si ces solutions permettent de diminuer les distances entre les nœuds du réseau, elles se révèlent cependant peu favorables au support des messages de type broadcast et à la mise en place de mécanismes de reconfiguration, tout comme les solutions 2D filaires. Finalement, l'utilisation de nouvelles technologies telles que l'optique et la radio fréquence permet de développer de nouvelles architectures réseaux impossible à mettre en œuvre avec les technologies filaires 2D et 3D.

Parmi toutes les solutions présentées dans l'état de l'art, aucune ne semble mettre en place un mécanisme de reconfiguration dynamique tout en tenant compte des trois autres critères d'amélioration des performances. Nous avons donc proposé WiNoCoD : un NoC RF hiérarchique reconfigurable dynamiquement grâce à l'OFDMA,

développé dans le cadre du projet ANR du même nom. L'utilisation combinée de la RF et du filaire dans un NoC hiérarchique permet de tirer pleinement partie des avantages de ces deux types de réseau. L'utilisation de l'OFDMA, permet de diviser la bande RF en sous-bandes de manière à créer des canaux de communication. Ces canaux de communications sont ensuite partagés entre les différents nœuds du NoC grâce aux mécanismes d'allocations dynamique. Ces mécanismes reposent sur un algorithme d'allocation distribué utilisant les besoins des différents nœuds du réseau pour allouer les canaux de communication. Le support du broadcast fourni par le NoC RF permet à un nœud de communiquer ses besoins à l'ensemble du réseau pour le même coût que s'il avait dû les communiquer à un seul. De plus, l'algorithme étant distribué dans chaque nœud du réseau, les messages de reconfiguration ne sont pas nécessaires. Si la distribution de l'algorithme entraine la réplication des composants matériels permettant son exécution, elle permet donc de diminuer le coût en communication des mécanismes de reconfiguration.

Pour évaluer les performances de notre solution, nous avons développé un modèle de l'architecture basé sur SystemC, un ensemble de classes C++ permettant de faire de la description de matériel. De plus ce modèle s'intègre dans SoClib, une bibliothèque de composants matériels développée grâce à SystemC. Nous avons utilisé ce modèle pour valider fonctionnellement l'architecture et évaluer ses performances.

Ainsi l'évaluation de notre solution s'est déroulée en plusieurs phases. Nous avons tout d'abord cherché à évaluer les gains théoriques de la solution proposée. Cette première phase nous a montré que notre solution devenait plus performante qu'une simple grille 2D pour les CMP de 256 tuiles et plus. Nous avons ensuite estimé le coût matériel d'un CMP utilisant le NoC RF WiNoCoD. Cette deuxième phase nous a montré que la réalisation physique de notre solution devenait pertinente en termes de surface et de consommation pour les CMP de 256 tuiles et plus. Nous avons alors tenté d'évaluer les performances de l'architecture proposée en exécutant des applications directement sur le modèle SystemC développé. Cependant, nous n'avons pu augmenter le nombre de tuiles d'une telle plateforme au delà de 64. Finalement, nous avons donc développé des générateur de trafic nous permettant de développer et d'évaluer des CMP comportant jusqu'à 1 024 tuiles. Nous avons alors pu soumettre notre architecture à plusieurs profils de communications dont un profil mixte, composé de communications point à point homogènes, de zone de contention et de broadcast. Cette dernière phase d'évaluation nous a permis de montrer que pour un CMP de 1 024 tuiles, notre solution permettait un gain en performance de l'ordre de 13 % lorsque le taux d'injection restait modéré. Cependant elle nous a aussi montré que notre architecture saturait plus vite qu'une grille 2D filaire ce qui nous a permis d'identifier une partie des limitations de l'architecture et donc les perspectives d'amélioration que nous présentons dans la section suivante.

7.2 Perspectives

7.2.1 Court terme

Dans le cadre de cette thèse, nous avons cherché à évaluer les performances du NoC RF WiNoCoD. Si l'évaluation théorique traite toutes les dimensions envisagées, l'évaluation utilisant le modèle SystemC présente certaines limitations concernant le nombre de cœurs. Si, avec le trafic synthétique, nous avons pu évaluer une architecture de 1 024 tuiles, équivalente à 4 096 cœurs, nous n'avons exécuté que deux applications sur une plateforme de 64 tuiles. Il serait donc intéressant de pousser l'évaluation dans ce sens. D'une part, en exécutant les applications sur des plateformes de 256, 512 et 1 024 tuiles. D'autre part, en ajoutant d'autres applications de manière à évaluer une plus grande variété de profils de communication.

7.2.2 Moyen terme

L'architecture que nous avons présentée dans ce manuscrit présente un découpage hiérarchique strict. Une première piste à explorer serait d'assouplir ce découpage hiérarchique. Ainsi, nous avons pu voir, notamment dans l'état de l'art, que les réseaux filaires étaient mieux adaptés aux communications locales. L'obligation pour les communications entre deux grappes de passer par la RF peut dégrader les performances si deux tuiles communiquent ensemble en étant dans deux grappes différentes alors qu'elles se trouvent physiquement proches. Il serait donc intéressant de rendre possible le choix du réseau à utiliser en fonction de la distance entre la source et la destination. À l'opposée, une seconde piste serait d'aller plus loin dans le découpage hiérarchique de l'architecture en mettant en place une hiérarchie mémoire correspondant à la hiérarchie du réseau. L'introduction d'un cache L3 partagé par toutes les tuiles d'un même nœuds RF pourrait ainsi permettre de diminuer le nombre de messages circulant sur le NoC RF tout en exploitant ses capacités de broadcast pour le maintient de la cohérence.

7.2.3 Long terme

Le NoC RF WiNoCoD n'a pour l'instant été envisagé que dans le cadre des architectures many-cœurs sans autres contraintes que les performances. Il serait intéressant d'étudier son potentiel dans d'autres domaines.

Il pourrait par exemple être utilisé dans les FPGA. Ce type de puce permet de réaliser des *System on Chip* (SoC) pouvant être reconfiguré pour disposer des composants matériels les plus adaptés à l'application exécutée. Il serait donc intéressant de

disposer d'un réseau reconfigurable dynamiquement pour faire communiquer ces différents composant de manière à rendre ces architecture encore plus adaptables.

Une autre piste serait de l'utiliser pour réaliser des architecture modulables. D'une part, cette approche pourrait permettre de morceler la fabrication de grosses puces pour en réduire les coût. D'autre part, cela pourrait permettre de proposer des SoC personnalisables en fonction des besoins.

De plus, nous n'avons pas exploré le domaine des applications temps réel. Or, WiNoCoD pourrait présenter des caractéristiques intéressantes dans ce domaine. Dans la configuration présentée, n'importe quel canal peut être alloué à n'importe quel nœud. Il serait tout à fait possible de garantir un nombre minimal de canaux à un ou plusieurs nœuds, de manière à borner les temps de communication. Dans une approche plus souple, les nœuds du NoC RF pourraient se voir attribuer différentes priorités dans le processus d'allocation en fonction de la criticité du code exécuté.

Finalement, il est possible de multiplexer plusieurs réseaux filaires sur le NoC RF WiNoCoD. Cette caractéristique pourrait être exploitée pour sécuriser les applications exécutées. Une application s'exécutant sur une grappe utilisant un réseau n'aurait aucun moyen d'accéder aux communications échangées sur un autre réseau.

Liste des publications

Conférences internationales avec actes et comité de lecture

— **DSD 2014**, Drillet, F., Hamieh, M., Zerioul, L., Brière, A., Unlu, et al. *Flexible Radio Interface for NoC RF-Interconnect*. 17th Euromicro Conference on Digital System Design.
— **NOCS 2014**, Unlu, E., Hamieh, M., Moy, C., Ariaudo, M., Louët, et al. *An OFDMA based RF interconnect for massive multi-core processors*. 8th IEEE/ACM International Symposium on Networks-on-Chip.
— **DASIP 2015**, Azeem, M. M., Brière, A., Bouyer, M., Denoulet, J., Pêcheux, F., et al. *Interfacing SoCLib CABA models with NoCBench for NoC performance evaluation*. Conference on Design and Architectures for Signal and Image Processing.
— **GLSVLSI 2015**, Brière, A., Denoulet, J., Pinna, A., Granado, B., Pêcheux, F., Unlu, E., Louët, Y., et Moy, C. (2015b). *A Dynamically Reconfigurable RF NoC for Many-Core*, 25th edition of the ACM Great Lakes Symposium on VLSI

Revue nationale

— **TSI 34–1-2/2015**, Brière, A., Denoulet, J., Pinna, A., Granado, B., et Pêcheux, F. *Un réseau sur puce RF reconfigurable dynamiquement pour les many-cœurs*. Technique et Science Informatiques.

Conférence nationale avec comité de lecture

— **ComPAS'2014**, Brière, A., Denoulet, J., Pinna, A., Granado, B., Pêcheux, F., et al. *WiNoCoD : Un réseau d'interconnexion hiérarchique RF pour les MPSoC*. Conférence d'informatique en Parallélisme, Architecture et Système.

Colloques nationaux

— **GDR SOC-SIP 2014**, Brière, A., Denoulet, J., Pinna, A., Granado, B., et Pêcheux, F. (2014). *A hierarchical RF interconnect for MPSoC*. 9ème colloque du GDR SOC-SIP du CNRS
— **GDR SOC-SIP 2017**, Brière, A., Denoulet, J., Pinna, A., Granado, B., et Pêcheux, F. (2017). WiNoCoD : *A Dynamically Reconfigurable RF NoC for Many-Core*. 12ème colloque du GDR SOC-SIP du CNRS.

Bibliographie

ABADAL, S., MESTRES, A., MARTÍNEZ, R., ALARCÓN, E. et CABELLOS-APARICIO, A. (2015). Multicast on-chip traffic analysis targeting manycore NoC design. *In 2015 23rd Euromicro International Conference on Parallel, Distributed, and Network-Based Processing*, pages 370–378. IEEE.

ADRIAHANTENAINA, A., CHARLERY, H., GREINER, A., MORTIEZ, L. et ZEFERINO, C. A. (2003). SPIN : a scalable, packet switched, on-chip micro-network. *In Design, Automation and Test in Europe Conference and Exhibition, 2003*, pages 70–73. IEEE.

ASANOVIĆ, K., BODIK, R., CATANZARO, B. C., GEBIS, J. J., HUSBANDS, P., KEUTZER, K., PATTERSON, D. A., PLISHKER, W. L., SHALF, J., WILLIAMS, S. W. et YELICK, K. A. (2006). The landscape of parallel computing research : A view from berkeley. Rapport technique UCB/EECS-2006-183, EECS Department, University of California, Berkeley.

ASLOT, V., DOMEIKA, M., EIGENMANN, R., GAERTNER, G., JONES, W. B. et PARADY, B. (2001). SPEComp : A new benchmark suite for measuring parallel computer performance. *In International Workshop on OpenMP Applications and Tools*, pages 1–10. Springer.

AZEEM, M. M., BRIÈRE, A., BOUYER, M., DENOULET, J., PÊCHEUX, F., PINNA, A. et GRANADO, B. (2015). Interfacing SoCLib CABA models with NoCBench for NoC perfomance evaluation. *In Conference on Design and Architectures for Signal and Image Processing*, DASIP.

BAILEY, D. H. (1989). FFTs in external of hierarchical memory. *In Proceedings of the 1989 ACM/IEEE conference on Supercomputing*, pages 234–242. ACM.

BIENIA, C., KUMAR, S., SINGH, J. P. et LI, K. (2008). The PARSEC benchmark suite : Characterization and architectural implications. *In Proceedings of the 17th international conference on Parallel architectures and compilation techniques*, pages 72–81. ACM.

BIGLARBEGIAN, B., NEZHAD-AHMADI, M. R., FAKHARZADEH, M. et SAFAVI-NAEINI, S. (2009). Millimeter-wave reflective-type phase shifter in CMOS technology. *IEEE Microwave and Wireless components letters*, 19(9):560–562.

BLAKE, G., DRESLINSKI, R. G., MUDGE, T. et FLAUTNER, K. (2010). Evolution of thread-level parallelism in desktop applications. *In ACM SIGARCH Computer Architecture News*, volume 38, pages 302–313. ACM.

BLOCH, E. (1959). The engineering design of the Stretch computer. *In Papers presented at the December 1-3, 1959, eastern joint IRE-AIEE-ACM computer conference*, pages 48–58. ACM.

BOGAERTS, W., LIU, L. et ROELKENS, G. (2013). Technologies and building blocks for on-chip optical interconnects. *In Integrated Optical Interconnect Architectures for Embedded Systems*, pages 27–78. Springer.

BORKAR, S. et CHIEN, A. A. (2011). The future of microprocessors. *Communications of the ACM*, 54(5):67–77.

BOWHILL, B., STACKHOUSE, B., NASSIF, N., YANG, Z., RAGHAVAN, A., MENDOZA, O., MORGANTI, C., HOUGHTON, C., KRUEGER, D., FRANZA, O. *et al.* (2016). The Xeon® Processor E5-2600 v3 : a 22 nm 18-Core Product Family. *IEEE Journal of Solid-State Circuits*, 51(1):92–104.

BRIÈRE, A., DENOULET, J., PINNA, A., GRANADO, B., PÊCHEUX, F., UNLU, E., LOUËT, Y. et MOY, C. (2015). A Dynamically Reconfigurable RF NoC for Many-Core. *In Proceedings of the 25th edition on Great Lakes Symposium on VLSI*, GLSVLSI, pages 139–144. ACM.

CHANG, M.-C., ROYCHOWDHURY, V. P., ZHANG, L., SHIN, H. et QIAN, Y. (2001). RF/wireless interconnect for inter-and intra-chip communications. *Proceedings of the IEEE*, 89(4):456–466.

CHANG, M. F. (2005). CDMA/FDMA-interconnects for future ULSI communications. *In ICCAD-2005. IEEE/ACM International Conference on Computer-Aided Design, 2005.*, pages 975–978. IEEE.

CHANG, M. F., CONG, J., KAPLAN, A., NAIK, M., REINMAN, G., SOCHER, E. et TAM, S.-W. (2008). CMP network-on-chip overlaid with multi-band RF-interconnect. *In 2008 IEEE 14th International Symposium on High Performance Computer Architecture*, pages 191–202. IEEE.

CHANG, M. F., ROYCHOWDHURY, V., ZHANG, L., ZHOU, S., WANG, Z., WU, Y., MA, P., LIN, C. et KANG, Z. (2000). Multi-I/O and reconfigurable RF/wireless interconnect based on near field capacitive coupling and multiple access techniques. *In Interconnect Technology Conference, Proceedings of the IEEE International*, pages 21–22. IEEE.

CHIU, G.-M. (2000). The odd-even turn model for adaptive routing. *IEEE Transactions on parallel and distributed systems*, 11(7):729–738.

COPPOLA, M., LOCATELLI, R., MARUCCIA, G., PIERALISI, L. et SCANDURRA, A. (2004). Spidergon : a novel on-chip communication network. *In System-on-Chip, 2004. Proceedings. 2004 International Symposium on*, page 15. IEEE.

DAI, P., CHEN, J., ZHAO, Y. et LAI, Y.-H. (2015). A study of a wire–wireless hybrid NoC architecture with an energy-proportional multicast scheme for energy efficiency. *Computers & Electrical Engineering*, 45:402–416.

DALLY, W. J. et TOWLES, B. (2001). Route packets, not wires : on-chip interconnection networks. *In Design Automation Conference, 2001. Proceedings*, pages 684–689. IEEE.

DAYA, B. K., CHEN, C.-H. O., SUBRAMANIAN, S., KWON, W.-C., PARK, S., KRISHNA, T., HOLT, J., CHANDRAKASAN, A. P. et PEH, L.-S. (2014). SCORPIO : A 36-core research chip demonstrating snoopy coherence on a scalable mesh NoC with in-network ordering. *In Computer Architecture (ISCA), 2014 ACM/IEEE 41st International Symposium on*, pages 25–36. IEEE.

DE DINECHIN, B. D., AYRIGNAC, R., BEAUCAMPS, P.-E., COUVERT, P., GANNE, B., de MASSAS, P. G., JACQUET, F., JONES, S., CHAISEMARTIN, N. M., RISS, F. *et al.* (2013). A clustered manycore processor architecture for embedded and accelerated applications. *In High Performance Extreme Computing Conference (HPEC), 2013 IEEE*, pages 1–6. IEEE.

DEB, S., CHANG, K., COSIC, M., GANGULY, A., PANDE, P., HEO, D. et BELZER, B. (2012). CMOS compatible many-core NoC architectures with multi-channel millimeter-wave wireless links. *In Proceedings of the great lakes symposium on VLSI*, pages 165–170. ACM.

DONG, X., WU, X., SUN, G., XIE, Y., LI, H. et CHEN, Y. (2008). Circuit and microarchitecture evaluation of 3D stacking magnetic RAM (MRAM) as a universal memory replacement. *In Design Automation Conference. DAC 2008. 45th ACM/IEEE*, pages 554–559. IEEE.

DOUGLAS, J. (2005). Intel 8xx series and Paxville Xeon-MP Microprocessors. *In Hot Chips XVII Symposium (HCS), 2005 IEEE*, pages 1–26. IEEE.

DOWECK, J. (2006). Inside Intel Core microarchitecture. *In Hot Chips 18 Symposium (HCS), 2006 IEEE*, pages 1–35. IEEE.

DRILLET, F., HAMIEH, M., ZERIOUL, L., BRIÈRE, A., UNLU, E., ARIAUDO, M., LOUET, Y., BOURDEL, E., DENOULET, J., PINNA, A. *et al.* (2014). Flexible Radio Interface for NoC RF-Interconnect. *In Digital System Design, 17th Euromicro Conference on*, DSD, pages 36–41. IEEE.

ERCOLI, M., DRAGOMIRESCU, D. et PLANA, R. (2012). An extremely miniaturized ultra wide band 10–67 ghz power splitter in 65 nm cmos technology. *In Microwave Symposium Digest (MTT), 2012 IEEE MTT-S International*, pages 1–3. IEEE.

ESMAEILZADEH, H., BLEM, E., AMANT, R. S., SANKARALINGAM, K. et BURGER, D. (2013). Power challenges may end the multicore era. *Communications of the ACM*, 56(2):93–102.

FEEHRER, J., JAIRATH, S., LOEWENSTEIN, P., SIVARAMAKRISHNAN, R., SMENTEK, D., TURULLOLS, S. et VAHIDSAFA, A. (2013). The Oracle Sparc T5 16-core processor scales to eight sockets. *IEEE Micro*, 33(2):48–57.

FERNANDES, R., BRAHM, L., WEBBER, T., CATALDO, R., POEHLS, L. B. et MARCON, C. (2015). OcNoC : Efficient One-Cycle Router Implementation for 3D Mesh Network-on-Chip. *In VLSI Design (VLSID), 2015 28th International Conference on*, pages 105–110. IEEE.

FERRISS, M., RYLYAKOV, A., TIERNO, J. A., AINSPAN, H. et FRIEDMAN, D. J. (2014). A 28 GHz Hybrid PLL in 32 nm SOI CMOS. *IEEE Journal of Solid-State Circuits*, 49(4):1027–1035.

FLUHR, E. J., BAUMGARTNER, S., BOERSTLER, D., BULZACCHELLI, J. F., DIEMOZ, T., DREPS, D., ENGLISH, G., FRIEDRICH, J., GATTIKER, A., GLOEKLER, T. *et al.* (2015). The 12-Core POWER8^{TM} Processor With 7.6 Tb/s IO Bandwidth, Integrated Voltage Regulation, and Resonant Clocking. *IEEE Journal of Solid-State Circuits*, 50(1):10–23.

GANGULY, A., CHANG, K., DEB, S., PANDE, P., BELZER, B. et TEUSCHER, C. (2011). Scalable hybrid wireless network-on-chip architectures for multicore systems. *Computers, IEEE Transactions on*, 60(10):1485–1502.

GONG, C.-S., HONG, C.-T., YAO, K.-W. et SHIUE, M.-T. (2008). A low-power area-efficient SRAM with enhanced read stability in 0.18-μm CMOS. *In Circuits and Systems. APCCAS 2008. IEEE Asia Pacific Conference on*, pages 729–732. IEEE.

GOODMAN, J. W., LEONBERGER, F. J., KUNG, S.-Y. et ATHALE, R. A. (1984). Optical interconnections for VLSI systems. *Proceedings of the IEEE*, 72(7):850–866.

GREINER, A. (2009). Tsar : a scalable, shared memory, many-cores architecture with global cache coherence. *In 9th International Forum on Embedded MPSoC and Multicore (MPSoC'09)*.

GUERRIER, P. et GREINER, A. (2000). A generic architecture for on-chip packet-switched interconnections. *In Proceedings of the conference on Design, automation and test in Europe*, pages 250–256. ACM.

HAMIEH, M., ARIAUDO, M., QUINTANEL, S. et LOUËT, Y. (2014). Sizing of the Physical Layer of a RF Intra-Chip Communications. *In Electronics, Circuits and Systems (ICECS), 2014 21th IEEE International Conference on*. IEEE.

HAMMARLUND, P., MARTINEZ, A. J., BAJWA, A. A., HILL, D. L., HALLNOR, E., JIANG, H., DIXON, M., DERR, M., HUNSAKER, M., KUMAR, R. *et al.* (2014). Haswell : The fourth-generation Intel Core processor. *IEEE Micro*, 34(2):6–20.

HOWARD, J., DIGHE, S., HOSKOTE, Y., VANGAL, S., FINAN, D., RUHL, G., JENKINS, D., WILSON, H., BORKAR, N., SCHROM, G. *et al.* (2010). A 48-core IA-32 message-passing processor with DVFS in 45nm CMOS. *In Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2010 IEEE International*, pages 108–109. IEEE.

HU, J., XU, J., HUANG, M. et WU, H. (2013). A 25-Gbps 8-ps/mm transmission line based interconnect for on-chip communications in multi-core chips. *In Microwave Symposium Digest (IMS), 2013 IEEE MTT-S International*, pages 1–4. IEEE.

HU, Q., LIU, P., HUANG, M. C. et XIE, X.-H. (2015). Exploiting Transmission Lines on Heterogeneous Networks-on-Chip to Improve the Adaptivity and Efficiency of Cache Coherence. *In Proceedings of the 9th International Symposium on Networks-on-Chip*, NOCS '15. ACM.

ITO, H., KIMURA, M., MIYASHITA, K., ISHII, T., OKADA, K. et MASU, K. (2008). A bidirectional and multi-drop transmission-line interconnect for multipoint-to-multipoint on-chip communications. *Solid-State Circuits, IEEE Journal of*, 43(4): 1020–1029.

ITRS (2013). International Technology Roadmap for Semiconductors, 2013 version. http://www.itrs.net/Links/2013ITRS/Home2013.htm.

JAGANNATHAN, A., YANG, H. H., KONIGSFELD, K., MILLIRON, D., MOHAN, M., ROMESIS, M., REINMAN, G. et CONG, J. (2005). Microarchitecture evaluation with floorplanning and interconnect pipelining. *In Proceedings of the 2005 Asia and South Pacific Design Automation Conference*, pages 8–15. ACM.

JERGER, N. E., PEH, L.-S. et LIPASTI, M. (2008). Virtual circuit tree multicasting : A case for on-chip hardware multicast support. *In Computer Architecture, 35th International Symposium on*, ISCA'08, pages 229–240. IEEE.

JONGSUN, K., GYUNGSU, B. et CHANG, M. (2011). A Low-Overhead and Low-Power RF Transceiver for Short-Distance On-and Off-Chip Interconnects. *IEICE transactions on electronics*, 94(5):854–857.

KARKAR, A., MAK, T., DAHIR, N., AL-DUJAILY, R., TONG, K. F. et YAKOVLEV, A. (2016). Network-on-Chip Multicast Architectures Using Hybrid Wire and Surface-Wave Interconnects. *IEEE Transactions on Emerging Topics in Computing*, (99):1–12.

KIM, J., BALFOUR, J. et DALLY, W. (2007). Flattened butterfly topology for on-chip networks. *In Microarchitecture, 2007. MICRO 2007. 40th Annual IEEE/ACM International Symposium on*, pages 172–182. IEEE.

KIRMAN, N. et MARTÍNEZ, J. F. (2010). A Power-efficient All-optical On-chip Interconnect Using Wavelength-based Oblivious Routing. *In Proceedings of the Fifteenth Edition of the International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS XV, pages 15–28. ACM.

KONGETIRA, P., AINGARAN, K. et OLUKOTUN, K. (2005). Niagara : a 32-way multithreaded sparc processor. *IEEE Micro*, 25(2):21–29.

KREWELL, K. (2003). UltraSPARC IV mirrors predecessor. *Microprocessor Report*.

KULL, L., PLIVA, J., TOIFL, T., SCHMATZ, M., FRANCESE, P. A., MENOLFI, C., BRAENDLI, M., KOSSEL, M., MORF, T., ANDERSEN, T. M. *et al.* (2014). A 110 mW 6 bit 36 GS/s interleaved SAR ADC for 100 GBE occupying 0.048 mm 2 in 32 nm SOI CMOS. *In Solid-State Circuits Conference (A-SSCC), 2014 IEEE Asian*, pages 89–92. IEEE.

KURD, N., CHOWDHURY, M., BURTON, E., THOMAS, T. P., MOZAK, C., BOSWELL, B., LAL, M., DEVAL, A., DOUGLAS, J., ELASSAL, M. *et al.* (2014). Haswell : A family of IA 22nm processors. *In Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2014 IEEE International*, pages 112–113. IEEE.

KURIAN, G., MILLER, J., PSOTA, J., EASTEP, J., LIU, J., MICHEL, J., KIMERLING, L. et AGARWAL, A. (2010). ATAC : A 1000-core cache-coherent processor with on-chip optical network. *In Proceedings of the 19th international conference on Parallel architectures and compilation techniques*, pages 477–488. ACM.

LE BEUX, S., LI, H., O'CONNOR, I., CHESHMI, K., LIU, X., TRAJKOVIC, J. et NICOLESCU, G. (2014). Chameleon : Channel efficient optical network-on-chip. *In Design, Automation and Test in Europe Conference and Exhibition (DATE), 2014*, pages 1–6. IEEE.

LEE, J., ZHU, M., CHOI, K., AHN, J. H. et SHARMA, R. (2011). 3D network-on-chip with wireless links through inductive coupling. *In SoC Design Conference (ISOCC), 2011 International*, pages 353–356. IEEE.

LEE, S., TAM, S., PEFKIANAKIS, I., LU, S., CHANG, M., GUO, C., REINMAN, G., PENG, C., NAIK, M., ZHANG, L. *et al.* (2009). A scalable micro wireless interconnect structure for CMPs. *In Proceedings of the 15th annual international conference on Mobile computing and networking*, pages 217–228. ACM.

LEE, Y., WATERMAN, A., AVIZIENIS, R., COOK, H., SUN, C., STOJANOVIĆ, V. et ASANOVIĆ, K. (2014). A 45nm 1.3 ghz 16.7 double-precision gflops/w risc-v processor with vector accelerators. *In European Solid State Circuits Conference (ESSCIRC), ESSCIRC 2014-40th*, pages 199–202. IEEE.

LI, F., NICOPOULOS, C., RICHARDSON, T., XIE, Y., NARAYANAN, V. et KANDEMIR, M. (2006). Design and management of 3D chip multiprocessors using network-in-memory. *ACM SIGARCH Computer Architecture News*, 34(2):130–141.

MANEVICH, R., CIDON, I., KOLODNY, A. *et al.* (2009). Best of both worlds : A bus enhanced NoC (BENoC). *In Networks-on-Chip, 3rd ACM/IEEE International Symposium on*, NoCS 2009, pages 173–182. IEEE.

MANSOOR, N., SHAMIM, M. S. et GANGULY, A. (2016). A Demand-Aware Predictive Dynamic Bandwidth Allocation Mechanism for Wireless Network-on-Chip. *In Proceedings of the 18th System Level Interconnect Prediction Workshop*, SLIP '16, pages 8 :1–8 :8. ACM.

MATSUTANI, H., BOGDAN, P., MARCULESCU, R., TAKE, Y., SASAKI, D., ZHANG, H., KOIBUCHI, M., KURODA, T. et AMANO, H. (2013). A case for wireless 3d nocs for cmps. *In Design Automation Conference, 18th Asia and South Pacific*, ASP-DAC, pages 23–28. IEEE.

MELPIGNANO, D., BENINI, L., FLAMAND, E., JEGO, B., LEPLEY, T., HAUGOU, G., CLERMIDY, F. et DUTOIT, D. (2012). Platform 2012, a many-core computing accelerator for embedded SoCs : performance evaluation of visual analytics applications. *In Proceedings of the 49th Annual Design Automation Conference*, pages 1137–1142. ACM.

MILDER, P., FRANCHETTI, F., HOE, J. C. et PÜSCHEL, M. (2012). Computer generation of hardware for linear digital signal processing transforms. *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, 17(2):15.

MIRO-PANADES, I., CLERMIDY, F., VIVET, P. et GREINER, A. (2008). Physical Implementation of the DSPIN Network-on-Chip in the FAUST Architecture. *In Proceedings of the Second ACM/IEEE International Symposium on Networks-on-Chip*, pages 139–148. IEEE Computer Society.

MIRO-PANADES, I., GREINER, A. et SHEIBANYRAD, A. (2006). A Low Cost Network-on-Chip with Guaranteed Service Well Suited to the GALS Approach. *In 2006 1st International Conference on Nano-Networks and Workshops*, pages 1–5. IEEE.

MISHRA, A. K., MUTLU, O. et DAS, C. R. (2013). A heterogeneous multiple network-on-chip design : an application-aware approach. *In Proceedings of the 50th Annual Design Automation Conference*, page 36. ACM.

MISHRA, A. K., VIJAYKRISHNAN, N. et DAS, C. R. (2011). A case for heterogeneous on-chip interconnects for CMPs. *In ACM SIGARCH Computer Architecture News*, volume 39, pages 389–400. ACM.

MOORE, G. E. (1965). Cramming More Components Onto Integrated Circuits. *Electronics*, 38(8):114–117.

MOORE, G. E. (1975). Progress in Digital Integrated Electronics. *In International Electron Devices Meeting. Technical Digest*, pages 11–13. IEEE.

MORE, A. et TASKIN, B. (2010). Simulation based study of on-chip antennas for a reconfigurable hybrid 3D wireless NoC. *In SOC Conference (SOCC), 2010 IEEE International*, pages 447–452. IEEE.

MORRIS, R., JOLLEY, E. et KODI, A. K. (2014). Extending the performance and energy-efficiency of shared memory multicores with nanophotonic technology. *IEEE Transactions on Parallel and Distributed Systems*, 25(1):83–92.

MÜLLER, M. S., BARON, J., BRANTLEY, W. C., FENG, H., HACKENBERG, D., HENSCHEL, R., JOST, G., MOLKA, D., PARROTT, C., ROBICHAUX, J. *et al.* (2012). SPEC OMP2012 - an application benchmark suite for parallel systems using OpenMP. *In International Workshop on OpenMP*, pages 223–236. Springer.

NALAMALPU, A., KURD, N., DEVAL, A., MOZAK, C., DOUGLAS, J., KHANNA, A., PAILLET, F., SCHROM, G. et PHELPS, B. (2015). Broadwell : A family of IA 14nm processors. *In VLSI Circuits (VLSI Circuits), 2015 Symposium on*, pages C314–C315. IEEE.

NAZEMI, A., HU, K., CATLI, B., CUI, D., SINGH, U., HE, T., HUANG, Z., ZHANG, B., MOMTAZ, A. et CAO, J. (2015). A 36gb/s pam4 transmitter using an 8b 18gs/s dac in 28nm cmos. *In 2015 IEEE International Solid-State Circuits Conference-(ISSCC) Digest of Technical Papers*, pages 1–3. IEEE.

NVIDIA (2016). NVIDIA Tesla P100 White Paper. `http://www.nvidia.com/object/pascal-architecture-whitepaper.html`.

O'CONNOR, I. (2004). Optical solutions for system-level interconnect. *In Proceedings of the 2004 international workshop on System level interconnect prediction*, pages 79–88. ACM.

O'CONNOR, I. et GAFFIOT, F. (2004). On-chip optical interconnect for low-power. *In Ultra Low-Power Electronics and Design*, pages 21–39. Springer.

OLOFSSON, A. (2016). Epiphany-v : A 1024 processor 64-bit risc system-on-chip. *arXiv preprint arXiv :1610.01832*.

OLUKOTUN, K., NAYFEH, B. A., HAMMOND, L., WILSON, K. et CHANG, K. (1996). The case for a single-chip multiprocessor. 31(9):2–11.

PAN, Y., KUMAR, P., KIM, J., MEMIK, G., ZHANG, Y. et CHOUDHARY, A. (2009). Firefly : illuminating future network-on-chip with nanophotonics. *ACM SIGARCH Computer Architecture News*, 37(3):429–440.

PANDA, P. R. (2001). SystemC - a modeling platform supporting multiple design abstractions. *In System Synthesis, 2001. Proceedings. The 14th International Symposium on*, pages 75–80. IEEE.

PASSAS, G., KATEVENIS, M. et PNEVMATIKATOS, D. (2012). Crossbar NoCs are scalable beyond 100 nodes. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 31(4):573–585.

PAVLIDIS, V. et FRIEDMAN, E. (2007). 3D topologies for networks-on-chip. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 15(10):1081–1090.

PEKKARINEN, E., LEHTONEN, L., SALMINEN, E. et HAMALAINEN, T. (2011). A set of traffic models for Network-on-Chip benchmarking. *In System on Chip (SoC), 2011 International Symposium on*, pages 78–81. IEEE.

RAPONI, P. G., PETRINI, F., WALKUP, R. et CHECCONI, F. (2011). Characterization of the communication patterns of scientific applications on Blue Gene/P. *In Parallel and Distributed Processing Workshops and Phd Forum (IPDPSW), 2011 IEEE International Symposium on*, pages 1017–1024. IEEE.

RICH, D. I. (2003). The evolution of SystemVerilog. *IEEE Design & Test*, 20(04):82–84.

RUSU, S., TAM, S., MULJONO, H., AYERS, D., CHANG, J., CHERKAUER, B., STINSON, J., BENOIT, J., VARADA, R., LEUNG, J. *et al.* (2007). A 65-nm dual-core multithreaded Xeon® processor with 16-MB L3 cache. *IEEE Journal of Solid-State Circuits*, 42(1):17–25.

SALIHUNDAM, P., JAIN, S., JACOB, T., KUMAR, S., ERRAGUNTLA, V., HOSKOTE, Y., VANGAL, S., RUHL, G. et BORKAR, N. (2011). A 2 Tb/s 6×4 Mesh Network for a Single-Chip Cloud Computer with DVFS in 45 nm CMOS. *Solid-State Circuits, IEEE Journal of*, 46(4):757–766.

SANSEN, W. (2015). 1.3 Analog CMOS from 5 micrometer to 5 nanometer. *In 2015 IEEE International Solid-State Circuits Conference-(ISSCC) Digest of Technical Papers*, pages 1–6. IEEE.

SHAH, M., GOLLA, R., GROHOSKI, G., JORDAN, P., BARREH, J., BROOKS, J., GREENBERG, M., LEVINSKY, G., LUTTRELL, M., OLSON, C. *et al.* (2012). Sparc T4 : A dynamically threaded server-on-a-chip. *IEEE Micro*, 2(32):8–19.

SHAHDAD, M., LIPSETT, R., MARSCHNER, E., SHEEHAN, K. et COHEN, H. (1985). VHSIC Hardware Description Language. *Computer*, 18(2):94–103.

SOCLIB CONSORTIUM (2003). The SoCLib project : An integrated system-on-chip modelling and simulation platform. Rapport technique, CNRS.

STARKE, W., STUECHELI, J., DALY, D., DODSON, J., AUERNHAMMER, F., SAGMEISTER, P., GUTHRIE, G., MARINO, C., SIEGEL, M. et BLANER, B. (2015). The cache and memory subsystems of the IBM POWER8 processor. *IBM Journal of Research and Development*, 59(1):3–1.

TAYLOR, M. B., KIM, J., MILLER, J., WENTZLAFF, D., GHODRAT, F., GREENWALD, B., HOFFMAN, H., JOHNSON, P., LEE, J.-W., LEE, W. *et al.* (2002). The raw microprocessor : A computational fabric for software circuits and general-purpose programs. *IEEE micro*, 22(2):25–35.

TENDLER, J. M., DODSON, J. S., FIELDS, J., LE, H. et SINHAROY, B. (2002). POWER4 system microarchitecture. *IBM Journal of Research and Development*, 46(1):5–25.

THOMAS, D. E. et MOORBY, P. R. (1991). *The VERILOG Hardware Description Language*. Kluwer Academic Publishers.

THONNART, Y., VIVET, P. et CLERMIDY, F. (2010). A fully-asynchronous low-power framework for GALS NoC integration. *In Design, Automation & Test in Europe Conference & Exhibition (DATE), 2010*, pages 33–38. IEEE.

TOMASULO, R. M. (1967). An efficient algorithm for exploiting multiple arithmetic units. *IBM Journal of research and Development*, 11(1):25–33.

VANTREASE, D., BINKERT, N., SCHREIBER, R. et LIPASTI, M. H. (2009). Light speed arbitration and flow control for nanophotonic interconnects. *In Microarchitecture, 2009. MICRO-42. 42nd Annual IEEE/ACM International Symposium on*, pages 304–315. IEEE.

VANTREASE, D., SCHREIBER, R., MONCHIERO, M., MCLAREN, M., JOUPPI, N. P., FIORENTINO, M., DAVIS, A., BINKERT, N., BEAUSOLEIL, R. G. et AHN, J. H. (2008). Corona : System implications of emerging nanophotonic technology. *ACM SIGARCH Computer Architecture News*, 36(3):153–164.

von NEUMANN, J. (1945). First Draft of a Report on the EDVAC.

WANG, Y., AN, H., YAN, J., LI, Q., HAN, W., WANG, L. et LIU, G. (2009). Investigation of factors impacting thread-level parallelism from desktop, multimedia and HPC applications. *In Frontier of Computer Science and Technology, 2009. FCST'09. Fourth International Conference on*, pages 27–32. IEEE.

WENTZLAFF, D., GRIFFIN, P., HOFFMANN, H., BAO, L., EDWARDS, B., RAMEY, C., MATTINA, M., MIAO, C.-C., BROWN, J. F. et AGARWAL, A. (2007). On-chip interconnection architecture of the tile processor. *IEEE Micro*, 27(5):15–31.

WOO, S. C., OHARA, M., TORRIE, E., SINGH, J. P. et GUPTA, A. (1995). The SPLASH-2 programs : Characterization and methodological considerations. *In ACM SIGARCH Computer Architecture News*, volume 23, pages 24–36. ACM.

WU, H., NAN, L., TAM, S., HSIEH, H., JOU, C., REINMAN, G., CONG, J. et CHANG, M. (2012). A 60GHz on-chip RF-Interconnect with $\lambda/4$ coupler for 5Gbps bidirectional communication and multi-drop arbitration. *In Custom Integrated Circuits Conference (CICC), 2012 IEEE*, pages 1–4. IEEE.

XIAO, C., CHANG, F., CONG, J., GILL, M., HUANG, Z., LIU, C., REINMAN, G. et WU, H. (2013). Stream arbitration : Towards efficient bandwidth utilization for emerging on-chip interconnects. *ACM Transactions on Architecture and Code Optimization (TACO)*, 9(4):639–648.

XIE, Y., LOH, G. H., BLACK, B. et BERNSTEIN, K. (2006). Design space exploration for 3D architectures. *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, 2(2):65–103.

YE, Y., XU, J., HUANG, B., WU, X., ZHANG, W., WANG, X., NIKDAST, M., WANG, Z., LIU, W. et WANG, Z. (2013). 3-D mesh-based optical network-on-chip for multiprocessor system-on-chip. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 32(4):584–596.

Glossaire

CAN
Convertisseur Analogique Numérique – composant électronique permettant de transformer un signal analogique en signal numérique. 103, 109

CMP
Chip Multiprocessor – puce intégrant plusieurs unités de calcul. iii, v, xi, xii, 2, 3, 6–8, 11–13, 15–18, 20–25, 28–31, 33, 34, 36–38, 40–47, 51, 54, 55, 57, 92, 94–103, 105, 110–112, 115, 116, 118, 119, 123, 124, 128, 129

CNA
Convertisseur Numérique Analogique – composant électronique permettant de transformer un signal numérique en signal analogique. 103, 109

DSPIN
Distributed Scalable Programmable Interconnection Network – réseau sur puce de type grille. 81, 82, 95, 115, 119–121

FLIT
FLow control unIT – unité atomique circulant sur le réseau. xi, xii, 34, 60, 62–65, 68–70, 80, 84–86, 88–91, 96–100, 116–118

GALS
Globally Asynchronous Locally Synchronous – système dans lequel des ilôts sont localement synchrones mais ne le sont pas entre eux. 22

HDL
Hardware Description Language – langage permettant de décrire un circuit électronique. 78, 79

HPC
High Performance Computing – développement et utilisation d'ordinateurs permettant un grand nombre de calcul par rapport aux ordinateurs personnels contemporains. 11, 12

ILP
Instruction Level Parallelism – exécution simultanée de plusieurs instructions. 6, 10, 15

IP core
Intellectual Property core – bloc de base permettant de composer un circuit ASIC ou FPGA. 80, 81

MAE
Machine À États – automate décrivant les différents états d'un système et les transitions entre ces états. 83–85, 90, 91

MRAM
Magnetic Random Access Memory – type de mémoire basée sur un support magnétique de l'information dans le but de mêler les avantages des mémoires flash et des mémoires volatiles. 35

NoC
Network on Chip – réseau de communication intégré sur puce. 2–4, 7, 8, 16, 20–26, 28–52, 54–57, 59, 60, 62, 63, 65, 66, 68–71, 74, 76, 80–84, 86, 89, 92, 94–103, 105, 110, 111, 115–125, 128–131

NUMA
Non Uniform Memory Access – Architecture où le temps d'accès à une donnée varie en fonction du banc mémoire où elle se trouve. 55

OCP
Open Core Protocol – interface de communication pour les systèmes sur puce. 80

OFDM
Orthogonal Frequency Division Multiplexing – Méthode de multiplexage fréquentielle d'un signal. 55–57

OFDMA
Orthogonal Frequency Division Multiple Access – Technique basée sur l'OFDM permettant de partagée une bande de fréquence entre plusieurs sources. 56

PC
Program Counter – Registre contenant l'adresse de l'instruction exécutée. 11

SoC
System on Chip – système intégré sur puce. 22, 23, 39, 130, 131

TLP
Thread Level Parallelism – exécution simultanée de plusieurs threads ou tâches. 6, 11, 12, 15

TSAR
Tera-Scale ARchitecture – Architecture many-cœur scalable avec cohérence de cache. 112

TSV
Through Silicon Via – connexion traversante entre deux couches d'une puce 3D. 34

UAL
Unité Arithmétique et Logique – composant interne d'un processeur chargé d'effectuer les opérations arithmétiques et logiques. 8

UC

Unité de Contrôle – composant interne d'un processeur chargé de gérer le flot d'exécution d'un processeur. 8

UCT

Unité Centrale de Traitement – terme généralement utiliser pour parler du processeur en lui même. 8, 9, 11

VCI

Virtual Component Interface – interface de communication pour les systèmes sur puce. 80, 81, 83–85, 89, 90