Méthodologie de modélisation et d'exploration d'architecture de réseaux sur puce appliquée aux télécommunications

Julien Delorme

N° d'ordre : D 07 - 02

Thèse

présentée devant

l'**Institut national des sciences appliquées de Rennes**

pour obtenir le titre de

Docteur

spécialité : *Electronique*

Méthodologie de modélisation et d'exploration d'architecture de réseaux sur puce appliquée aux télécommunications

par

DELORME Julien

Soutenue le 21 Février 2007 devant la commission d'Examen

Composition du jury

Rapporteurs

M. Jean-Philippe DIGUET	Chargé de recherche CNRS, UBS, LESTER, Lorient
M. El-Bay BOURENNANE	Professeur des universités, UB, Le2i, Dijon

Examinateurs

Mme. Daniela DRAGOMIRESCU	Maître de conférences, LAAS-CNRS, Toulouse
M. Jean-Luc PHILIPPE	Professeur des universités, UBS, LESTER, Lorient
M. Jérôme MARTIN	Ingénieur chercheur, CEA LETI, Grenoble
M. Dominique HOUZET	Professeur des universités, ENSERG, LIS, Grenoble

Institut d'électronique et de télécommunications de Rennes
Institut national des sciences appliquées de Rennes

à Marie-Pierre,
à ma famille

Remerciements

En premier lieu, je tiens à remercier Dominique HOUZET, désormais professeur à l'INPG de Grenoble, de m'avoir accueilli au sein de la composante INSA de l'IETR et d'avoir accepter de diriger mes travaux de recherche. Je le remercie pour ses conseils, son encadrement et pour la confiance qu'il m'a accordé durant ces trois années de thèse pour mener à bien ces travaux de recherche.

Je tiens également à remercier l'ensemble des membres de mon jury d'avoir accepter de juger et d'évaluer ces travaux de thèse. Je remercie sincèrement Jean-Philippe DIGUET, chargé de recherche CNRS au LESTER à Lorient et El-bay BOURENNANE, professeur des universités à l'université de Bourgogne de Dijon et président du jury pour l'attention qu'ils ont accordés à la lecture de ce manuscrit ainsi que pour leur participation au jury en tant que rapporteurs. Je remercie également Daniela DRAGOMIRESCU, maître de conférences à l'INSA de Toulouse, Jean-Luc PHILIPPE, professeur des universités à l'université de Bretagne sud (UBS) de Lorient ainsi que Jérôme MARTIN, ingénieur chercheur au CEA LETI de Grenoble d'avoir pris de leur temps et d'avoir participé au jury en tant qu'examinateurs.

En outre, je tiens à remercier Saurinkumar Patel pour le sérieux de son travail effectué durant son stage de fin d'études d'ingénieur dont les résultats font partie intégrante de cette thèse. Son encadrement s'est révélé être une expérience particulièrement enrichissante.

Pour leur bonne humeur, leur joie de vivre au quotidien et leurs conseils précieux, j'adresse un grand merci à l'ensemble des permanents, doctorants et stagiaires que j'ai côtoyés durant ces trois années de thèse, anciens comme nouveaux qui ont contribués à la bonne ambiance qui règne quotidiennement au laboratoire. Ces échanges entre anciens et nouveaux doctorants ont été très enrichissants tant personnellement que professionnellement. Ainsi, j'aimerai les remercier et plus particulièrement Arnaud le corse, David the low power man, Wilfried le tombeur de chaise, Florent la mauvaise langue, Christophe le recycleur de thé, Sylvie, Yvan, Marie-Anne, François, Romain, Emeric et tous ceux dont le nom n'est pas mentionné.

Je tiens également à remercier sincèrement mes amis et ma famille, et plus particulièrement mes parents pour leur soutient constant durant toutes ces années d'études.

Pour finir, je ne pourrai clore ces remerciements sans remercier du fond du cœur ma moitié pour son soutien, sa patience et son écoute pendant ces trois années de thèse. Cette thèse est le fruit d'un effort vécu à deux dans la complexité de l'adéquation entre réussite personnelle et professionnelle. Je ne la remercierai jamais assez pour m'avoir encouragé et soutenu en permanence notamment lors du sprint final de la rédaction de ce manuscrit qui a connu des moments difficiles sur la fin ...

Table des matières

Introduction

Depuis les années 70, les techniques d'intégration de transistors dans les systèmes électroniques ne cessent de s'améliorer. Ainsi, les systèmes à base de puces électroniques font de plus en plus partie de notre quotidien. Ceux-ci intègrent maintenant plusieurs millions de transistors comme en témoigne l'évolution de leur nombre dans la gamme des microprocesseurs de la société INTEL (cf. figure 1).

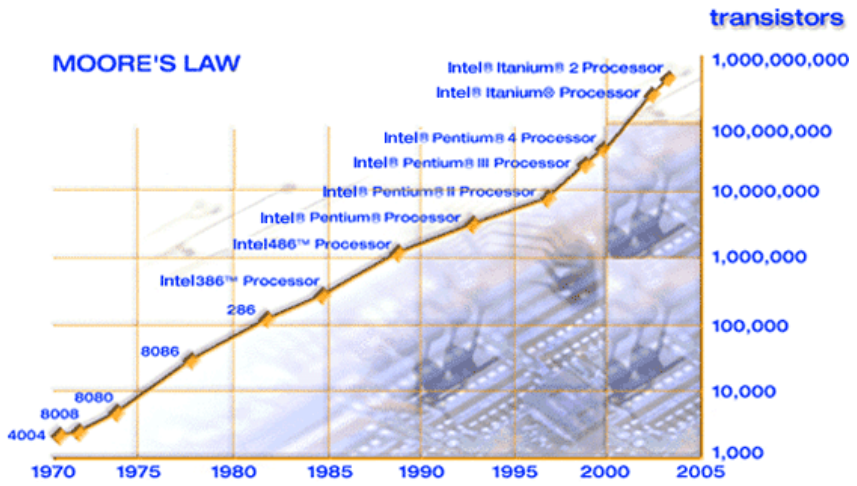

Fig. 1 – Courbe de densité d'intégration des transistors dans les processeurs Intel

Cette croissance d'intégration a donc permis la réalisation de circuits numériques aux fonctionnalités plus nombreuses et plus complexes. Aujourd'hui, les concepteurs peuvent implanter un circuit numérique complet sur une seule et même puce, c'est ce que l'on appelle le système sur puce, aussi appelé SoC (System on Chip). Ces SoC permettent d'intégrer différents éléments comme des processeurs, des mémoires, des blocs de propriété intellectuelle (IP), des blocs d'entrée/sortie ou des médias de communications.

De plus, l'augmentation des performances et la miniaturisation des circuits ont amené les concepteurs à la réalisation de systèmes électroniques de plus en plus flexibles et complexes. Cette demande réside principalement dans la nécessité d'intégrer de plus en plus de standards dans une même puce avec des applications exigeant de plus en plus de débit (téléphone mobile, appareil photo, PDA, ...). Par conséquent, le nombre de fonctionnalités présentes au sein d'un même SoC conduisent à la mise en communication d'un nombre de blocs fonctionnels de nature différente de plus en plus important. Ainsi, les contraintes

de l'application du SoC se répercutent directement sur les liens de communication entre blocs, entraînant des goulets d'étranglement au niveau des interconnexions.

L'augmentation de la complexité des systèmes embarqués, notamment dans le domaine des télécommunications, la réduction des temps de mise sur le marché (Time-to-Market), les besoins de flexibilité et de bande passante sont des facteurs nécessitant l'adoption d'une méthodologie de conception adéquate. Deux approches permettent de gérer cette complexité :

- l'augmentation du niveau d'abstraction de la description du système
- la réutilisation de blocs déjà conçus (IP).

Ces deux approches sont en général complémentaires. Une difficulté majeure lors de la conception de tels systèmes est la communication entre les différents modules composant le système. Les techniques couramment employées dans les SoC actuels consistent à adopter une topologie en flot de donnée ou bien une topologie bus. La première n'offre aucune flexibilité au système pour une éventuelle évolution ou bien un ajout de service par exemple. La deuxième, quant à elle, malgré une bonne flexibilité, trouve ses limites en termes de bande passante à mesure que le nombre de blocs fonctionnels utilisés dans le SoC augmente.

C'est pourquoi, les connexions dédiées et les bus seront probablement nettement moins utilisés dans cinq ou dix ans, à cause de leur manque de flexibilité et de leur faible scalabilité. Afin de répondre à ces nouvelles difficultés de conception, de nombreuses recherches ont fait émerger le concept de réseau intégré sur silicium (Network-on-Chip ou NoC) à commutation de paquet ("Packet switching") reprenant l'organisation en couches des réseaux informatiques, appliquée à la conception de circuit intégré. Ce paradigme permet de séparer les ressources de calcul et les communications grâce à l'utilisation de couches indépendantes les unes des autres. Les NoC offriront donc une solution performante et économique basée sur une infrastructure de communication configurable préétablie. Le recours à un NoC s'impose pour les circuits de grande dimension, aussi bien du point de vue technologique (asynchronisme entre les parties éloignées d'une même puce), que du point de vue fonctionnel (réutilisation de fonctions IP).

Plusieurs grandes firmes et laboratoires de recherches, acteurs majeurs des développements de SoC, ont d'ores et déjà commencé à explorer des solutions à base de NoC pour remplacer les technologies bus utilisées actuellement dans les SoC. Parmi ces industriels, la jeune société française Arteris a affirmé offrir les premiers outils de réseau sur puce commerciaux.

C'est dans ce contexte précis que les travaux de recherche présentés dans ce manuscrit se sont déroulés afin d'étudier et proposer des solutions sur l'intérêt des NoC pour les futurs SoC et leurs principaux enjeux dans leur phase de mise en œuvre. En effet, ces réseaux sur puce possèdent une structure plus complexe, et de ce fait, leur mise en oeuvre est autrement plus complexe qu'une topologie bus classique. Par conséquent, il est nécessaire de fournir des outils d'aide à la conception de ces NoC afin d'aider le concepteur dans sa démarche de mise en œuvre en réalisant des études, simulations et émulations pour valider l'ensemble de son design avant la conception et la réalisation finale.

Ainsi, les travaux de recherche qui ont été effectués durant cette thèse ont permis de contribuer à la réalisation d'un SoC d'un terminal mobile de 4^{e} génération. Ce SoC était l'un des objectifs majeures du projet Européen 4MORE dans lequel nous étions impliqués. L'objectif de ce projet était de proposer des solutions algorithmiques innovantes permettant de remplir le cahier des charges qui imposait des débits allant de 10 à 100 Mb/s

suivant les conditions d'utilisation (qualité du canal de transmission). Par conséquent, la technique de transmission multi-porteuses combinée à un étalement de spectre (MC-CDMA pour Multiple Carrier Code Division Multiple Access) associée à une technique d'exploitation de la diversité spatiale des antennes (MIMO pour Multiple-Input Multiple-Output) a été retenue.

La réalisation du démonstrateur final permettant de valider ces choix algorithmiques était nécessaire afin de démontrer la faisabilité de ce terminal mobile de 4^{e} génération. Ainsi, le NoC FAUST (Flexible Architecture of Unified System for Telecommunication) proposé par le CEA LETI a été choisi comme solution innovante en terme de média de communication entre les blocs IP de l'application pour la réalisation du SoC final.

Compte-tenu des objectifs imposés par ce projet, et de l'aspect novateur des NoC, il a été nécessaire de réaliser des explorations et simulations de l'application avec les contraintes imposées par l'architecture pour guider les concepteurs dans la réalisation du démonstrateur.

Dans ce contexte, les travaux de recherche réalisés durant cette thèse ont permis de proposer une méthodologie de modélisation et d'exploration d'architecture de réseaux sur puce appliquée aux télécommunications. Ces études se sont basées sur des simulations et explorations au niveau NoC en SystemC TLM.

Le projet 4MORE a également vu la réalisation d'un ASIC intégrant une structure de communication NoC intégrant une partie des blocs IP de l'application. Celui-ci a été réalisé dans une technologie 0.13μm et a été associé à des composants reprogrammables de type FPGA pour réaliser la plate-forme matérielle servant de démonstrateur final du projet.

Dans ce flot de conception mis en œuvre, nous avons également intégré une méthodologie d'Adéquation Algorithme Architecture (AAA) permettant au concepteur de réaliser l'adéquation de son application avec l'architecture NoC FAUST.

Outre cette introduction, ce manuscrit comporte 5 chapitres. Le premier chapitre présente un état de l'art des réseaux sur puce et de leurs architectures associées. Il expose les différentes contraintes qu'imposent ces nouvelles structures et les précautions à prendre en compte lors de leur implantation.

Le chapitre 2 présente la chaîne de traitement de l'application 4G développée au sein du projet 4MORE et utilisée comme application lors de nos expérimentations. Nous présentons dans ce même chapitre un modèle de représentation des blocs de traitement permettant d'intégrer leurs caractéristiques au modèle SystemC.

Le chapitre 3 décrit l'architecture du réseau FAUST utilisé et ses caractéristiques concernant notamment son flot associé avec son modèle SystemC.

Le chapitre 4 décrit la méthodologie de conception développée autour de ce NoC afin d'améliorer les phases d'exploration par simulations mais également en donnant la possibilité de réaliser des explorations sur plate-forme matérielle.

Puis, dans le chapitre 5 sont présentés les différents résultats obtenus durant ces travaux de thèse avec notre méthodologie de conception. Ces résultats d'expérimentation concernent également les contributions apportées dans le cadre du projet Européen 4MORE.

Pour finir, nous présentons nos conclusions sur ces travaux de thèse accompagnées des perspectives à court et moyen terme sur les travaux futurs.

Communications et rapports

Les différents travaux réalisés durant cette thèse nous ont conduit à la rédaction de différentes communications et rapports [1, 2, 3, 4, 5] dont les références sont listées ci-après. Par ailleurs, j'ai contribué personnellement au sein du comité d'organisation des JNRDM 2006 dont quelques précisions sont donnés à la fin de cette introduction.

Communications internationales

- J. DELORME, D. HOUZET, "Fast prototyping PCI platform for real time SoC emulation with SCSI capabilities", ISPASS, USA Texas Austin, mars 2005 (11 pages). Article refusé et en cours de resoumission.
- J. DELORME, D. HOUZET, "A complete 4G radiocommunication application mapping onto a 2D mesh NoC architecture", NEWCAS, CANADA Ottawa, juin 2006 (4 pages).
- J. DELORME, D. HOUZET, "An automatic design flow for mapping application onto a 2D mesh NoC architecture", NoCsymposium, USA New Jersey Princeton, mai 2007 (6 pages).

Communications nationales

- J. DELORME, D. HOUZET, "Technique de mapping pour les réseaux sur puce de structure 2D", JNRDM, FRANCE Rennes, mai 2006 (4 pages).

Communications invités

- J. DELORME, D. HOUZET, "Latency-constrained embedded benchmark for evaluation of cores mapping onto NoC architectures", RecoSoC, FRANCE Montpellier, juin 2005 (5 pages).

Rapports techniques - projet 4MORE

- J. DELORME, D. HOUZET, "Experience plan : part 1", 4MORE, contribution dans le cadre du WP4 : spécifications des blocs IP, 23 mars 2005 (61 pages).
- J. DELORME, D. HOUZET, "Experience plan : part 2", 4MORE, contribution dans le cadre du WP4 : études du contexte mono-composant, 19 septembre 2005 (24 pages).
- J. DELORME, D. HOUZET, "Experience plan : part 3", 4MORE, contribution dans le cadre du WP4 : études du contexte multi-composant (démonstrateur final), 23 février 2006 (22 pages).

Contribution au déroulement des JNRDM

La conférence JNRDM (Journées Nationales du Réseau Doctoral en Micro-électronique) est une conférence nationale organisée en FRANCE par les doctorants du domaine de la micro-électronique. En 2006, celle-ci s'est déroulée à Rennes sur le campus de Beaulieu. Le concept de cette conférence est de trouver des fonds auprès des écoles doctorales, des

organismes publiques et des industries afin de financer cette conférence qui prend en charge à 100% les frais des participants (transports, hébergement, inscriptions à la conférence, proceedings, ...).

Par conséquent, j'ai contribué au sein du comité d'organisation de cette conférence présidé par Mr Ludovic Barrandon [6] à différentes tâches permettant son bon déroulement. Outre le bon déroulement de cette conférence, ma contribution a été de participer à :

- l'élaboration du site Internet
- la réalisation du CD des actes de la conférence
- la réalisation de l'affiche
- la recherche de fonds auprès des industriels
- la responsabilité des transports des participants de la zone nord-est
- l'accueil des participants la veille de la conférence

Chapitre 1

Etat de l'art

Sommaire

1.1 Introduction

Depuis plusieurs années, l'interconnexion de composants électroniques a été une préoccupation majeure dans la conception et la réalisation d'un système électronique performant. Il y a une dizaine d'années, cette interconnexion se faisait au niveau des composants élémentaires du SoC par l'intermédiaire de technologies PCB (Printed Circuit Board ou circuits imprimés). Or du fait de la complexité croissante des SoC et dans un but de miniaturisation et de gain en performances, nous sommes maintenant face à une nouvelle problématique où l'interconnexion se fait à l'intérieur du circuit (intra-chip) avec des dizaines de coeurs (IP Intellectual Property) à connecter. De ce fait, un goulot d'étranglement des SoC (System On Chip) réside dans la gestion des communications entre ces différents blocs.

Ainsi, les réseaux d'interconnexions jouent un rôle majeur dans les performances des systèmes sur puce actuels et futurs. Plusieurs facteurs peuvent influer sur le choix d'une architecture appropriée puisqu'une interconnexion de système sur puce doit satisfaire certaines contraintes afin que celle-ci soit réalisable. Elle doit être performante, flexible, simple, respecter les contraintes de coût imposées par le cahier des charges et être physiquement implantable sur les technologies disponibles (FPGA ou ASIC par exemple).

Or, les SoC deviennent obsolètes de plus en plus rapidement de part leur durée de vie de plus en plus courte, et d'autre part, par la nécessité d'intégrer de plus en plus de standards le plus rapidement possible (Temps de mise sur le marché "Time to Market"). Ainsi, pour le marché des téléphones mobiles, on voit apparaître de plus en plus de fonctionnalités (GPRS, 3G, WLAN, Wi-Fi, Bluetooth, Appareil photo, Vidéo, lecteur mp3,...) qui nécessitent une intégration des anciens et des nouveaux standards à chaque mise sur le marché d'un nouveau produit avec un temps de conception et de fabrication de plus en plus court. Cette contrainte implique que tous les blocs fonctionnels anciens et nouveaux correspondant à ces différents standards puissent être intégrés rapidement dans le SoC tout en offrant une grande flexibilité et une bande passante assez conséquente pour satisfaire les exigences de toutes ces normes.

Nous allons donc voir dans un premier temps les différents modes de connexions disponibles pour les SoC actuels et futurs et la raison pour laquelle nous avons concentré nos études sur les réseaux sur puce (NoC pour Network on Chip). Ensuite, nous verrons les différentes architectures de réseaux existantes ainsi que les techniques de gestion de ceux-ci. Enfin, nous listerons quelques exemples de réseaux disponibles actuellement dans le monde de la recherche universitaire mais également dans le domaine de la recherche industrielle.

1.2 Les différents types d'interconnexions pour les SoC

Comme nous l'avons vu précédemment, pour satisfaire les contraintes de "time to market" de plus en plus exigeantes, plusieurs standards de communication existent ou émergent avec leurs avantages et leurs inconvénients. Nous allons dresser une liste des principaux modes d'interconnexions disponibles pour les SoC.

1.2.1 La connexion point à point

Il s'agit de la connexion la plus simple qui soit pour connecter deux IP (Intellectual Property). Les blocs fonctionnels sont donc reliés entre eux directement sans aucun protocole

de gestion de communications (cf. Figure 1.1). Cette solution impose d'avoir le même format d'encapsulation des blocs fonctionnels notamment concernant les interfaces d'entrées/sorties pour permettre une intégration rapide. Cette contrainte entraîne des limitations car certaines IP peuvent avoir besoin de communiquer avec des IP différentes en fonction des évolutions des besoins de l'application. De plus, cette technique est désastreuse lorsqu'un bloc fonctionnel est défectueux dans la chaîne, car l'ensemble du SoC devient inutilisable. C'est pourquoi la topologie bus a été adoptée pour pallier à ces inconvénients.

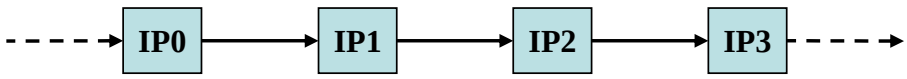

FIG. 1.1 – La connexion point à point

1.2.2 Le bus standard

Beaucoup de SoC utilisent des architectures à bus ou dite à "média partagé" car la structure d'interconnexion est la plus simple [7]. La topologie bus contrairement à la connexion point à point permet de connecter toutes les IP de l'application à un seul et même média de communication appelé bus. Cette solution permet d'avoir une encapsulation des blocs fonctionnels identique et d'avoir un protocole de communication plus flexible, contrairement à la connexion point à point (cf. 1.2.1). Ceci permet de faire évoluer un SoC beaucoup plus rapidement car il suffit simplement d'étendre le bus et de rajouter d'autres IP pour intégrer de nouvelles normes ou applications à un SoC. Les communications au sein de ce média de communication étant toutes gérées par l'arbitre du bus, les évolutions des besoins de l'application peuvent être gérées en temps réel. La figure 1.2 montre un exemple de topologie bus standard.

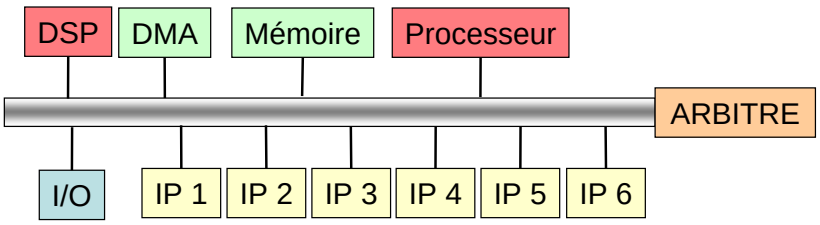

FIG. 1.2 – Topologie bus

Une amélioration de ce type de bus est appelée bus "Backplane" où 3 types de ligne au sein du bus existent. Celles-ci sont dédiées au transport des informations de contrôle, d'adresse et de données. Voici quelques exemples de bus utilisés dans les systèmes sur puce dans l'industrie :

– Le "AMBA-AHB" (Advanced High-performance Bus) de la société ARM [8]

- Le "Sonics SMART Interconnects" de la société SONICS [9]
- Le "PI-BUS" (Peripheral Interconnect Bus) du European OMI (Open Microprocessor Initiative) constitué par les sociétés Siemens, Philips, Matra-MHS, ARM et ST-INMOS [10] [11]
- Le "AVALON" de la société ALTERA [12]
- Le bus CoreConnect de IBM [13].

L'inconvénient majeure d'une topologie bus standard ou encore appelée "réseaux à média partagé" est la contrainte de ne pouvoir réaliser qu'une seule communication à la fois (les taches peuvent se dérouler en parallèle mais les communications ne se font que de manière séquentielle). Ces communications étant toutes gérées par l'arbitre du bus, on crée un goulet d'étranglement lorsque le nombre de communications croît, mais également lorsque les contraintes de bande passante de plusieurs communications deviennent importantes. Cet arbitrage joue un rôle prépondérant car c'est lui qui autorise les communications sur le bus mais il est également en charge de résoudre les conflits (plusieurs requêtes de communications en même temps). Cet arbitrage implique donc une limitation sur le nombre d'IP connectées au bus à une dizaine d'éléments [14].

De plus, ce mode d'interconnexion physique devient un facteur limitant en performance pour les SoC actuels et futurs car les lignes du bus deviennent de plus en plus longues à mesure que le nombre d'IP connectées augmente. On voit ainsi apparaître des capacités parasites qui engendrent une augmentation du temps de charge (proportionnel au nombre d'éléments raccordés). La fréquence de fonctionnement du bus s'en trouve donc réduite et la consommation électrique augmentée. Enfin, le temps de propagation des signaux sur les longs fils de données et de contrôle du bus entraîne une dérive des horloges et engendre intrinsèquement des problèmes de synchronisation (longueur des interconnexions, bande passante, consommation d'énergie). Compte tenu de la complexité croissante des SoC intégrant plus d'une cinquantaine d'IP avec des bandes passantes importantes, les systèmes à base de média partagé deviennent un goulot d'étranglement en terme de performance mais également de consommation et de complexité de mise œuvre physique sur le silicium.

1.2.3 Les bus hiérarchiques

Pour solutionner les problèmes du bus standard ou dit "Backplane" énoncés précédemment, une solution permettant de contourner les limites de cette topologie est de créer un bus hiérarchique (figure1.3).

Le concept de ces bus hiérarchiques est de faire communiquer plusieurs bus entre eux par l'intermédiaire d'un pont ("Bridge") reliant deux bus entre eux. L'intérêt de cette topologie hiérarchique est de permettre d'ordonnancer les différentes tâches qui composent le SoC en les répartissant sur les différents bus qui la composent. Ceci permet d'équilibrer les charges des communications de l'application de façon à ne pas surcharger un seul et même bus mais également de permettre un ordonnancement des tâches de l'application. Les différents segments du bus hiérarchique possèdent donc des longueurs courtes avec peu d'IP connectées, offrant une faible capacité sur chacun de ces segments et permettant d'utiliser chaque bus à des fréquences élevées. Ainsi, les transactions peuvent se faire en parallèle sur les différents segments de bus et à fréquence élevée. De cette façon, on offre une qualité de service plus élevée à l'application et une meilleure bande passante globale.

Cependant, l'accès d'un bus à un autre au travers d'un pont, implique un coût en latence d'où l'importance de bien répartir les différentes IP communicantes pour limi-

ter l'utilisation du pont et ainsi favoriser le fonctionnement parallèle des bus. Ainsi, on distingue trois types principaux de bus standard constituant ces bus hiérarchiques :

- Les bus processeur/mémoire : Ils sont dédiés aux transferts de données entre processeur et mémoire cache. Ils sont très rapides et offrent donc une grande bande passante.
- Les bus entrée/sortie : Ils permettent d'adapter beaucoup d'interfaces de communication.
- Les bus classiques : Il s'agit de la topologie bus classique sur laquelle on retrouve processeurs, éléments mémoires et entrées/sorties n'ayant pas besoin d'une grande bande passante.

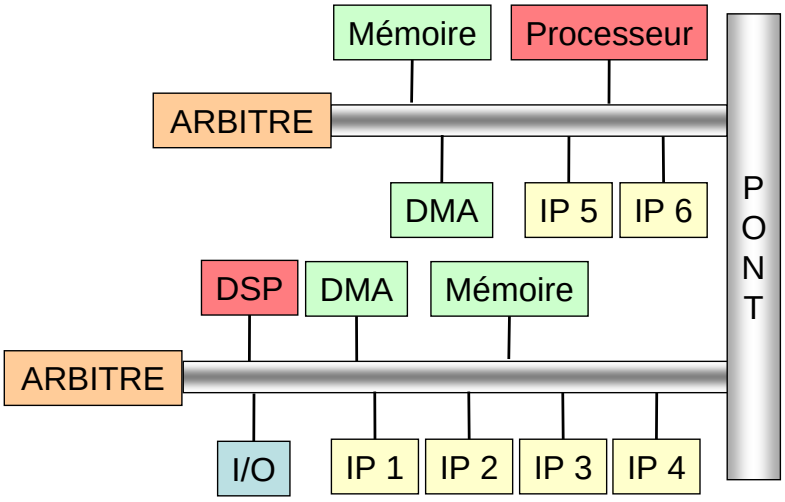

FIG. 1.3 – Topologie bus hiérarchiques

Ces trois catégories permettent un ordonnancement général des taches qui convient dans la plupart des applications SoC que l'on peut rencontrer. Ceci sous contrainte de besoin en bande passante entre les tâches de l'application. Un exemple de bus hiérarchique est le "AMBA-APB" [8] (Advanced Peripheral Bus) de la société ARM se connectant au "AMBA-AHB" qui est présenté sur la figure1.4. On peut également citer un autre bus hiérarchiques similaire qui est le bus CoreConnect de IBM [15].

Enfin, le bus hiérarchique consomme en principe moins que le bus partagé puisque la capacité des éléments connectés au bus est plus faible sur chaque segment du bus. Ce découpage en segment est un premier pas qui tend vers l'approche réseau mais le goulot d'étranglement que l'on peut observer au niveau des ponts devient un facteur limitant compte tenu des besoins en bande passante des futures applications. C'est pourquoi les topologies réseaux ou "cross-bar" offrent une alternative intéressante pour les futurs SoC.

1.2.4 Le crossbar

Une alternative offrant un compromis intéressant entre les topologies bus et celles des réseaux est le crossbar. Dans ce cas, tous les blocs fonctionnels de l'application sont reliés

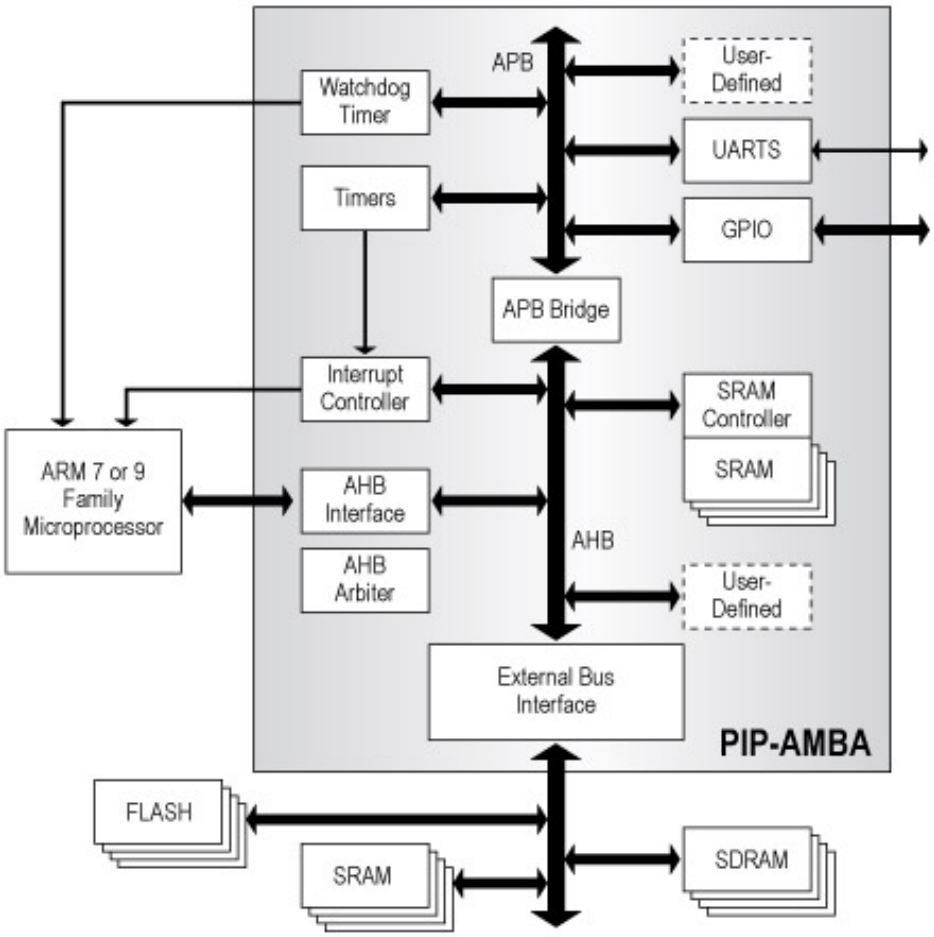

FIG. 1.4 – Exemple de topologie bus hiérarchiques pour des bus AMBA

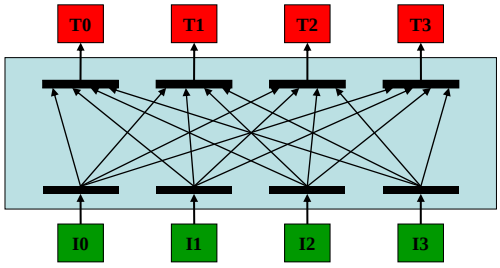

FIG. 1.5 – Topologie d'un crossbar 4 vers 4

les uns aux autres par l'intermédiaire du crossbar (figure 1.5). Celui-ci a l'avantage de permettre des communications parallèles (contrairement au bus) et d'offrir une grande bande passante pour chaque communication car celles-ci ne sont plus partagées avec les autres communications comme c'est le cas dans la topologie bus standard ou hiérarchique. Cependant, la complexité de câblage d'une telle architecture croit en fonction du nombre d'IP qui composent l'application. Celle-ci augmente avec le carré du nombre d'éléments communicants soit une complexité de $O(n^2)$, ce qui devient vite exorbitant.

1.2.5 Le réseau

Les topologies réseaux arrivent naturellement comme une solution de remplacement des standards actuels de communication inter-IP dans les SoC. Compte-tenu des contraintes de bande passante de plus en plus grandes dans les applications actuelles et surtout futures, il est donc indispensable de proposer de nouvelles architectures d'interconnexions. Les réseaux d'interconnexion sont étudiés depuis plusieurs décennies notamment pour les réseaux informatiques, les calculateurs parallèles, les réseaux téléphoniques ou encore les interconnexions sur PCB [16] [17]. Cependant depuis une dizaine d'années, nous avons vu une évolution rapide de la technologie d'interconnexion des systèmes sur puce, en particulier, depuis [18] qui a proposé de remplacer les bus par des réseaux sur puce avec des interconnexions à base de routeurs.

Les composants de base du NoC sont donc les suivants :

- Les NI (Network Interface) ou interface réseau : elles réalisent l'interface entre le protocole du NoC et celui des blocs IP qui sont connectés au routeur. Leur rôle est de séparer le traitement (effectué dans les IP) des communications (gérées par le réseau). Un adaptateur réseau peut être composé de deux parties, l'une prenant en charge l'interface réseau en elle-même, l'autre réalisant l'adaptation de protocole aussi appelée "wrapper".
- Les routeurs : ils aiguillent les paquets de données dans le réseau en fonction du protocole choisi en respectant une stratégie de routage qui constitue l'arbitrage du routeur.
- Les liens : ils relient les routeurs entre-eux ou les routeurs aux NI. Ils offrent la bande passante pour les communications entre la source et la destination. Celui-ci peut posséder plusieurs canaux virtuels et peut être mono-

directionnel ou bi-directionnel. Dans la plupart des NoC que nous verrons dans la section 1.5, les liens sont de type bi-directionnel.

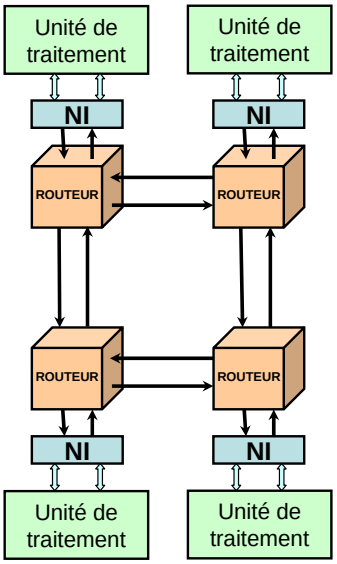

FIG. 1.6 – Topologie Réseau

Les NoC sont susceptibles de proposer des solutions efficaces aux problèmes d'intégrations complexes des systèmes sur puce [19]. Ces architectures d'interconnexion devront tout de même faire face à de nombreuses contraintes lors de leurs phases de conception comme celles listées ci-dessous :

- consommation d'énergie
- surface de silicium
- performances
- synchronisation
- électrique

De plus, selon les applications considérées, le coût et les caractéristiques de mise en œuvre du réseau peuvent varier grandement. De ce fait, l'architecture du réseau ainsi que ses modes de commutation et de gestion de flux sont des caractéristiques importantes à prendre en compte pour dimensionner au mieux l'architecture avec l'application.

1.3 Les caractéristiques des réseaux

On peut trouver dans la littérature différentes topologies de réseau apportant des avantages et des inconvénients par rapport aux contraintes énoncées dans la section 1.2.5. Nous allons donc voir les principales structures que l'on peut trouver dans les NoC du domaine de la recherche universitaire et de la recherche industrielle.

1.3.1 Les topologies de réseaux

Étant donné l'aspect répétitif d'un réseau au niveau de son architecture, plusieurs topologies existent dans la littérature afin d'offrir le meilleur compromis possible entre les performances requises par l'application et le coût matériel engendré par le réseau lui-même. Nous allons voir quelques exemples de topologies fréquemment rencontrées.

La topologie la plus souvent employée est celle de structure 2D régulière (2D mesh) représentée sur le figure 1.7. Elle est simple de mise en œuvre et facilement implantable sur une technologie silicium. Les algorithmes de routage sont simples à instaurer et elle offre une grande scalabilité (capacité d'accroître facilement la structure matérielle pour répondre à une exigence de performances).

Le torus (figure 1.8) est une topologie dérivée de la structure 2D qui possède la particularité d'un repliement des bords extérieurs sur eux même, offrant une bande passante légèrement supérieure. Cette topologie est utilisée dans [14, 20, 21].

La topologie 3D (figure 1.9) quant à elle offre une plus grande bande passante que les deux autres mais elle souffre d'un inconvénient majeur qui est la difficulté de son implantation sur silicium (routage complexe).

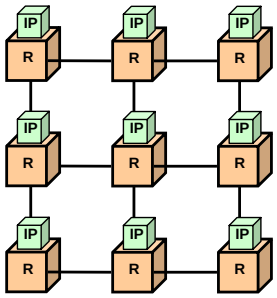

FIG. 1.7 – Topologie 2D

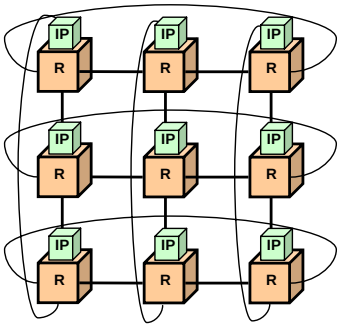

FIG. 1.8 – Topologie 2D Torus

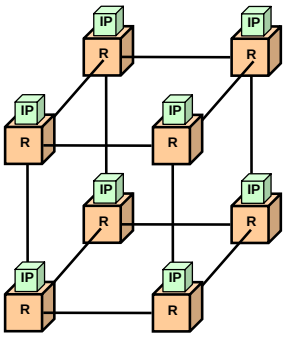

FIG. 1.9 – Topologie 3D

La topologie en arbre élargi représentée sur la figure 1.10 (dite "Fat-tree") a pour intérêt d'être scalable et d'offrir une latence pouvant être plus faible que la topologie en grille 2D. Cette architecture nécessite un bon placement des IP sur le réseau car les routeurs intermédiaires deviennent rapidement des goulots d'étranglement lorsque plusieurs feuilles d'une même branche veulent communiquer avec des feuilles d'autres branches. C'est précisément dans ce cas que cette topologie trouve ses limites. De nombreux travaux sur ce type d'interconnexion sont menés par le LIP6 (Université Pierre et Marie Curie)[22, 23, 24, 25].

La topologie en anneau, représentée sur la figure 1.11, est facile à mettre en œuvre au niveau de l'algorithme de routage et de l'implantation physique car les routeurs sont reliés uniquement à leurs deux voisins par des liens unidirectionnels. Mais cette topologie souffre d'un inconvénient majeur, car pour un réseau de grande taille, le message doit traverser un nombre important de routeurs, engendrant inévitablement une limite de bande passante. Une alternative proposée à la limitation de la topologie en anneau, est celle de l'octogone représentée sur la figure 1.12. Utilisée dans [26], elle permet de réduire la latence en offrant la garantie de ne traverser au maximum que deux routeurs pour n'importe quelle communication au sein du réseau.

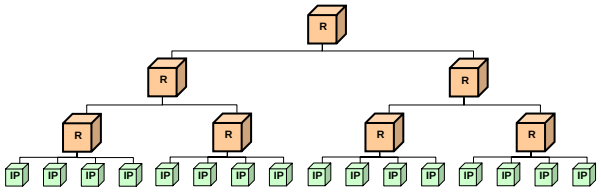

FIG. 1.10 – Topologie arbre

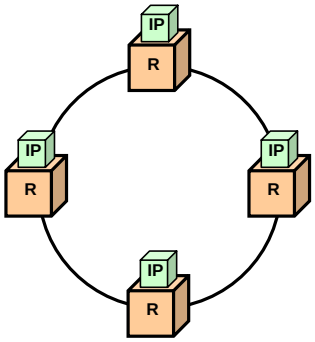

FIG. 1.11 – Topologie anneau

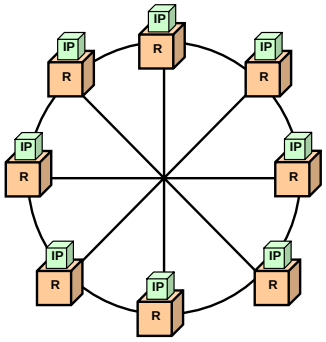

FIG. 1.12 – Topologie octogone

Dans la pratique, les NoC de la littérature utilisent très généralement des topologies régulières car elles présentent l'intérêt d'être une structure mathématique simple, simplifiant l'utilisation des règles de routage. Cependant, en pratique, le placement des IP lors de l'implantation sur silicium permet rarement d'intégrer une topologie régulière. En effet, les surfaces des IP étant complètement différentes les unes des autres, cela entraîne une déformation de la grille 2D lors de leur placement sur les routeurs. C'est pourquoi, on peut observer, dans la littérature, différents travaux permettant de faire l'AAA en offrant la possibilité de réaliser des topologies irrégulières. Une topologie irrégulière permet donc plus de liberté en dimensionnant précisément le réseau requis[27, 28]. Elle peut être issue d'une topologie régulière qui a été retaillée au plus juste de façon à enlever les éléments non utilisés ou bien être construite irrégulièrement dans sa phase de conception.

Une topologie irrégulière nécessite en revanche une plus grande attention pour le routage car les règles à appliquer ne sont plus triviales. Ainsi, compte-tenu des avantages apportés par la structure régulière 2D, nous privilégions l'utilisation de celle-ci pour nos simulations et implantations futures. Nous allons voir maintenant les modes de commutations disponibles au sein des réseaux sur puce.

1.3.2 Les modes de commutations

Afin de mieux situer les avantages des différents modes de commutation nous allons rappeler les définitions de base caractérisant le format des données des communications au sein d'un NoC. En effet, lorsque l'on veut effectuer une communication, l'information que l'on veut transmettre est appelée *message*. Il correspond à la totalité de l'information que

l'on veut transmettre pour une communication. Celle-ci peut être découpée sous forme de paquet ("*packet*") afin de permettre un parallélisme des communications lorsqu'une application le requiert mais également lorsque la longueur du message devient trop grande. Ce paquet est lui-même décomposé sous forme de FLIT (**FL**ow control un**IT**) qui constitue le plus petit élément de base constituant les paquets ou les messages. L'ensemble de ces éléments est représenté sur la figure 1.13.

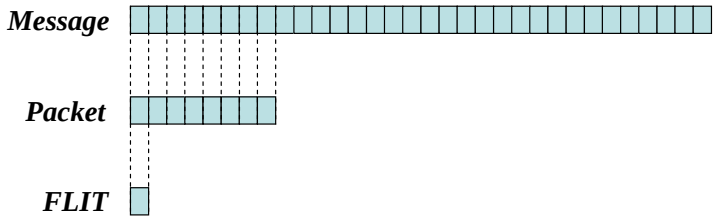

FIG. 1.13 – Format des éléments de base caractérisant les communications dans les NoC

Ainsi, pour gérer les communications au sein d'un réseau sur puce, différents mécanismes de base existent, s'appuyant sur les définitions présentées ci-dessus. Parmi ces modes de commutation, on peut en retenir deux principaux qui sont : le circuit switching (commutation de circuit) et le packet switching (commutation de paquet).

1.3.2.1 Le circuit switching

La commutation de circuit est une méthode de transfert de données consistant à établir un circuit dédié au sein d'un réseau pour chaque paire d'émetteur/récepteur. Le contrôle est réalisé par le réseau pour effectuer et paramétrer une connexion. Ce principe est donc le même au sein d'un NoC où plusieurs ressources sont connectées à une structure de routeurs qui vont se charger de faire les connexions entre les différentes ressources qui souhaitent communiquer entre elles. Les systèmes de commutation de circuits sont idéaux pour les communications qui exigent des transmissions de données en temps réel, ils sont parfois appelés réseaux orientés connexion.

Un chemin dédié est initialisé pour réaliser la connexion entre la source et la destination. Le chemin peut être réservé soit au moyen de signaux de contrôle auxiliaire au réseau, soit par un paquet contenant l'information de routage mentionnant l'adresse de destination ainsi que des informations de contrôle. Dans ce dernier cas, la source émet ce message à travers le réseau jusqu'à arriver à l'adresse de destination et réserve les liens qu'il traverse au fur et à mesure de sa progression au sein du réseau. Ceci permet donc de bloquer les liens afin de réaliser une ligne dédiée entre les deux ressources. La transmission du message se fait donc par les liens qui ont été réservés lors de l'établissement de la connexion. La terminaison de la communication réservée peut être libérée soit par la source, soit par la ressource cible soit par le dernier paquet de la transmission.

La latence de ce type de commutation dépend du délai d'établissement de la connexion (réservation du chemin) mais également du délai de transmission des données qui est conditionné par la taille du message. Ce mode de commutation fournit une très grande Qualité de Service (QoS), un trafic garanti (GT voir section1.3.4.1), mais il est très pénalisant du point de vue ressource car un chemin entier est dédié pour une communication. Toute

autre communication voulant s'établir en empruntant le même chemin ou une partie de celui-ci ne pourra être satisfaite. Cette technique est appropriée pour les communications peu fréquentes et également pour les transmissions de grandes quantités de données.

Le réseau à commutation de circuit le plus omniprésent est le système de réseau téléphonique, qui consiste à relier l'ensemble des fils du réseau pour n'en former qu'un seul afin de restituer une image d'une ligne ininterrompue simple pour chaque appel téléphonique ; c'est le Réseau Téléphonique Commuté (**RTC**). En effet, en réservant une ligne téléphonique entre deux abonnés, il est possible de garantir la meilleure performance possible pour le transfert des données. Dans le cas d'une communication vocale par exemple, il est essentiel que la ligne ne soit pas coupée pendant tout le temps de la transmission.

1.3.2.2 Le packet switching

Dans le cas d'un mode de commutation par paquets dit "packet switching", le message est découpé en paquets dont chacun est composé d'une partie contrôle appelée l'en-tête et d'une partie donnée qui est l'information de la communication à transporter. Les routeurs du réseau analysent et modifient l'en-tête des paquets qui les traversent sur leurs ports et orientent le flux de données sur les ports de sortie ciblés par l'en-tête du paquet. Ainsi, les paquets ne constituent que des blocs de petite taille et il n'y a plus besoin de la phase d'allocation de ressources qui est nécessaire dans le "circuit switching". En contre-partie, cette technique implique la nécessité de devoir stocker l'intégralité du paquet dans le routeur avant de prendre une décision d'aiguillage des données sur les ports de sorties.

Lorsqu'un routeur reçoit un paquet, il le stocke dans son buffer, vérifie l'adresse de destination dans l'en-tête du paquet, sélectionne un routeur se trouvant dans la destination de routage (indiqué dans l'en-tête du paquet) et envoie le paquet au routeur suivant si le lien qui le relie est disponible. Ainsi l'inconvénient lié à la taille des messages pouvant être variable lors des communications en technique de commutation de circuits dit "Message Switching" et rendant les mémorisations au sein des routeurs difficiles à mettre en œuvre, est levée. Ce principe de commutation par paquet appliquée au NoC apporte un gain en performance sur le débit du réseau, une diminution de la latence et une réduction de la taille des buffers au niveau des routeurs.

Par conséquent, les réseaux de commutation par paquets sont plus efficaces si un certain retard dans la transmission des données est tolérable et acceptable mais nécessite un besoin de ressource matériel supplémentaire dû au stockage temporaire des paquets dans les routeurs. Naturellement, la qualité de service fournit dans le cas d'un réseau de commutation par paquets est de type Best Effort (BE) exploitant mieux les ressources matérielles mais ne garantissant pas le débit. Cette qualité de service sera détaillée dans la section 1.3.4.2.

1.3.3 Les Mécanismes de gestion de flux des communications

Comme nous l'avons vu précédemment, le mode de commutation de circuit ne peut pas entraîner de conflit puisque la communication est établie par le réseau lui-même. Par conséquent, il ne peut y avoir de conflit avec les autres communications sur le même lien puisque celle-ci est dédiée à l'application en cours. Par contre dans le cas de la commutation par paquets, un mécanisme de contrôle de flux de données est nécessaire car les données des paquets doivent être mémorisées dans chaque routeur du réseau qui est traversé avant

d'être envoyées sur le routeur suivant. A cause de leur espace de mémorisation limité, les routeurs ne stockent les données que lorsqu'ils ont suffisamment d'espace mémoire pour sauvegarder les données entrantes. Ainsi, il existe plusieurs mécanismes de gestion de contrôle de flux de données au sein d'un réseau sur puce dont les quatre principaux sont les suivants :

- Store-and-forward
- Virtual Cut-Through
- Wormhole
- Time division

1.3.3.1 Store-and-forward

Dans cette technique de contrôle de flux de données, la communication au sein des routeurs se fait par une mémorisation complète du paquet entrant sur le port d'entrée du routeur avant d'être expédié vers un autre routeur du réseau. Ceci implique le besoin d'avoir une mémoire sur chaque port entrant du routeur ayant une taille équivalente à celle du paquet reçu. Ceci implique également une latence par routeur qui est au moins égal au temps nécessaire pour recevoir le paquet entrant et le stocker. Le principe de cette gestion de flux SF (Store and forward : stocke puis renvoie) est décrite sur la figure 1.14. T_r correspond au temps nécessaire au routeur pour prendre sa décision de routage mais aussi le temps d'attente pour obtenir le lien du routeur cible. T_m représente le temps de mémorisation du paquet au sein du routeur, celui-ci étant le même pour tous.

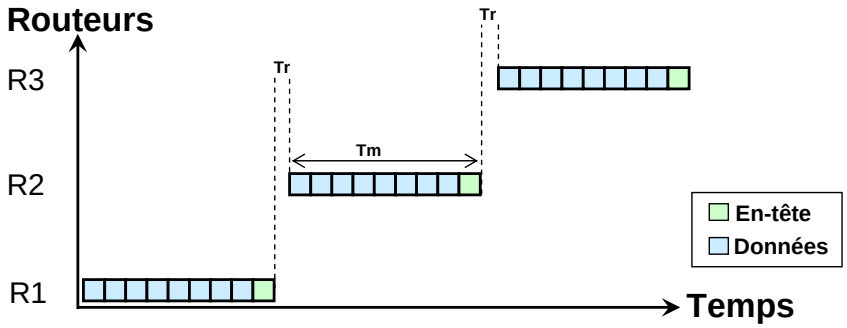

FIG. 1.14 – La technique "Store and forward"

Dans cette approche, le routeur source attache un en-tête au paquet à envoyer qui contient l'adresse de destination de la ressource cible et l'envoie au lien de sortie si celui-ci est disponible. Ensuite, lorsque le routeur intermédiaire le reçoit, il le stocke dans son buffer, vérifie l'adresse de destination dans l'en-tête du message, sélectionne le routeur suivant dans la destination de routage et lui envoie le message si celui-ci est disponible, c'est à dire si le lien vers le routeur suivant est libre. Les décisions de routage sont effectuées par l'arbitre du routeur qui dans la plupart des cas est à priorité tournante[29].

Cette technique est avantageuse quand les messages sont courts et répétitifs. Par contre, le besoin de stocker l'intégralité du message dans le routeur devient coûteux au niveau de la complexité du routeur car ceci engendre une taille de paquet limitée par la taille mémoire

sur chaque port entrant. Nécessairement, la taille des paquets de données est limitée ce qui peut limiter l'évolution des contraintes de l'application lorsque l'on souhaite augmenter la qualité de service en augmentant la taille des paquets par exemple. Cette technique implique nécessairement la recherche du chemin le plus court possible lors de la phase de conception.

Dans les réseaux informatiques, cette technique est utilisée pour délivrer des messages à des réseaux qui ne sont pas accessibles en permanence. Elle est également employée pour jouer sur le décalage horaire, pour que des informations arrivent aux heures ouvrables à leurs destinataires, ou au contraire pour qu'elles soient acheminées la nuit afin de bénéficier de tarifs réduits de télécommunication

1.3.3.2 Virtual Cut-Through (VCT)

La technique du "Virtual Cut-Through" (VCT) s'appuie sur les mêmes bases que le SF, mais se différencie par la possibilité d'émettre le paquet avant même d'avoir stocké l'intégralité du paquet dans la mémoire du port entrant du routeur. Le paquet entrant sur un routeur est expédié vers l'autre routeur cible dès que celui-ci garantit que la totalité du paquet peut être acceptée par ce dernier. Quand aucune garantie n'est donnée, le routeur doit pouvoir stocker l'ensemble du paquet, c'est-à-dire basculer dans un mode de gestion de flux classique SF. Ce contrôle de flux VCT nécessite aussi d'avoir un espace mémoire équivalent à la taille d'un paquet de données pour le pire cas. Cette technique est donc similaire à celle du store-and-forward mais elle permet de réduire la latence des communications car on réduit le temps d'attente lors du passage d'un routeur car il n'est plus nécessaire de stocker l'intégralité du paquet avant de prendre une décision de routage sur le port de sortie. Cette technique est détaillée sur la figure1.15.

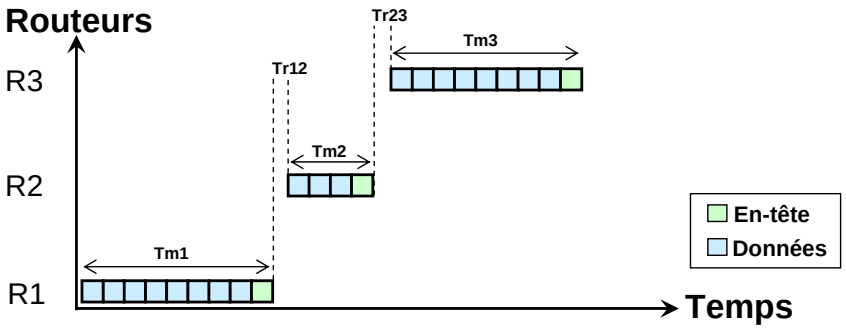

FIG. 1.15 – La technique "VCT"

Où T_{r12}, T_{r23} correspondent au temps de décision de routage pour passer respectivement du routeur 1 à 2 et 2 et 3. T_{m1}, T_{m2}, T_{m3} représentent le temps de mémorisation du paquet au sein des routeurs 1, 2 et 3. Ce temps de mémorisation pouvant varier en fonction du temps de routage. Avec la technique VCT, plus le temps de routage est court et plus le temps de mémorisation sera court car le paquet ne sera pas automatiquement mémorisé dans le routeur et par conséquent on diminue la latence.

Cette technique permet de réduire le problème de latence présent dans le SF mais reste quand même coûteuse au niveau ressource matérielle. En effet, il est toujours nécessaire d'avoir une mémoire équivalente à la taille du paquet pour chaque port entrant du routeur.

1.3.3.3 Wormhole

Dans cette technique, la capacité de mémorisation sur les ports entrant est réduit à la taille d'un FLIT (Cf. 1.3.2). Comparé aux deux techniques présentées précédemment, le Wormhole permet de diminuer considérablement les besoins en mémoire dans les routeurs et indirectement la surface en silicium .

Le FLIT d'en-tête contenant l'information de chemin est passé à un autre routeur lorsque le lien ciblé est disponible, c'est à dire lorsque le buffer sur le port d'entrée est libre. Dès que le FLIT d'en-tête est émis sur le port de sortie d'un routeur, ce chemin est automatiquement réservé aux autres FLIT de données appartenant au même paquet. Ceux-ci suivent donc le chemin emprunté par le FLIT d'en-tête. Il faut également préciser que lorsque le premier FLIT d'un paquet est bloqué dans un routeur, les FLIT suivants constituant le paquet sont répartis sur les autres routeurs bloquant ainsi les liens intermédiaires. La technique du Wormhole est représentée sur la figure 1.16.

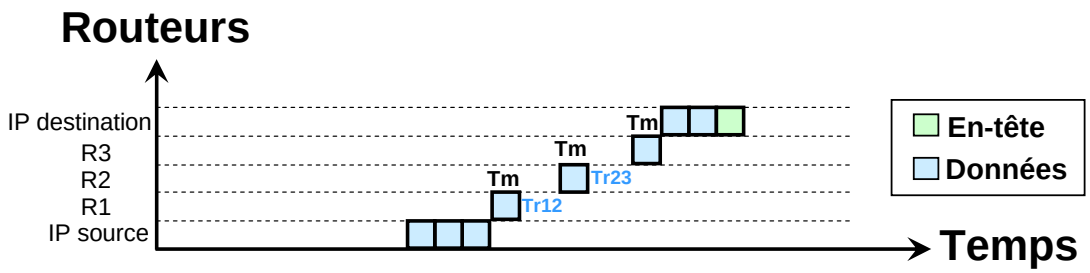

FIG. 1.16 – La technique wormhole

Par conséquent, cette technique offre l'avantage d'avoir une latence de communication plus faible puisque celle-ci ne mémorise pas le paquet en entier dans le routeur, contrairement au SF et au VCT. Elle permet également de réduire les coûts en ressource matérielle (buffer de 1 FLIT sur chaque port entrant du routeur) mais en contrepartie, un inconvénient apparaît : elle est plus sensible aux deadlock [30]. En effet, le paquet est ainsi acheminé en pipeline sur plusieurs routeurs ce qui peut conduire, en cas de contention d'un des paquets dans le réseau, à une contention en cascade de tout le réseau, et entraîner une étreinte mortelle (deadlock).

La technique Wormhole associée à un réseau de type packet switching est la plus populaire, car elle a le moindre coût en mémoire dans les routeurs et la plus faible latence d'acheminement des données au travers du réseau, ceci malgré un risque de contention accru. Dans la suite du document, nous ne nous intéresserons qu'à cette technique.

1.3.3.4 Time division circuit switching

Cette technique permet de faire un compromis entre les avantages des modes de commutations "Circuit switching" et "Packet switching" pour permettre de garantir des bandes passantes pour les communications le tout en étant en mode "Packet switching".

En effet, comme nous l'avons vu précédemment, la technique "circuit switching" bloque tous les liens empruntés par une communication empêchant d'autres communications d'être multiplexées temporellement sur les mêmes liens. Ainsi, la solution apportée par la méthode TDMA [31] est de conserver les avantages du mode "Packet switching" en gestion de flux de type "Wormhole" mais en corrigeant les inconvénients (risques de deadlock et de latence). Pour corriger ces inconvénients, la solution est de réserver des tranches de temps pour les différentes communications que l'application va demander. De ce fait, chaque routeur possède une table d'allocation pour toutes les communications de l'application qui vont lui permettre de prendre des décisions d'arbitrage en tenant compte de l'aspect temporel. On a donc un ordonnancement des communications qui va permettre de garantir des bandes passantes pour chaque communication mais sur de courtes périodes de temps. On transforme donc un mode de commutation "Packet switching" en "Circuit switching" virtuel, c'est-à-dire que le réseau est vu de type "Circuit switching" mais sur de courtes périodes. Ainsi, les contraintes de bandes passantes peuvent être garanties pour l'application. Les NoC Ætheral, Nostrum ou encore aSoC ont adopté ce type de gestion de flux.

Par contre, cette technique entraîne une complexité supplémentaire au niveau des routeurs car il faut intégrer les tables de routage mais également modifier l'arbitrage de ceux-ci. Un autre problème peut apparaître dans cette technique qui est la nécessité pour chaque tâche matérielle ou logicielle connectée sur le réseau d'avoir ses données prêtes à être envoyées sur le réseau lorsque la tranche de temps allouée à cette communication arrive. Si ce n'est pas le cas, on a une bande passante réservée qui est sous utilisée et des performances moindres que dans le cas d'un réseau de type "Wormhole" "Packet switching" malgré ses faiblesses au niveau des risques de deadlock et de latence variables et non prédictibles.

Par la suite, pour nos simulations et expérimentations, nous avons choisi de prendre un réseau de type "Wormhole" "Packet switching".

1.3.4 Les qualités de services dans les réseaux sur puce

Comme nous l'avons vu ci-dessus, suivant le type de réseau utilisé, on voit apparaître des qualités de service (QoS) plus ou moins bonnes pour les applications interfacées au réseau. Parmi les QoS les plus répandues dans les NoC on trouve le Best Effort (BE) et le Guaranteed Traffic (GT).

1.3.4.1 Guaranteed Traffic(GT)

Il s'agit d'une technique permettant une réservation d'allocation de ressources pour les scénarii pire cas. Typiquement, pour une application exigeante en bande passante, le réseau permet de réserver la bande passante nécessaire pour permettre de respecter les contraintes temps réel lorsque les besoins sont ponctuels. L'inconvénient de cette réservation réside dans le fait que cette allocation est effectuée pour des conditions pire cas où la quantité de données à traiter est importante. Or si le flux de données est modifié et que la bande passante exigée ensuite devient plus faible, alors la ressource est sous utilisée. On n'a

donc pas une utilisation maximale des ressources disponibles des liens dans un NoC en trafic garanti. Par contre, en GT, avec une bonne adéquation Algorithme architecture, les contraintes réelles de l'application seront toujours respectées même si l'exploitation des ressources offertes par le réseau est sous utilisée. Pour une meilleure utilisation de ces ressources, il faut se tourner vers une qualité de service dite en Best effort détaillée dans la partie ci-après.

1.3.4.2 Best Effort(BE)

La technique BE quant à elle ne réserve aucune ressource ce qui, par conséquent, ne garantit aucun débit ni respect de contraintes temps réel (notamment pour les pires cas...). Ceci ayant pour intérêt de simplifier la complexité des routeurs et de permettre une exploitation maximale des ressources du réseau. Par contre, cette technique permet une meilleure utilisation des ressources car le dimensionnement de celles-ci est effectué par rapport à la moyenne des contraintes de tous les scénarii, contrairement au GT qui s'effectue sur le scenario pire cas. On voit bien ici que le BE permet une meilleure utilisation des ressources du réseau dans des contextes multi-standards, c-à-d multi-scénarii.

Certains travaux ont été menés afin de donner la possibilité au réseau d'exploiter les qualités de chacun de ces services en fonction de l'application et de faire un compromis adéquat par rapport aux performances requises [32].

1.4 Les techniques de placement et de routage appliquées au NoC

Comme nous l'avons vu précédemment, il existe différents types de réseaux disponibles dans le domaine de la recherche académique et industrielle, chacun ayant leurs avantages et leurs inconvénients. Nous avons opté pour un réseau de type "Wormhole" "Packet switching" pour les caractéristiques que l'on connaît, mais également parce que ce type de réseau était employé dans le projet 4MORE [21] dans lequel nous étions impliqués (cf. 1.5.2) et pour lequel nous avons contribué au sein du WP4 en charge de l'exploration architecturale du SoC du terminal mobile de 4^{e} génération.

Les réseaux sur puce souffrent d'un inconvénient majeur induit par leur difficulté et leur complexité de mise en œuvre. En effet, les outils de conception doivent permettre de dimensionner l'architecture en adéquation avec l'application tout en prenant en compte plusieurs critères séquentiellement ou parallèlement dans la phase d'AAA (Adéquation Algorithme Architecture). Or les critères de dimensionnement d'un réseau sur puce sont nombreux et mènent vers des explorations longues et coûteuses en temps de simulation. Ainsi, pour trouver un compromis entre la meilleure adéquation possible et le temps d'exploration de l'architecture, on trouve dans la littérature plusieurs techniques de placement (mapping) et de routage des IP sur le réseau permettant de faire l'AAA. Ces techniques permettent de paramétrer au mieux le NoC en terme de placement des IP sur la matrice mais également dans le choix des chemins de routage des communications entre IP. Elles permettent de contribuer à améliorer la phase d'AAA en prenant en compte plusieurs critères qui peuvent être utilisés en parallèle ou bien de manière séquentielle pour raffiner l'espace d'exploration et s'approcher de la solution la plus optimale. Le choix de notre technique de placement routage sera décrite dans le chapitre 3, elle s'inspire des travaux ef-

fectués par [33, 34, 35]. Ces techniques se différencient les unes des autres par leur manière d'aborder l'espace d'exploration mais aussi par leur intégration de critères de consommation ou de fréquence d'horloge devenant important dans les SoC actuels et futurs. Nous allons donc voir les principales techniques de placement et de routage développées dans la littérature en fonction des types de NoC utilisés.

1.4.1 La technique UMARS

La technique de mapping **UMARS** (Unified Mapping, Routing and Slot allocation) [31] est celle utilisée par Philips research laboratories. Cette approche permet de réaliser une AAA sur un NoC TDMA en prenant en compte trois critères primordiaux examinés en même temps. Il s'agit ici de prendre des décisions de placement des IP et de routage des communications conjointement, tout en considérant les contraintes de bande passante des applications en rapport avec celle disponible dans le réseau. Cette approche se compare à une seule technique dite "originale" qui est celle présentée dans [36].

Les NoC garantissant une qualité de service (QoS) possèdent au sein de l'architecture de leurs routeurs, des tables d'allocations permettant de garantir des communications avec des performances maintenues sans perte de données, ceci en garantissant une bande passante minimum ainsi qu'une latence maximum [36].

Les communications garanties dans un NoC ont des impacts positifs sur le flot de données. En effet, le NoC et les blocs IP peuvent être découplés ce qui signifie que le comportement d'une IP lors de communications ne peut pas influer sur celui des autres IP présentes sur le même NoC. Par ce biais, tous les blocs IP peuvent être modélisés et validés indépendamment les uns des autres (contrairement au cas des NoC en BE où toutes les IP ainsi que le NoC doivent être simulés conjointement). Ainsi, un modèle de performance du NoC peut être utilisé dans le but de générer une application spécifique au NoC qui va obtenir ses contraintes en communication dans toutes les circonstances. La contrainte dans cette garantie de services se retrouve sur les IP où les contraintes en communications doivent être décrites de manière détaillée. Cette information est normalement toujours disponible dans le cahier des charges de la conception d'un SoC.

1.4.2 La technique BB

La technique **BB** (Branch and Bound) est un algorithme d'optimisation déterministe appliqué au placement d'IP sur une topologie 2D ayant pour objectif de minimiser le coût en énergie tout en garantissant les contraintes de bande passante pour un réseau de type "Wormhole" "Packet switching". Dans cette approche [34, 33], la technique est basée sur une architecture 2D mesh de taille n × n figée sur laquelle on veut venir placer une application. Chaque routeur possède 5 liens bi-directionnels dont 4 sont connectés à des routeurs voisins différents et un connecté à l'IP de traitement représentant une tâche de l'application. Un modèle d'énergie au niveau bit est intégré au niveau de l'algorithme consistant à caractériser l'énergie consommée par un bit d'information dans les routeurs ($E_{S_{bit}}$) et dans les liens ($E_{L_{bit}}$) entre ceux-ci. Ainsi, le modèle appliqué suit l'équation 1.1 :

$$E_{bit} = E_{S_{bit}} + E_{L_{bit}} \tag{1.1}$$

Par conséquent, le modèle de coût moyen de consommation d'énergie pour un bit circulant d'un routeur t_i à un autre routeur t_j est modélisé dans 1.2 :

$$E_{bit}^{t_i,t_j} = n_{hops} * E_{S_{bit}} + (n_{hops} - 1) * E_{L_{bit}} \tag{1.2}$$

Où n_{hops} correspond au nombre de liens à parcourir entre les routeurs t_i et t_j.

Ensuite, l'algorithme se base sur une description de l'application et de l'architecture sous forme de graphe. L'AAA se fait par une intersection entre les deux graphes en ayant comme contrainte le choix des liens ayant le plus faible coût en énergie (respectant l'equation 1.2) et en prenant en compte le respect de la bande passante théorique de chaque lien.

Cette approche est basée sur l'algorithme BB. Le but de cet algorithme est de trouver dans l'arbre du graphe des solutions une feuille (placement de toutes les IP sur la matrice) ayant un coût en énergie le plus faible possible. Pour remplir cette condition, lors du parcours de l'arbre des possibilités de placement des IP sur la matrice, un algorithme dit "Branch and Bound" est utilisé pour prendre des décisions de routage qui vont permettre de supprimer rapidement l'exploration des cas trop coûteux en énergie. Celui-ci est constitué de deux étapes dites de "branche" et l'autre de "limite". La première étape consiste à parcourir l'arbre de mapping et à rechercher les routeurs de celui-ci dont les IP ne sont que partiellement placées. Un routeur n'ayant pas d'IP connectée est sélectionné et l'IP suivante dans la liste est placée sur un des routeurs non occupés. Ainsi est créé le routeur fils qui va hériter de la table d'allocation de son père et ajouter les chemins de communication de la nouvelle IP qui vient d'être placée. Ensuite, vient la deuxième phase dite de "limite" (bound). Chacun de ces nouveaux routeurs fils est examiné pour voir s'il est possible de générer un meilleur routeur fils plus tard. Une solution de l'arbre représentant le placement des IP sur les routeurs pourra être supprimée sans être complètement développée (placement total des IP) si son coût ou sa limite de coût la plus basse (LBC) est plus grand que la plus petite des grandes limites de coût (UBC) qui a été trouvée. Le coût pour un routeur correspond à l'énergie consommée par les communications, au travers de celui-ci, des IP ayant déjà été placées. Les calculs des UBC et LBC sont détaillés dans [34].

L'UBC d'un nœud est défini comme une valeur qui ne peut pas être inférieure au coût minimum de ses nœuds descendants (nœuds feuilles). Par définition, le coût de n'importe quel nœud fils descendant peut être utilisé comme l'UBC du nœud étudié. Cependant parmi les différents cas de nœuds fils potentiels pour un nœud étudié, il est nécessaire de retenir celui qui a le plus faible coût en énergie parmi ceux-ci. Pour se faire, il faut choisir le nœud fils descendant en utilisant la méthode glouton (GMAP [36]) pour réaliser le mapping des IP non placées sur les routeurs disponibles de la matrice du NoC. Cette méthode consiste à sélectionner l'IP qui requiert le plus de besoins en ressources de communications parmi celles qui ne sont pas placées et de calculer sa position idéale sur la puce.

Le LBC d'un nœud est défini comme étant le coût minimal que ses nœuds descendants pourront probablement atteindre dans le meilleur des cas. En d'autres termes, si un nœud possède un LBC de x, alors chacun de ses nœuds descendants auront au moins un coût de x. Ce coût en communication est décomposé en une somme de 3 coûts. Le premier consiste à calculer le coût entre les IP qui sont placées et qui communiquent entre elles. Le second donne le coût entre les IP placées et celles qui restent à placer. Et enfin le dernier calcule le coût entre les IP non placées.

Dans [35], une notion d'ordonnancement de tâches est introduite. En effet, les ressources de calculs connectées aux routeurs ne sont plus uniquement des blocs IP mais peuvent également être des processseurs sur lesquels on vient ordonnancer une ou plusieurs tâches. De ce fait, lors du parcours du graphe d'application, plusieurs taches peuvent se retrouver placées sur un seul processeur. Ainsi, lorsque l'on regarde les dépendances de données au niveau des communications sur le réseau, cette dépendance de communication inter-tâches dans le même processeur doit être prise en compte lors de l'ordonnancement global de l'application. Ces travaux font suite à ceux réalisés auparavant dans [33, 34] mais avec une prise en compte d'un ordonnancement de taches sur plusieurs éléments de calculs (micro-processeurs, DSP, mémoire système, ...) tout en conservant l'aspect de minimisation des coûts de consommation électrique lors des communications au sein du réseau.

Cette technique est aussi appelée EPAM (**E**nergy/**P**erformance **A**ware **M**apping) et se trouve déclinée sous différentes versions (EPAM XY, EPAM-OE et EPAM-WF) et est présentée plus en détail dans [33].

1.4.3 La technique NMAP

La technique NMAP présentée dans [37, 38] est un algorithme rapide permettant de placer des blocs IP sur une architecture 2D Mesh avec des contraintes de bande passante permettant de minimiser le délai moyen des communications. Cette technique de mapping offre la possibilité de permettre à une ressource d'émettre ses données soit par un seul et unique chemin soit en multi-trajet vers une autre ressource. Dans cet article, une comparaison de performances est faite avec trois autres techniques existantes qui sont le GMAP [34], le PMAP[39] et le PBB[34].

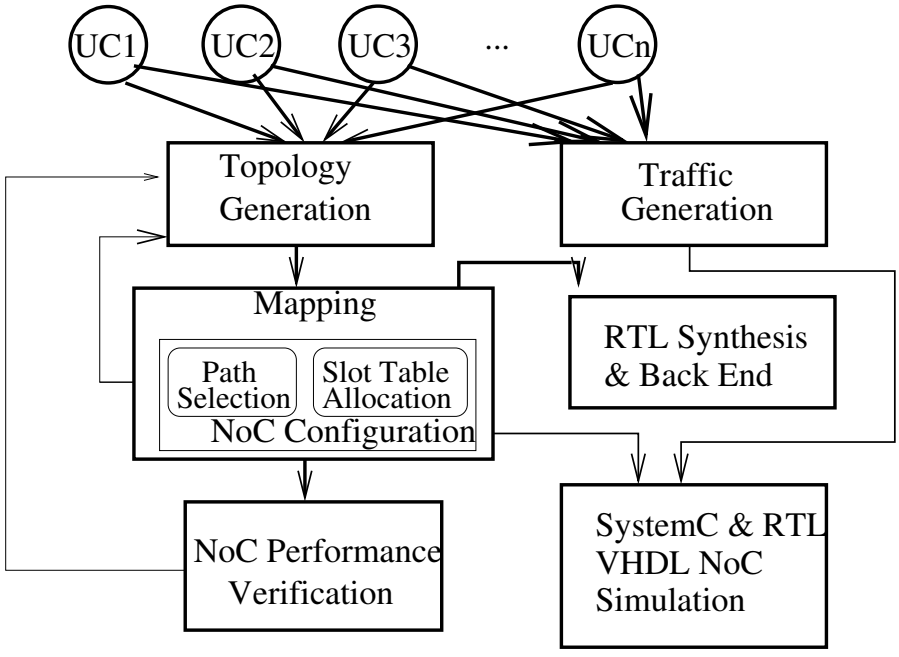

FIG. 1.17 – Flot de conception DVS-DFS

Toutes les simulations NMAP sont réalisées au niveau SystemC en Cycle Accurate utilisant des macros de la xpipeslibrary [40] [41]. Dans [42] est présentée une technique de mapping et de configuration d'un réseau sur puce permettant de garantir des critères de contraintes au niveau du design. Il est également présenté une nouvelle approche qui permet de reconfigurer dynamiquement le réseau au travers des différents cas d'applications (UC) s'effectuant sur le même SoC et aussi la possibilité d'intégrer un Dynamic Voltage and Frequency Scaling(DVS/DFS). Il s'agit d'une méthode permettant de diminuer les besoins en ressources matérielles au niveau réseau en ayant une reconfiguration dynamique du réseau. Le flot de conception DVS-DFS est présenté sur la figure 1.17.

La technique DVS/DFS permet de diminuer la consommation énergétique du réseau selon les applications. Le placement utilisé est le même que celui présenté dans[31] (UMARS) mais avec une amélioration consistant à offrir la possibilité de changer de placement dynamiquement en fonction de l'application exigée. Il s'agit d'un mapping multi-applications. Le dimensionnement de l'application est fournit au flot de conception sous forme de fichier Excel permettant de caractériser l'application en donnant différents paramètres tels que :

- La bande passante requise pour toutes les connexions des ressources composant l'application
- La latence maximale autorisée pour chacune des connexions
- Le niveau de la qualité de service requise pour chaque connexion

Les SoC devenant de plus en plus complexes et surtout multi-standards, ceci a pour effet direct d'engendrer un accroissement en surface de silicium au niveau du SoC. L'amélioration apportée par cette technique est de pouvoir dimensionner un réseau pour pouvoir supporter toutes les applications requises dans le même SoC sans pour autant surdimensionner le réseau avec tous les blocs IP nécessaires.

Dans cet optique, l'approche proposée permet de reconfigurer dynamiquement le réseau pour pouvoir passer d'une application à une autre en utilisant la même surface de silicium. Ceci implique l'utilisation d'une mémoire externe au réseau permettant de charger le contexte de chaque application. Pour la mise en œuvre de ce mapping multi-utilisations, chaque utilisation est étudiée afin de donner une application pire-cas permettant de dimensionner le réseau dans le pire cas afin qu'il puisse garantir le fonctionnement de toutes les applications. Par ce biais, on voit facilement que pour le cas d'un sur-dimensionnement du réseau pour les applications dites lentes (faible bande passante) par rapport à la plus exigeante (grande bande passante) il offre la possibilité de réduire la fréquence et implicitement la tension pour permettre un gain en consommation énergétique.

1.4.4 Compromis entre GT et BE

Dans [32], la technique de mapping employée permet d'intégrer les deux types de services (cf. 1.3.4.1), à savoir le GT et le BE. En effet, de part leur conception, le BE est plutôt orienté pour les réseaux à commutation dit VCT ou "Wormhole", alors que le GT est, quant à lui plutôt adapté pour les réseaux de type "circuit switching" ou "packet switching". Ici l'approche présentée permet d'intégrer les deux types de QoS dans le même routeur. Une couche logique permettant de gérer les deux services est ajoutée et elle permet ainsi par un jeu d'instructions de programme de pouvoir utiliser l'un ou l'autre suivant les contraintes de l'application. Ceci permet d'avoir la meilleure utilisation des liens offerte par le BE

et la possibilité de garantir un trafic par le GT. L'utilisation de ces deux services est complémentaire et permet une utilisation théorique du réseau proche des 100%.

1.4.5 La technique MOCA : mesh based on-chip architecture

Cette technique réalise le placement et le routage des IP sur un NoC en tenant compte des contraintes de bande passante de l'application mais également des contraintes de latence de celle-ci le tout en minimisant la consommation électrique du NoC. Cette méthode est décrite dans [28] et fait une comparaison de performances avec d'autres techniques comme le NMAP[38] et le MILP (Mixed Integrated Linear Programming)[27].

La technique MOCA s'effectue en deux phases. La première phase consiste au placement des IP sur les différents routeurs du Mesh en prenant en compte les critères de contrainte de bande passante et de latence, puis à un découpage bi-partie de la matrice successivement verticalement et horizontalement afin de restreindre les ressources matérielles. Cette étape est réalisée sur un mesh surdimensionné par rapport au nombre de cœurs de l'application pour permettre un placement de l'ensemble des IP plus facile et plus rapide sur celui-ci. Ensuite le découpage de l'arbre des solutions permet de parcourir ce mesh pour éliminer des parties de la matrice ne possédant aucune IP de placée et qui, par conséquent, devient inutile de conserver. A la fin de cette étape, la matrice du réseau possède une structure régulière optimisée pour la contrainte de l'application placée. Dans la deuxième phase, les chemins de communication sont calculés en utilisant un routage XY au plus court prenant en compte la non violation de bande passante. Si la bande passante n'est pas respectée, dans ce cas un routage alternatif n'étant pas forcément au plus court est mis en place pour permettre de solutionner d'éventuels problèmes. La violation de bande passante est un problème récurrent dans les réseaux sur puce. Lorsque les IP sont placées sur la matrice, les chemins de communication entre IP peuvent être communs ce qui engendre un besoin de bande passante sur ces liens équivalent au cumul des besoins de toutes les communications passant par ce même lien. C'est pourquoi il est obligatoire de prendre en compte ce paramètre lors de la phase de routage, car une violation de la bande passante d'un lien va entraîner un non respect des contraintes temps réel de l'application. La solution adoptée par MOCA, dans ce cas précis, est de permettre de réaliser un routage des chemins qui posent problème en ayant une longueur supérieure au chemin idéal le plus court (routage XY). Cette alternative introduit nécessairement un coût supplémentaire en latence sur ce chemin de communication. Par contre, un non respect de cette règle va entraîner nécessairement une congestion du réseau qui, dans ces conditions particulières d'utilisation, devient sujet au deadlock. Cette phase permet de faire un compromis entre une probabilité de deadlock accrue et un coût supérieur de latence sur les communications.

1.4.6 Conclusions sur les techniques de placement et de routage sur les NoC

Pour conclure sur ces différentes techniques de placement routage d'une application sur un NoC, on constate que les critères pris en compte par chaque technique restent les mêmes à savoir les contraintes de bande passante entre IP ainsi que leur latence. Le placement des IP ayant des communications inter-dépendantes doit se faire au plus près, mais ce placement systématique peut s'avérer non optimal lorsque l'on arrive à une violation de bande passante lors de la phase de routage. Il est à noter que ces techniques sont principa-

lement dédiées au réseau de type “Wormhole” “packet switching” qui ne garantissent pas le trafic pour chaque communication. Ceci a pour effet de réaliser un placement routage dans le pire cas d'utilisation pour pouvoir garantir les contraintes temps réel imposées par l'application. Les NoC en GT ne sont pas aussi sensibles lors de leur phase de placement routage car, lorsque les IP sont placées, les tables d'allocation présentes dans les routeurs permettent de garantir des bandes passantes pour chaque communication dans le réseau. Nous allons donc voir maintenant quelques exemples de NoC présents dans le domaine de la recherche académique et industrielle tirant avantage des différentes caractéristiques des NoC énoncées dans les sections précédentes.

1.5 Quelques exemples de NoC universitaires et industriels

Il existe un grand nombre de NoC dans la littérature, et il est impossible de tous les détailler. Cependant, ils évoluent en même temps que nous avons contribué à la mise en œuvre du réseau FAUST dans le contexte du projet 4MORE mais également lorsque nous avons travaillé au développement d'un outil d'aide à la conception pour le NoC FAUST du CEA LETI [43]. Dans cette section, nous décrivons les NoC qui nous semblent les plus représentatifs du domaine de la recherche actuellement.

1.5.1 SPIN

Le réseau SPIN (Scalable Programmable Integrated Network) a été développé au sein du laboratoire LIP6 en 2000 [23, 24]. Le réseau SPIN et son routeur RSPIN ont constitué la première intégration complète d'un réseau sur puce à commutation de paquets. Ce réseau a une topologie originale car il possède une topologie en arbre élargie quaternaire. Ainsi, le routage des paquets dans le réseau peut être dynamique ou statique. Celui-ci est dynamique lorsque l'on parcourt l'arbre dans le sens montant de l'arbre, et il est nécessairement statique dans le sens descendant. Le routage dynamique offre la possibilité d'utiliser des chemins alternatifs pour pouvoir éviter des points de congestion dans le réseau qui entraînent une latence mais également une possibilité de famine. Ce réseau permet de faire transiter les paquets qui composent un message en utilisant différents trajets. Le fait que les paquets d'information puissent arriver dans le désordre entraîne la nécessité de réordonnancer les paquets au niveau de la ressource réceptrice.

Les NI prennent en charge la gestion du contrôle de flux tout au long du chemin de communication lors du passage de crédits d'émission et prennent aussi en charge le réordonnancement des paquets à la réception lorsqu'un routage dynamique a été utilisé. Ce réordonnancement est effectué grâce à un tampon circulaire dont le pointeur d'écriture permet de réorganiser les paquets dans le bon ordre lors de la réception. Le pointeur de lecture s'arrête si le paquet requis n'est pas présent dans la FIFO (First In First Out). Les paquets ne pouvant pas être stockés dans le tampon circulaire par manque de place (pointeur d'écriture et de lecture au même niveau) sont réexpédiés temporairement dans le réseau en attendant que de la place soit faite. Ainsi, on constate rapidement que la taille du tampon circulaire de réception est un point critique, car elle doit être minimisée pour des raisons de surface et de consommation, mais néanmoins suffisante pour éviter les débordements. Le SPIN dispose également de NI compatibles avec le standard VCI [44, 45] pour interfacer les IP avec le réseau, permettant de faire des conversions entre les deux protocoles de communication VCI et SPIN [46].

Le réseau SPIN de part sa conception ne peut pas fournir de garantie en terme de latence et de bande passante, lors de l'acheminement des paquets au sein du réseau. En effet, lorsque les besoins de l'application requièrent des communications à fort besoin en bande passante, le réseau introduit de grandes latences lors des communications [47]. Aucun outil destiné à aider le concepteur lors de la mise en œuvre du réseau n'a été réalisé pour l'instant, mais il existe un modèle systemC du réseau permettant d'effectuer des simulations rapides. Le LIP6 travaille actuellement sur des évolutions de ce réseau, appelées DSPIN et ASPIN. Le DSPIN utilise une topologie non plus en arbre élargi mais une topologie 2D. Le routage est déterminé par la source et utilise un routage XY. Le ASPIN est un DSPIN mais avec des routeurs asynchrones.

1.5.2 FAUST

FAUST (Flexible Architecture of Unified System for Telecommunication) [48, 49, 50, 51] est un réseau développé par le CEA LETI de Grenoble qui a été proposé dans le cadre du projet Européen 4MORE [21] dans lequel nous avons également été impliqués. Ce réseau a été développé pour proposer une nouvelle architecture d'interconnexion dans un SoC de terminaux mobiles de la 4^{e} génération de téléphonie mobile. L'application utilisée sur cette topologie est un modem MC-SS-MA en voie montante pouvant fonctionner aussi bien en MIMO (Multiple Input Multiple Output) qu'en SISO (Single Input Single Output), et un modem MC-CDMA en voie descendante en MIMO ou en SISO. Cette application sera détaillée plus en détail dans le chapitre 2.

Ce réseau a également été expérimenté sur une plate-forme de prototypage dans un contexte multi-composant comportant 4 composants dont 2 ASIC et 2 FPGA. Le réseau est en commutation de paquets avec une gestion de flux de données de type Wormhole, et utilise deux canaux virtuels pour séparer les trafics lorsque cela est nécessaire au niveau de l'application. La connexion des IP sur le réseau se fait au moyen d'une interface réseau (NI) avec des adaptateurs "maison" non normalisés. Cependant, un adaptateur a été réalisé pour se connecter à un bus AHB du processeur ARM qui a pour fonction de paramétrer le réseau pour définir les communications mais également pour faire du monitoring de l'application. Ce NoC existe en version synchrone et asynchrone avec un modèle en SystemC TLM précis au niveau cycle et un modèle VHDL au niveau RTL.

1.5.3 QNoC

Le réseau QNoC est un réseau en topologie 2D qui peut être retaillé en fonction de l'application et qui par conséquent peut posséder des irrégularités. La gestion de flux de donnée employé dans ce réseau est du type Wormhole à commutation par paquets. Le routage au sein du réseau est déterminé par la source et utilise la technique XY améliorée pour permettre le routage dans une topologie 2D irrégulière. Enfin, il peut fonctionner avec des échanges synchrones ou asynchrones [52].

La particularité de ce réseau réside dans la possibilité d'attribuer des niveaux de priorité dans le réseau en fonction du type de service demandé. Ainsi, le trafic de données ou de contrôle est séparé suivant l'une des quatre classes disponibles, chacune d'elle ayant un ordre de priorité plus ou moins important. Chacune de ces classes est répartie sur un canal virtuel différent lors de leur traitement au travers des routeurs du réseau [53, 54, 55]. Ces quatre classes de qualité de service caractérisant le QNoC sont :

- *Signaling* : Ce service couvre les messages à priorité urgente ou les paquets de petite taille pour lesquels on souhaite la priorité la plus haute au sein du réseau et par conséquent assurer la latence la plus faible possible. Ce service convient particulièrement pour des informations de contrôle ou des interruptions.
- *Real-Time* : Ce service permet de garantir des bandes passantes et des latences pour des applications temps réel telles que le traitement de données de type flux audio ou vidéo. Un niveau maximal de bande passante peut être alloué pour chaque lien en qualité "temps réel", celui-ci ne pouvant être dépassé.
- *Read/Write* : Ce service est identique à celui proposé par les topologies bus et est par conséquent conçu pour de courts accès à la mémoire ou des accès à des registres.
- *Block-Transfert* : Ce service est utilisé pour les communications de messages de grande taille ou pour des transferts de gros blocs de données (comme les transferts du type DMA ou les remplissages de cache). Cette classe de trafic a la plus faible priorité.

Un ordonnancement préemptif est présent au sein de ce réseau permettant aux paquets appartenant à un trafic plus prioritaire d'utiliser les ressources en cours d'utilisation par un trafic moins prioritaire. L'ordre de priorité de ces services au sein du NoC est représenté dans le tableau 1.1.

Niveau de service	**Type d'application**	**Priorité**
Signaling	Messages à priorité urgente, interruptions, signaux de contrôle à faible latence	Très Haute
Real-Time	Applications temps réel (vidéo, audio)	haute
Read/Write	Accès courts aux mémoires et aux registres	moyenne
Block-Transfer	Messages longs, transferts de gros blocs de données	basse

TAB. 1.1 – Niveau de priorité des classes de traffic dans le réseau QNoC

1.5.4 Spidergon

La société STMicroelectronics a également développé son propre NoC appelé STNoC utilisant une topologie nommée Spidergon [56, 57]. Cette toplogie Spidergon est une topologie en anneau pour laquelle des liens diagonaux transverses ont été ajoutés réalisant une topologie proche de celle en octogone. Cette topologie Spidergon est présentée sur la figure 1.18.

La gestion de flux de données employée dans ce réseau est du type Wormhole à commutation par paquets. Dans [57], une étude comparative entre une topologie en anneau et une topologie 2D est faite par rapport à celle du Spidergon. Elle a montré qu'elle offre un bon compromis par rapport aux topologies habituelles au niveau des performances.

Depuis le 15 mars 2006, STMicroelectronics a choisi de prendre la société Arteris et son NoC pour réaliser les communications dans les SoC de ses infrastructures sans fils [58].

1.5.5 Arteris

Le NoC Arteris est le premier réseau sur puce commercialisé par la société Arteris créée en 2003, proposant des outils d'exploration et d'implantation [58, 59]. Ce NoC est extrêmement paramétrable et n'offre aucune garantie de service hormis celle proposée par l'outil d'exploration. Cependant, il est possible de donner une priorité plus élevée à une communication (pseudo QoS) au moyen d'une technique de priorité présente au sein des routeurs.

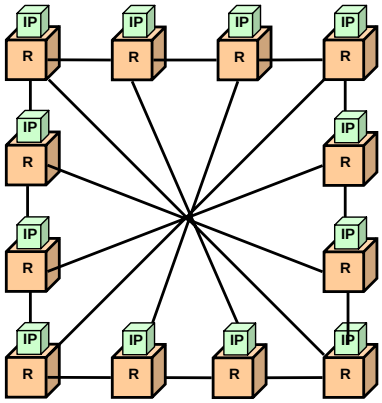

FIG. 1.18 – Topologie Spidergon

Ce NoC offre la possibilité de fonctionner de manière synchrone ou bien en Globalement Asynchrone Localement Synchrone (GALS). Ainsi, la gestion de contrôle de flux est de type Wormhole (meilleur compromis pour minimiser la latence) dans le cas où le réseau est complètement synchrone ou bien de type "Store and forward" lorsque le réseau est en GALS. Le protocole de transport adopté (NTTP : NoC Transaction and Transport Protocol) est propriétaire de Arteris, permettant de faire des transactions en acquittements intégrant les fonctionnalités traditionnelles des bus (adresse, transaction, load/store, burst, bit de parité, ...). Ce type de protocole assure une compatibilité avec un grand nombre d'interfaces standards disponibles dans le monde industriel et universitaire (AMBA AHB, AMBA AXI, OCP 2.0). Arteris offre également dans ses outils une librairie de composants nommée Danube qui contient des IP configurables et interfaçables avec les standards proposés par son NoC. Comme nous l'avons dit précédemment, afin d'aider le concepteur dans les étapes d'exploration et d'implantation, il existe deux outils complémentaires qui sont NoCExplorer et NoCCompiler servant respectivement à la phase d'exploration de l'application et à la phase d'implantation matérielle.

NoCExplorer est un environnement qui prend en compte les besoins en bande passante des IP pour analyser les différentes topologies qui peuvent satisfaire les contraintes de l'application. L'outil NoCCompiler utilise les informations fournit par NoCExplorer (Topologie, placement, ...) pour créer une base de donnée qui décrit le NoC. Ce logiciel permet

de générer un modèle systemC simulable précis au niveau cycle (RTL), ou des descriptions en VHDL ou Verilog. La force d'Arteris est de proposer un produit adaptable et qui est entièrement compatible avec les outils de conception disponible sur le marché actuel (la cible peut être du FPGA ou de l'ASIC). L'engouement que Arteris rencontre avec bon nombre de leaders des outils de conception en électronique (CoWare) montre l'intérêt que suscitent les NoC et prouvent qu'ils offrent une bonne alternative aux problèmes de limitations des bus rencontrés actuellement.

1.5.6 ×PIPES

Le réseau ×PIPES [41] est un réseau de type Wormhole à commutation de paquet ayant un routage des communications statique à travers le réseau, les chemins de communication étant stockés dans les NI des blocs IP connectés aux routeurs. Ce réseau peut être généré avec un modèle systemC simulable précis au niveau cycle (RTL). La modélisation de l'application se fait au moyen d'un graphe de tâches dans lequel est mentionné les besoins en bande passante entre toutes les tâches qui doivent communiquer entre-elles. Une simulation initiale consistant à parcourir le graphe de description de l'application permet d'obtenir les caractéristiques de trafic de l'application. C'est durant cette phase que la notion de qualité de service est prise en compte [60]. Chaque tâche est assignée à une IP et les bandes passantes sont alors utilisées par l'outil SUNMAP pour générer une topologie pour le NoC [38].

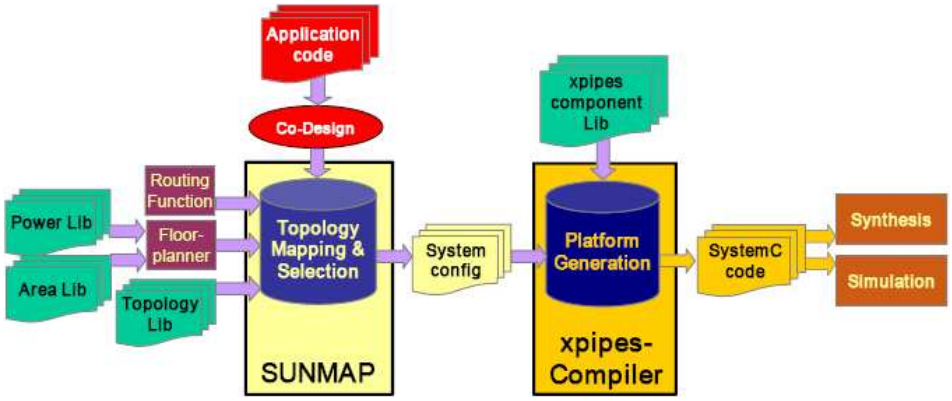

FIG. 1.19 – Flot de conception du NoC ×PIPEs

L'outil SUNMAP a pour objectif de réaliser l'implémentation de l'application sur le réseau en prenant en compte les délais moyens des communications, la surface des IP, la consommation électrique qui est étroitement liée aux besoins en bande passante et le placement des IP sur la matrice. Ainsi, l'outil va tester plusieurs topologies en analysant le coût en surface et la consommation du NoC. Les choix matériels sont fait de manière itérative et les liens et ports de routeurs non utilisés sont supprimés. Durant cette phase, une estimation est faite pour dimensionner au mieux les FIFO des NI de chaque IP.

Dès lors que cette phase de dimensionnement matériel est effectuée, les données provenant de cette étape sont fournies à l'outil xpipesCompiler [40] qui, à partir de la bibliothèque de composants et des paramètres choisis, va réaliser la génération du code

SystemC pour la simulation, ainsi que le code pour la synthèse. Le flot de conception du réseau Sunmap est décrit dans la figure 1.19.

1.5.7 Ætheral

Le réseau Ætheral est un NoC développé par le groupe de recherche de la société Philips. Comme le xPIPE, celui-ci est de type Wormhole à commutation de paquets ayant une technique de routage de type street-sign, le routage des communications étant réalisé par la source, c'est à dire le NI encapsulant les blocs IP connectés aux routeurs. Ce réseau utilise une horloge unique qui est donc commune à l'ensemble des éléments qui composent le réseau (IP, routeur, NI). Celui-ci intègre deux qualités de service qui sont le BE et le GT [61, 62, 63]. Ces deux qualités de service sont implantées sur deux canaux virtuels au sein des routeurs l'un étant en BE, l'autre étant en GT.

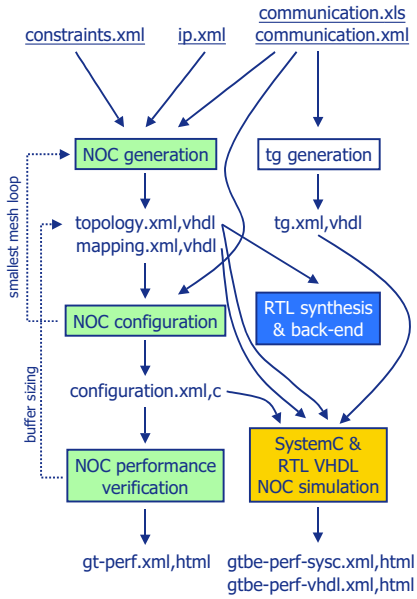

FIG. 1.20 – Flot de conception du NoC Æthereal

Comme nous l'avons vu précédemment (section 1.3.4.1), afin d'avoir une qualité de service de type trafic garanti, l'utilisation de tables d'allocation (TDMA) est nécessaire. Dans un premier temps, ces tables d'allocation avaient été mises dans les routeurs du réseau mais il s'est avéré que la complexité du routeur devenait trop grande et le coût en surface trop important. C'est pourquoi, celles-ci ont été intégrées directement dans les NI. Lorsque les deux types de QoS cohabitent au sein du réseau, le BE qui est moins prioritaire que le GT utilise par conséquent le reste de la bande passante. Les interfaces réseau permettent également de réaliser des adaptations de protocoles standards qui sont AXI [64] et DTL [65]. De plus, comme nous l'avons vu dans la section 1.4, les NI permettent de réaliser des émissions de paquets en multi-chemin pour émettre vers des destinataires multiples ou bien vers un seul destinataire. Cette dernière possibilité permet de soulager des liens de communication arrivant en limite de violation de bande passante. Cette caractéristique

peut permettre à un outil de conception de trouver des solutions de routage sans avoir besoin de trouver une nouvelle solution de mapping lorsqu'une violation de bande passante est rencontrée et qu'aucune solution n'est envisageable. Le flot de conception de la méthode UMARS (Unified MApping, Routing and Slot allocation) prend notamment en compte cette possibilité d'émettre en mode multi-chemin. Cette méthode est présentée sur la figure 1.20.

Récemment, le groupe de recherche de Philips a développé des méthodes et outils pour automatiser les étapes de son flot de conception [66]. Il offre ainsi un flot prenant comme contrainte la bande passante et la latence des communications et génère le placement des IP autour du NoC, le routage des chemins et l'affectation des slots de temps, ainsi que le dimensionnement des tailles des FIFO. A la fin de ce flot, le code VHDL synthétisable du NoC ainsi que les fichiers de configuration associés sont générés. Ce flot de conception amélioré est présenté dans [31, 67]. L'algorithme UMARS permet de prendre en compte conjointement les objectifs des phases de placement des IP (mapping), de sélection du chemin et d'allocation des slots.

1.5.8 μspider

Le réseau μspider [68, 69] [70] est un NoC développé en collaboration entre l'IETR [71] et le laboratoire universitaire LESTER [72] à Lorient. Il s'agit d'un réseau sur puce de type Wormhole à commutation de paquets basé sur un mécanisme de crédits permettant d'éviter la perte d'information lors de la réception des messages au niveau des NI des récepteurs (places disponibles dans les FIFO). Ce réseau est capable d'assurer trois types de qualité de service que sont le GT, le BE prioritaire et le BE. Chacune de ces trois QoS ont un ordre de priorité différent qui est mentionné dans le tableau 1.2.

Trafic	**Priorité**	**Classe**	**Réservation**
Temps-réel	1	GT	Réservation cohérente des slots consécutifs sur l'ensemble de son chemin
Message court et urgent	2	BE prioritaire	un slot de libre minimum sur chaque lien de son chemin
Message sans garantie	3	BE	pas de réservation

Tab. 1.2 – Répartition des trafics sur les canaux virtuels

Ainsi, pour la qualité de service de type GT, celle-ci est effectuée au moyen de l'utilisation de la technique de réservation de bande passante TDMA basée sur les mêmes principes que celle utilisée dans le réseau Ætheral (section 1.5.7). Les politiques de routage supportées sont le routage XY, et le codage street-sign, le premier étant adapté aux topologies grille 2D alors que le second est plutôt adapté aux topologies irrégulières. Le nombre de ports disponibles sur les routeurs est paramétrable mais celui-ci doit en disposer d'au minimum 3, dont 1 servant de connexion vers l'IP connecté au routeur. Il existe 4 politiques d'arbitrage différentes pour chaque canal virtuel au sein des routeurs du réseau. Il a également été intégré un protocole standard de communication au sein des NI qui est un wrapper OPB. Celui-ci permet d'être compatible avec le bus CoreConnect de IBM et

le processeur Microblaze de Xilinx. Ainsi, les IP aux standards OPB peuvent être connectées facilement au NoC. Ces wrappers permettent également de connecter plusieurs IP qui s'apparentent à un bus possédant plusieurs IP et processeurs connectés et réalisant des échanges de données à travers le réseau vers d'autres IP.

1.5.9 Comparatif des différents réseaux

Nous avons vu quelques exemples de NoC disponibles dans le domaine de la recherche industrielle et académique. Nous présentons ci-après un tableau récapitulatif (Tableau 1.3) permettant de faire un comparatif entre les topologies, les modes de gestion de flux et les qualités de service offerts par les principaux réseaux sur puce disponibles.

Nom du réseau	**Topologie**	**Gestion de flux**	**Qualité de service**
SPIN	arbre élargi et Grille 2D	Wormhole	BE
aSoC	Grille 2D	Wormhole	BE
Spidergon	Octogone	Wormhole	BE
Proteo	Anneau	Wormhole	BE
Ætheral	Grille 2D	Wormhole	BE et GT
×PIPES	Grille 2D adaptable	Wormhole	GT
Hermes	Grille 2D	Wormhole	BE
QNoC	Grille 2D adaptable et irrégulière	Wormhole	BE avec niveau de priorité
Arteris	Grille 2D	Wormhole (synchrone) ou SF (asynchrone)	BE
FAUST	Grille 2D	Wormhole	BE
μspider	Grille 2D	Wormhole	GT, BE, BE prioritaire

Tab. 1.3 – Tableau comparatif des réseaux

Ce tableau comparatif met en évidence le fait que la topologie 2D est la plus employée. C'est une topologie assez simple de mise œuvre et qui facilite les outils de placement et de routage des blocs fonctionnels de l'application sur les routeurs du réseau car il s'agit de règles mathématiques plutôt simples de mise en œuvre. De plus, le mécanisme de gestion de flux de données Wormhole est le plus utilisé car il a un coup en ressource matérielle faible et offre une latence la plus faible dans le cas d'une qualité de service de type BE. Enfin, la qualité de service offerte par tous les réseaux est de type BE. Mais du fait de sa conception, on ne peut garantir une bande passante ni une latence pour les communications, ceci ayant pour effet d'avoir recours à des simulations permettant de vérifier si dans le pire cas les contraintes temps réels de l'application peuvent être garanties ou non. C'est pourquoi on voit de plus en plus de réseaux qui font apparaître soit des niveaux de priorité dans le BE, soit la possibilité d'avoir du BE et du GT qui cohabitent.

1.6 Conclusion

Dans cet état de l'art sur les réseaux sur puce, nous avons abordé les raisons pour lesquelles les bus trouvent leurs limites dans les systèmes sur puce actuels. Ces limitations étant liées au faible parallélisme des communications mais également à une limitation en terme de bande passante pour les applications. C'est pourquoi les NoC apparaissent naturellement pour proposer des solutions à ces critères limitatifs des topologies bus.

En effet, les NoC offrent de grandes perspectives de part leur structure mais également par la possibilité de proposer une qualité de service pour les communications de l'application. Ces réseaux peuvent avoir des topologies différentes mais également des gestions de flux de données différentes influant directement sur la complexité matérielle du NoC mais également sur la latence du transport des données. Ces NoC sont donc extrêmement paramétrables. Ainsi, de nombreux paramètres sont à prendre en compte lors de leur conception, car chacun d'entre eux peuvent influencer directement ou indirectement les performances globales de l'application. Voici ci-après, une liste non exhaustive de ces paramètres qui peuvent être pris en compte dans la phase de conception :

- Topologie du réseau
- Placement des blocs IP sur la structure
- Routage des données
- Qualité de service
- Tailles des FIFO des interfaces réseau
- Gestion de flux de données dans le réseau
- Fréquence de fonctionnement
- Norme d'encapsulation des blocs IP
- Coût en consommation
- Coût en surface de silicium des IP
- ...

Tous ces paramètres engendrent nécessairement une complexité de mise en œuvre accrue en comparaison avec les SoC en topologie bus. C'est dans cet objectif que de nombreux travaux de recherche sont en cours afin d'offrir au concepteur des outils d'aide à la conception dédiés aux NoC. Ceci dans le but de réaliser rapidement un SoC à base de NoC et d'optimiser le NoC pour l'application à placer, en prenant en compte un ou plusieurs paramètres énoncés précédemment.

Nous avons vu qu'il existe plusieurs méthodes de placement routage pour les réseaux sur puce, chacune ayant leurs avantages et leurs inconvénients. Ces méthodes prennent en général en compte les mêmes critères de base mais se différencient dans la façon d'aborder l'espace d'exploration des solutions en prenant en compte d'autres critères propres. Ainsi, certaines méthodes de placement routage vont privilégier la bande passante entre blocs IP, le placement des IP sur la structure, les chemins de routage des communications (simple trajet ou multi-trajet), la latence, la minimisation de la consommation énergétique, ... Ces critères permettent d'affiner la création du NoC dans une orientation précise (performances, minimisation du coût en surface, minimisation de la consommation électrique, respect des contraintes temps réel, ...) mais tout en conservant l'objectif principal qui est de respecter et garantir les contraintes temps réel de l'application.

Ces méthodes de placement routage se distinguent également selon le type de qualité de service que le réseau va proposer. En effet, nous avons vu qu'il existait deux types de

qualité de service pour les réseaux sur puce qui sont le GT et le BE. C'est précisément dans le cas d'un GT que la méthode de placement routage est un peu plus complexe car un pré-ordonnancement des communications entre tâches doit être fait afin de réserver des tables d'allocation des communications pour garantir des bandes passantes pour les communications de l'application et par conséquent respecter les contraintes temps réel de l'application.

Pour finir, nous avons vu qu'il existe un grand nombre de NoC dans le monde de la recherche universitaire mais également dans le monde de la recherche industrielle ce qui montre bien l'intérêt grandissant des NoC pour les futurs SoC. Dans la suite de ce manuscrit, nous allons centrer nos études sur le réseau sur puce FAUST du CEA LETI. Nous avons contribué aux explorations architecturales prévu par le Work Package 4 (WP4) du projet Européen 4MORE. Ce projet visait à proposer des solutions algorithmiques et matérielles pour les terminaux mobiles de 4^{e} génération (le contexte de l'application sera détaillée dans le chapitre 2). Nous avons contribué aux explorations architecturales de l'application sur le NoC FAUST mais également à son dimensionnement et à sa validation en terme de respect des contraintes temps réel de l'application. Cette contribution s'est également faite sur le choix du placement des IP sur la topologie grille 2D et le choix des chemins de routage des communications. Ce NoC ayant des caractéristiques communes aux autres NoC existants, nous avons choisi de conserver celui-ci pour proposer un outil d'aide à la conception se basant sur des simulations en SystemC précis au niveau cycle.

Chapitre 2

Contexte de l'application 4G dans le cadre du projet Européen 4MORE

Sommaire

Tandis que le système mobile terrestre de troisième génération (UMTS-UTRA) est actuellement déployé, il existe une activité de recherche significative pour les futurs systèmes qui constitueront la 4^{e} génération de téléphonie mobile (4G). La vision européenne pour cette nouvelle génération repose sur un système intégré entièrement basé sur des SoC à base d'IP offrant tous les services. Afin de s'adapter aux futurs services qui exigeront des ressources de plus en plus importantes notamment en terme de bande passante, un composant à large bande remplissant ces critères est nécessaire. En effet, les contraintes de la 4G devront garantir un débit d'information binaire maximum compris entre 2 et 20Mbps dans un environnement véhiculaire et probablement entre 50 et 100Mbps en environnements piétonniers ou intérieur, le tout en utilisant une largeur de bande de 50 à 100MHz.

Une des technologies les plus prometteuses pour ce type de composant à large bande est l'Accès Multiple par division de codes en Multi-Porteuses (MC-CDMA), qui combine les mérites de l'OFDM avec ceux des techniques d'étalement de spectre. Il y a eu ces dernières années un engouement très fort pour cette technique, notamment au Japon où des essais pratiques ont été effectués. Au niveau Européen et dans le programme d'IST (Information Society Technology), le projet MATRICE [73] a entrepris la recherche et la validation des concepts d'accès et de transmission basés sur la technologie de MC-CDMA

pour la fourniture de composants à large bande des futurs systèmes cellulaires mobiles. L'objectif du projet 4MORE est de compléter le projet MATRICE et ses résultats obtenus afin de réaliser la conception d'un SoC pour un terminal mobile de 4^{e} Génération utilisant des techniques basées sur du MC-CDMA avec des antennes multiples.

Nous allons donc voir dans un premier temps les objectifs du projet Européen 4MORE afin de proposer des solutions qui s'inscrivent dans les contraintes des critères de définition de la 4G énoncés ci-dessus. Ensuite, nous décrirons la chaîne algorithmique de manière globale, et nous décrirons de manière détaillée les voies montante et descendante avec leur modèle associé. Un schéma de modélisation des blocs fonctionnels composant les algorithmes est proposé afin de modéliser cette application. Ces paramètres de modélisation seront ensuite intégrés dans le modèle SystemC du NoC FAUST pour explorer l'espace des solutions architecturales pour pouvoir garantir le respect des contraintes temps réel de l'application. Celles-ci sont précisées dans la section ci-après.

2.1 Description des objectifs du projet Européen

2.1.1 Spécifications techniques

Comme nous l'avons vu précédemment, le projet Européen 4MORE [21] vise à définir les futurs algorithmes de la 4^{e} génération de téléphonie cellulaire sans fil qui permettront de remplir les critères de contrainte en terme de débit d'information. Ce débit est fortement accru pour offrir des services de plus grande qualité pour ces futurs terminaux sans fil (notamment pour les applications vidéo ou audio où le débit d'information devient de plus en plus important). Ainsi, dans le cadre du projet 4MORE, les contraintes de débit d'information fixées dans le cahier des charges sont de 10Mbps dans un environnement véhiculaire et 100Mbps en environnement piétonnier ou fixe. Le cahier des charges en terme de contraintes de débit d'information pour le terminal mobile est récapitulé dans le tableau 2.1.

Bande passante	**Vitesse**
100Mb/s	3 Km/h
10Mb/s	300Km/h

Tab. 2.1 – Spécification du projet Européen 4MORE

Afin de tenir ces contraintes de débit, des choix algorithmiques et techniques ont été fait. Pour cette transmission à porteuses multiples, il est utilisé 695 sous-porteuses dont chacune se voit affecter une fonction particulière :

- 672 porteuses dédiées pour les données des utilisateurs
- 22 porteuses dédiées pour les pilotes
- 1 porteuse à zéro pour la synchronisation

Ainsi, lors de l'envoi ou la réception des données utilisateur, la couche physique va répartir celles-ci sur les 672 porteuses de données réservées. Afin de pouvoir communiquer en voie montante et en voie descendante, la trame de transmission a été découpée en slots qui, dans un cas, sont dédiés aux communications montantes (du terminal mobile vers la station de

base) et dans l'autre cas, sont dédiées aux communications descendantes (station de base vers le terminal mobile). La trame de communication est représentée sur la figure 2.1.

FIG. 2.1 – Structure de la trame de transmission 4MORE

Chacun de ces slots TX et RX sont eux-mêmes découpés en 32 symboles OFDM. Ces symboles vont permettre de disposer sur les sous-porteuses de données, les données utilisateurs à transmettre dans le cas du TX, et les données utilisateurs à recevoir dans le cas du RX. Parmi les 32 symboles OFDM de ces slots TX et RX, certains d'entre eux sont réservés pour les symboles pilotes, les symboles de synchronisation et les symboles à zéro. Les symboles pilotes servent à réaliser une estimation de canal lors de la réception des données, soit au niveau de la station de base, soit au niveau du terminal mobile. Ces symboles pilote sont au nombre de 6. Le symbole de synchronisation sert à synchroniser le slot complet lors de son décodage pour permettre d'ajuster temporellement les données par rapport à un éventuel retard introduit par le canal de transmission. Ce symbole sert à la démodulation OFDM afin de "caler" la FFT sur le symbole et éviter un chevauchement entre deux symboles lors du calcul (évite un décodage erroné ou une perte d'informations). Enfin, le symbole à zéro sert quant à lui à éviter les recouvrements entre deux types de slots, basculement du mode émission vers le mode réception (chevauchement temporel des données de la voie montante et descendante). Pour chaque slot, l'utilisateur émet ses données sur les 24 symboles OFDM de données qui sont disponibles, les autres servant à la synchronisation et à l'égalisation du slot. La structure d'un symbole OFDM d'une trame définie dans le projet 4MORE est représentée sur la figure 2.2.

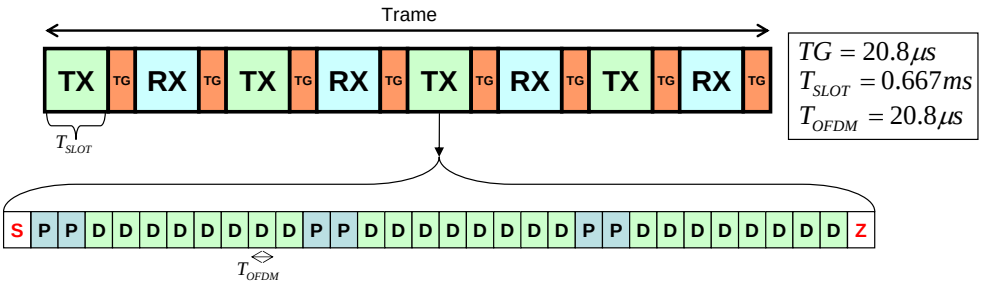

FIG. 2.2 – Structure symbole OFDM

La définition des paramètres caractérisant la structure de la trame de communication s'est bien sûr basée sur les contraintes du canal de propagation en prenant en compte les environnements "*indoor*"et "*outdoor*" (Canal BRAN-A et BRAN-E). Ces paramètres ont été choisis en fonction du temps et de la bande de cohérence du canal pour remplir les contraintes de la 4G des futurs systèmes cellulaires sans fil.

Ainsi, l'objectif premier du dimensionnement du système était de s'adapter au canal de propagation, mais d'autres critères ont également été pris en compte, notamment la possibilité de réutiliser les techniques et blocs matériels provenant de la 3G (UMTS). Ce système étant basé sur une technique OFDM, celui-ci s'est vu constitué d'éléments ou de techniques déjà existantes dans d'autres normes comme l'HIPERLAN/2 ou les propositions de NTT DoCoMo. Enfin, la faisabilité de la réalisation matérielle n'a pas été négligée afin de définir un système réaliste, capable d'être implanté dans un démonstrateur matériel. Les principaux paramètres qui définissent la structure de la trame de communication du projet sont listés dans le tableau 2.2.

Fréquence porteuse	5 GHz
Étalement des retards maximum	4 μs
Fréquence d'échantillonnage	61.44 MHz
Bande occupée	41.46 MHz
Taille de la FFT	1024
Nombre de sous-porteuses modulées	695
Durée de l'intervalle de garde	>6.66 μs
Nombre d'échantillons de l'intervalle de garde	256
Durée totale d'un symbole OFDM	20.8 μs
Nombre d'échantillons d'un symbole OFDM T_{OFDM}	1280
Durée d'un slot	0.666 ms
Durée d'un slot (en nombre de symboles OFDM)	32
Nombre de symboles pilotes dans un slot	6
Nombre de symboles de synchronisation dans un slot	1
Nombre de symboles de garde dans un slot	1
Codes d'étalement	Walsh-Hadamard
Longueur des codes d'étalement S_f	8-32
Nombre maximum d'utilisateurs	23
Nombre d'antennes au niveau du terminal mobile	2
Nombre d'antennes au niveau de la station de base	4
Espacement entre antennes au niveau du terminal mobile	1 λ
Espacement entre antennes au niveau de la station de base	10 λ

TAB. 2.2 – Principaux paramètres du système

Ainsi, les contraintes temps réel de la trame à respecter sont celles de la cadence des symboles OFDM T_{OFDM} qui est de 20.8μs. Cette contrainte temporelle va nous servir de référence lors de nos phases d'exploration architecturale au niveau NoC afin de connaître le respect ou non des contraintes temps réel de l'application.

2.1.2 Spécification algorithmique de la voie montante

En ce qui concerne la voie montante, la technique de communication employée est de type SS-MC-MA (Spread Spectrum Multi Carier Multiple Access). La particularité de celle-ci est de permettre à l'utilisateur d'avoir des blocs de sous-porteuses dédiés et réservés pour ses communications. Il a été définit dans le projet une restriction à 24 sous-porteuses

par utilisateur pour la voie montante avec un facteur d'étalement $S_f = 8$. Ceci offre donc 3 blocs de 8 sous-porteuses pour pouvoir émettre des données. Dans ce cas précis, tous les codes d'étalement alloués pour ces sous-porteuses sont tous attribués à un seul et même utilisateur. Ainsi, pour chaque symbole OFDM, seuls 24 des 672 complexes de données qui composent le symbole seront utilisés. Les autres symboles seront utilisés par les autres terminaux mobiles qui seront dans la même cellule autour de la station de base. Bien entendu il est impossible qu'un utilisateur puisse émettre sur les sous-porteuses de données d'un autre utilisateur, notamment grâce à l'attribution des codes d'étalement. Nous verrons par la suite que le debit d'information binaire B_{info} peut être accru suivant le rendement du codage canal choisi ou encore le type de Codage Binaire à Symbole (CBS) employé. Nous verrons en détail les caractéristiques des éléments de la voie montante dans la section 2.2.3.

2.1.3 Spécification algorithmique de la voie descendante

Pour la voie descendante, la technique de communication employée est de type MC-CDMA (Multi-Carier Code Division Multiple Access). Pour chaque symbole OFDM de données, l'utilisateur retrouve ses données au moyen du code qui lui a été attribué. Ainsi, la capacité de débit d'information est fortement accrue par rapport à la voie montante car l'utilisateur peut voir ses données étalées sur l'ensemble des sous-porteuses. Le facteur d'étalement pour la voie descendante peut être de $S_f = 8$ ou $S_f = 32$, influant directement le débit d'information binaire B_{info}. Dans le cas d'un $S_f = 32$ le débit d'information sera moindre que pour un $S_f = 8$ car le nombre de symboles de données complexes par utilisateur sera de 84 contre 21. Nous verrons en détail les caractéristiques des éléments de la voie montante dans la section 2.2.4 avec notamment l'impact de certains blocs sur le débit d'information utile pour l'utilisateur.

2.2 Description de la chaîne algorithmique

Dans cette section, nous allons voir les différents éléments qui composent la chaîne algorithmique des voies montante et descendante. En effet, pour pouvoir explorer l'espace des solutions architecturales en vue d'une implantation matérielle sur la structure NoC FAUST, nous devons représenter et modéliser chaque élément fonctionnel de la chaîne algorithmique selon un modèle simple. C'est dans cet objectif que nous avons définit une modélisation simple de chaque bloc fonctionnel pour pouvoir transcrire les caractéristiques de la chaîne au modèle SystemC du NoC FAUST. En effet, ce NoC est disponible en SystemC TLM permettant de réaliser des simulations qui vont nous permettre de valider le placement et le routage des blocs fonctionnels sur la structure NoC. L'intérêt de modéliser de manière simple chaque bloc fonctionnel permet de s'affranchir de la nécessité d'avoir le bloc algorithmique réel, et par conséquent d'éventuels retards de livraison des blocs IP. Nous verrons donc dans un premier temps la vue d'ensemble de la chaîne algorithmique. Ensuite, nous verrons plus en détails chacun des éléments fonctionnels de la voie montante puis ceux de la voie descendante.

2.2.1 Présentation de la chaîne

Comme nous l'avons vu précédemment, le système cellulaire prévu dans le projet 4MORE est composé d'une voie montante et d'une voie descendante. Ces deux voies sont complètement distinctes l'une de l'autre. Il faut également noter que compte tenu de la trame présentée ci-dessus, chacune fonctionne de manière exclusive, la chaîne TX ne pouvant émettre lorsque la chaîne RX reçoit des données et vice et versa. Ceci est imposé par la trame de communication mais également par une restriction physique au niveau du terminal qui ne possède qu'une seule antenne qui sert à la réception mais également à l'émission. Chacun des blocs fonctionnels qui composent la voie montante et la voie descendante réalise une fonction particulière dans le flot de traitement de l'information binaire ou symbole. Certains d'entre eux sont paramétrables en fonction des qualités de service demandées par l'utilisateur mais également en fonction de la qualité du canal (environnement véhiculaire ou piétonnier) lors de la transmission ou de la réception des données. Nous verrons dans les sections 2.2.3 et 2.2.4 les blocs permettant de faire varier le débit d'information binaire. L'ensemble des blocs IP composant les voies montante et descendante ainsi que leur ordre d'apparition sont décrits sur la figure 2.3.

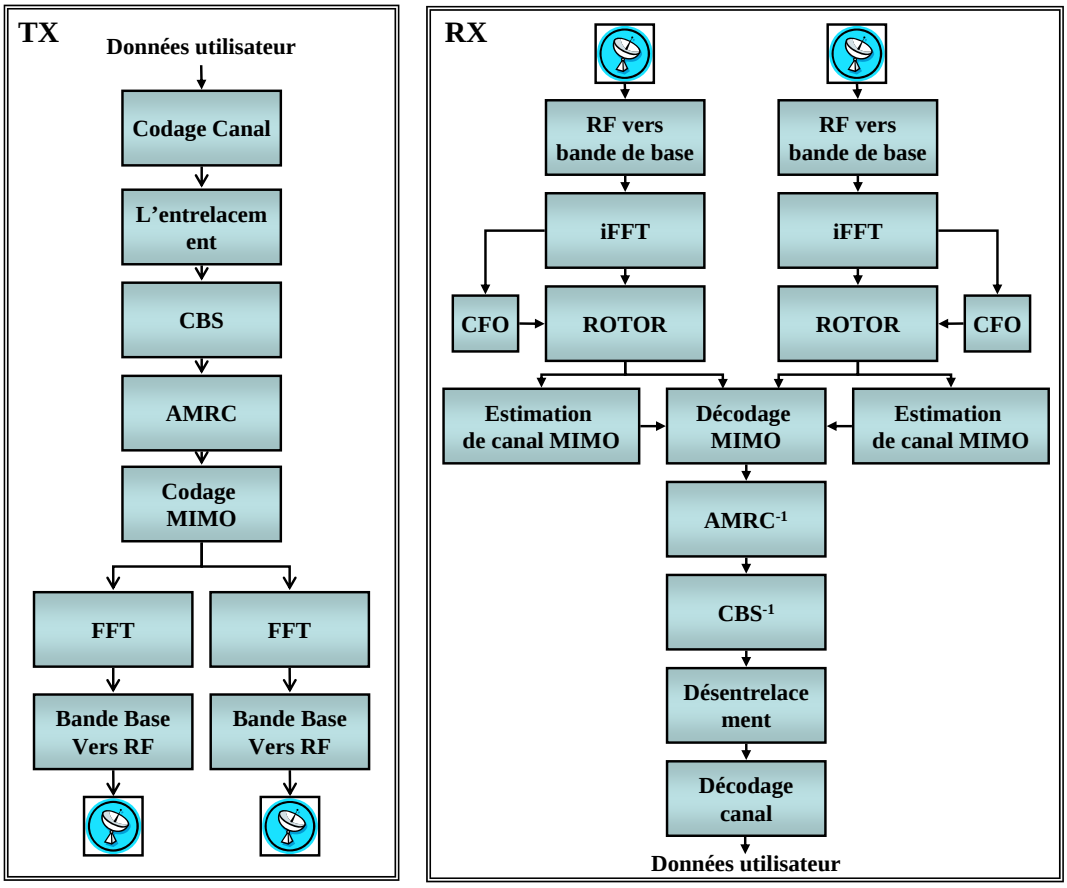

FIG. 2.3 – Vue globale de la chaîne algorithmique

Comme énoncé précédemment, nous avons besoin de modéliser les blocs IP composant les chaînes TX et RX. Nous allons donc voir dans la section suivante, le modèle de représentation de ces blocs IP.

2.2.2 Schéma de modélisation des blocs fonctionnels (IP)

Compte-tenu des caractéristiques de cette application et des contraintes d'exploration architecturale au niveau NoC, nous avons choisi de définir un modèle simple de chaque bloc fonctionnel. Afin de pouvoir intégrer le modèle comportemental des blocs IP dans le flot de simulation SystemC [74] TLM du NoC FAUST, nous avons défini un format de description simplifié qui peut s'appliquer à n'importe quel bloc fonctionnel d'une application dont on cible son implantation sur une structure de communication de type NoC. Pour cela, nous modélisons les flux de données en entrée et en sortie des blocs IP et le temps de latence au sein de ce bloc. Ainsi, chaque bloc fonctionnel est décrit selon trois blocs qui sont :

- la taille des données en entrée : quantité de données nécessaire en entrée pour faire un traitement
- le temps de traitement : nombre de cycles d'horloge nécessaire pour réaliser le traitement des données
- la taille des données en sortie : quantité de données produite après la phase de traitement

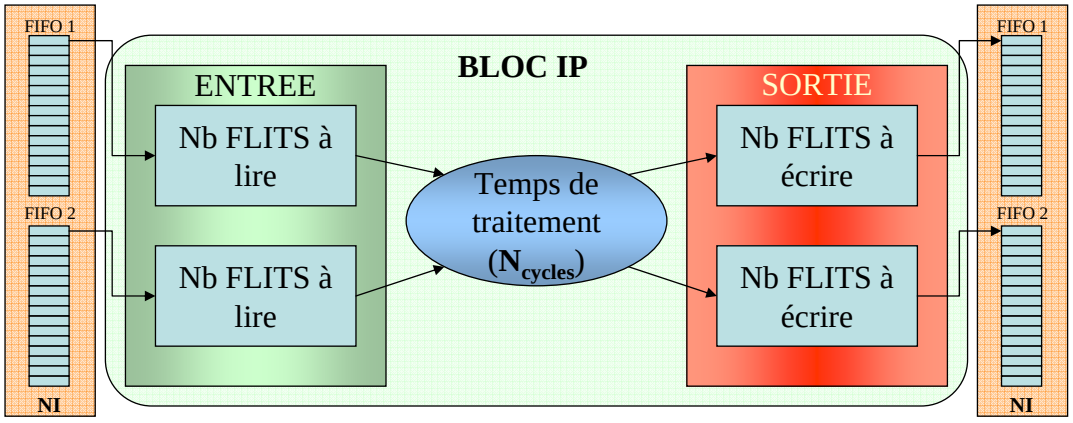

FIG. 2.4 – Schéma de modélisation bloc IP pour simulation sur modèle SystemC TLM de NoC

Ce modèle convient dans la plupart des applications classiques (flot de données). Il permet de caractériser l'information binaire nécessaire avant que le traitement ne s'effectue. Puis, la phase de traitement représentée par un temps d'attente spécifié en nombre de cycles représente le temps de traitement réel des données (nombre d'itérations pour réaliser une FFT par exemple). La spécification au niveau cycle permet de s'affranchir de la technologie utilisée lors de l'implantation matérielle, celle-ci ayant un impact non nul sur la fréquence de fonctionnement du SoC. Enfin, nous caractérisons la taille des données produites après traitement. Ces informations simples permettent de caractériser réellement les blocs IP d'une chaîne de traitement sans avoir les "vrais" blocs IP (VHDL ou Verilog pour une

implantation matérielle) ou les "vrais" algorithmes (C ou C++ dans le cas d'une simulation algorithmique logicielle). Cette modélisation trouve son intérêt lorsque l'ensemble des blocs IP d'une chaîne de traitement ne sont pas tous disponibles ou réalisés à un instant t. Mais également dans le cas où le SoC à réaliser est complexe et que celui-ci demande une ou plusieurs phases d'AAA permettant de guider le concepteur dans ses choix de conception. Ce formalisme de représentation des tâches de l'application est couramment employé dans les outils d'AAA (Syndex [75]) que l'on peut trouver dans les milieux de la recherche académique ou industrielle. Ce formalisme est également souvent employé dans les travaux de la communauté de recherche sur les NoC.

Les dépendances de données entre ces différents blocs fonctionnels sont renseignées par le diagramme de description de l'application comme le montre la figure 2.3. Nous allons maintenant décrire brièvement chaque bloc fonctionnel de la voie montante et de la voie descendante de l'application 4MORE et donner leur modèle simplifié associé que nous utiliserons par la suite pour nos simulations et explorations architecturales au niveau NoC.

2.2.3 La voie montante

Comme nous l'avons précisé précédemment (2.1.2) la technique employée ici est de type SS-MC-MA. Nous allons donc voir chacun des blocs fonctionnels qui constitue la chaîne algorithmique de la voie montante et leur modèle simplifié associé qui va nous servir à modéliser le graphe d'application utilisé pour nos explorations de choix architecturaux. Il est à noter que certains blocs de la chaîne sont paramétrables, ce qui a pour effet d'augmenter ou diminuer les débits d'information binaires entre blocs. C'est pour cette raison que nous avons choisi de modéliser ces blocs dans les conditions pire cas. Ceci sera justifié par la suite lors de la phase d'exploration architecturale au niveau NoC où les contraintes temps réel de l'application sont directement liées aux besoins en bande passante entre les tâches de l'application. Tous les modèles simplifiés des blocs présentés ci-après sont exprimés dans la plus petite unité de représentation des données au sein d'un NoC, le FLIT (1 FLIT = 32 bits).

2.2.3.1 Le codage canal

Il s'agit d'un codage canal qui permet d'augmenter la robustesse de la transmission en insérant dans le flot de bits d'information utile B_{info} de l'utilisateur des bits de parité. Ces bits de parité permettent de corriger l'information binaire reçue dans les cas où un ou plusieurs bits de l'information binaire originale ont été modifiés à cause d'une transmission de mauvaise qualité (atténuation de certaines porteuses dans le canal de transmission). Trois codeurs ont été envisagés au sein du projet 4MORE, les codes LDPC, les Turbo codes et les codes convolutifs. Pour ce qui concerne la voie montante, seuls les Turbo codes et le codeur convolutif ont été retenus. Ces deux codes ont la même complexité algorithmique et ont des modes de traitement assez similaires. Leurs performances différent en fonction de leur rendement mais également en fonction de l'entrelaceur qui leur est associé. Ces codes ont des performances similaires mais leur complexité de mise en œuvre en terme de surface de silicium peut être différente. Ces codes peuvent opérer avec différents rendements suivant la qualité de service requise (qualité du canal de transmission). Les rendements de codes disponibles sont $\frac{1}{2}$, $\frac{2}{3}$ et $\frac{3}{4}$.

Ainsi, nous avons choisi de modéliser ce bloc pour un rendement de $\frac{1}{2}$, paramètre pire cas générant le plus de données en sortie du bloc. Les paramètres du codeur canal sont donc modélisés par les paramètres figurant ci après :

- Taille des données en entrée : 1 FLIT
- Temps de traitement : 64 cycles
- Taille des données en sortie : 2 FLIT

2.2.3.2 L'entrelacement

L'entrelacement a pour but d'éviter d'avoir de longues séquences de bits corrompues de même valeur induite par une dégradation du canal lors de la transmission. Grâce à la structure même de la modulation OFDM, l'entrelacement peut être effectué au niveau temporel mais également au niveau fréquentiel. Cela signifie que l'entrelacement peut être fait sur la totalité du slot (32 symboles OFDM). La fonctionnalité de ce bloc a pour objectif de répartir équitablement les bits à 0 et 1 sur l'ensemble du slot afin d'améliorer le rendement du codage canal. En effet, un codage canal est efficace si et seulement si les erreurs introduites lors de la transmission ne se font pas sur une longue séquence de bits. Sinon, le décodage devient impossible car les bits d'information et les bits de parité sont erronés et on peut atteindre la limite des capacités du codage. L'entrelacement permet de s'affranchir de ce genre de problèmes dans le cas d'une forte atténuation d'une ou plusieurs fréquences. Ainsi, ce bloc va travailler sur l'ensemble des données du slot TX et les caractéristiques de l'entrelaceur peuvent être modélisées suivant les paramètres figurant ci-après :

- Taille des données en entrée : 96 FLIT
- Temps de traitement : 6144 cycles
- Taille des données en sortie : 96 FLIT

Il faut noter que dans l'implantation réelle, l'entrelacement se fait en deux étapes. La première permutation est réalisée au niveau bit avec une contrainte induite par l'implantation qui limite la taille des blocs à traiter à 256 bits. Le deuxième permutation est quant à elle réalisée au niveau mot, c'est à dire sur 32 bits. La quantité d'information binaire contenu dans un slot est 96 FLIT (3072 bits) et le traitement nécessite deux passes, c'est pourquoi nous avons estimé le temps de traitement à 2 cycles par bit.

2.2.3.3 Conversion Binaire Symbole (CBS) ou "mapping"

Le codage binaire à symbole (CBS) consiste à associer à chaque groupe d'éléments binaires un symbole M. L'ensemble des symboles générés définit un alphabet de la modulation, dit M-aire. La règle d'affectation de n-uplets éléments binaires aux différents symboles est souvent décrite par une représentation graphique appelée "mapping" ou constellation. Cette affectation permet de minimiser la probabilité d'erreur des éléments binaires. Ici, dans cette implantation, les différents types de CBS employés sont du QPSK (Quadrature Phase Shift Keying), 8PSK (Phase Shift Keying), 16QAM (Quadrature Amplitude Modulation) et 64QAM. Cela implique nécessairement que les bits qui composent un slot sont regroupés par paquet de 2, 3, 4 et 6 bits pour chaque symbole complexe généré. Considérant l'implantation matérielle, la représentation des symboles complexes doit se faire en virgule fixe. Dans ce but, nous avons fait le choix de représenter un symbole complexe sur 32 bits, soit 1 FLIT.

Comme nous l'avons déjà dit, la modélisation des blocs fonctionnels se fait dans le pire cas notamment pour les blocs paramétrables. Ainsi, le CBS générant le plus de données en sortie de ce bloc est le cas d'une modulation 64QAM (6 bits d'information binaire par symbole contre 2 bits d'information binaire par symbole pour une QPSK). Nous garderons cette modulation dans toutes nos simulations par la suite car elle générera le plus de trafic au niveau de l'implantation NoC, nous positionnant ainsi dans les conditions pire cas. Les caractéristiques de la conversion binaire à symbole sont donc les suivantes :

- Taille des données en entrée : 1 FLIT
- Temps de traitement : 6 cycles
- Taille des données en sortie : 6 FLIT

2.2.3.4 Le AMRC

L'Accès Multiple par Répartition en Code (AMRC) ou Code Division Multiple Access (CDMA) est un système de codage permettant de réaliser l'étalement des symboles de données en utilisant les séquences de codes d'étalement de Walsh-Hadamard [76]. Le principe de cette technique est de multiplier chacun des éléments de données d'un utilisateur par un code qui lui est associé. Ainsi, les vecteurs de données de chaque utilisateur actif sont sommés pour produire le flux multi-utilisateur. Par conséquent, selon le facteur d'étalement S_f utilisé, les données étalées sont plus ou moins grandes. Comme nous l'avons mentionné dans l'introduction, pour la voie montante, la technique utilisée est du SS-MC-MA. Ainsi, chaque utilisateur se voit attribué 3 blocs de 8 sous-porteuses, impliquant $S_f = 8$. Dans ce cas précis, tous les codes de ces 3 blocs sont attribués à un seul et même utilisateur lui offrant la réservation exclusive de ses sous-porteuses pour l'émission de ses données. L'utilisateur pourra donc émettre 24 complexes par symbole OFDM. Les caractéristiques de l'AMRC sont modélisées par les paramètres figurant ci-après :

- Taille des données en entrée : 8 FLIT
- Temps de traitement : 48 cycles
- Taille des données en sortie : 8 FLIT

2.2.3.5 L'encodeur MIMO

L'encodeur MIMO (Multiple Input Multiple Output) met en forme les données utilisateur pour les émettre sur plusieurs antennes. Comme mentionné précédemment, le nombre d'antennes au niveau du terminal mobile est de 2 contre 4 pour la station de base. L'algorithme utilisé ici est basé sur l'encodage de Alamouti en étalement 1D et un mapping sur porteuse adjacente.

Les 3 vecteurs de 8 complexes produits par la fonction AMRC sont groupés en un seul et même symbole $S0$ de 24 complexes. Pour la phase d'encodage, deux symboles $S0$ et $S1$ consécutifs temporellement sont nécessaires, et un traitement mathématique est effectué sur chacun d'eux. Après calcul des symboles, on place sur l'antenne 1 à T_{S0} et T_{S1} respectivement $S0$ et $-S1^*$, et $S1$ et $S0^*$ sur l'antenne 2 à T_{S0} et T_{S1} (Cf. Figure 2.5).

Les caractéristiques de l'encodeur MIMO sont représentées par les paramètres figurant ci-après :

- Taille des données en entrée : 48 FLIT (2 symboles OFDM consécutifs)
- Temps de traitement : 50 cycles

- Taille des données en sortie : 96 FLIT (2 symboles OFDM par antenne soit 48 FLIT)

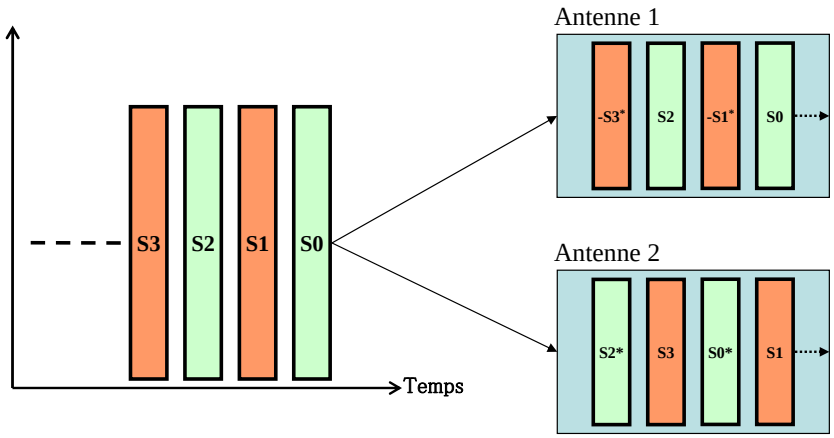

FIG. 2.5 – Schéma fonctionnel de l'encodeur MIMO

2.2.3.6 La modulation FFT

La fonction de modulation FFT réalise deux fonctions en une. La première réalise la transformée de Fourier des 24 complexes composants le symbole OFDM sur $N_{FFT} = 1024$ points. Un bourrage à zéro est effectué sur toutes les sous-porteuses qui ne possèdent aucune donnée complexe. Une fois cette étape terminée, l'étape suivante consiste à insérer un intervalle de garde au symbole OFDM. Cet IG (Interval de Garde) est nécessaire pour s'affranchir de l'interférence inter-symbole. Celui-ci consiste en une recopie des $N_{\triangle} = 256$ premiers complexes du symbole OFDM en sortie de la FFT à la fin des N_{FFT} complexes. On a ainsi en sortie de cette modulation OFDM un symbole OFDM constitué de $N_{FFT} + N_{\triangle} = 1280$ FLIT ou complexes. Les caractéristiques de la modélisation de la modulation OFDM sont données par les paramètres ci-dessous :

- Taille des données en entrée : 24 FLIT (1 symbole OFDM)
- Temps de traitement : 2620 cycles
- Taille des données en sortie : 1280 FLIT (1 symbole OFDM : symbole + IG)

Le terminal mobile fonctionnant en MIMO, ce bloc existe en deux exemplaires au sein de la chaîne, chacun prenant le flux calculé pour une antenne en sortie du bloc encodeur MIMO vu précédemment (section 2.2.3.5).

Remarque : Il faut noter que c'est précisément sur ce bloc fonctionnel que nous nous focaliserons lors de nos simulations en SystemC pour la voie montante comme pour la voie descendante. En effet, comme nous l'avons vu à la section 2.1.1 la cadence symbole à respecter pour être dans les conditions temps réel est de 20.8μs pour une trame 4MORE.

2.2.3.7 Conversion Bande de Base vers Radio-Fréquence

La conversion bande de base vers la partie radio-fréquence a en charge l'insertion du symbole de synchronisation en début de trame et la gestion de la conversion du signal

numérique en analogique au travers des CNA (Convertisseur Numérique Analogique). Ce bloc fonctionnel permet également de contrôler en permanence la puissance du signal aux transmetteurs RF, réalisant un asservissement au niveau des amplificateurs en fonction de la charge de la trame. En effet, le gain n'est pas le même en fonction du débit d'information binaire B_{info}, nécessitant un ajustement entre les pleines charges, les moyennes charges ou encore dans les cas où rien n'est transmis.

Pour ce bloc IP, nous avons choisi de modéliser uniquement le flux en entrée de celui-ci car ensuite la connexion vers la partie RF est réalisée directement au sein du bloc et ne requiert pas de communication au travers du NoC. De plus, la modélisation de ce bloc est relativement complexe car il s'agit de consignes d'asservissement de correction, pour les amplificateurs et convertisseurs, qui sont aléatoires. Ainsi, les caractéristiques de la modélisation de la conversion Bande de Base vers la partie Radio-Fréquence sont donnés par les paramètres ci-dessous :

- Taille des données en entrée : 1280 FLIT (1 symbole OFDM)
- Temps de traitement : 0
- Taille des données en sortie : 0 FLIT

2.2.4 La voie descendante

Comme nous l'avons vu auparavant (2.1.3) la technique employée ici est de type MC-CDMA. Nous allons donc voir chacun des blocs fonctionnels qui constituent la chaîne algorithmique de la voie descendante et leur modèle simplifié associé qui va nous permettre de modéliser le graphe d'application utilisé pour nos explorations de choix architecturaux. Ici aussi, certains blocs de la chaîne sont paramétrables ayant pour impact d'augmenter ou diminuer les débits d'information binaire entre blocs. Tous les modèles simplifiés des blocs présentés ci-après sont exprimés dans la plus petite unité de représentation des données au sein d'un NoC, le FLIT (1 FLIT = 32 bits).

2.2.4.1 Conversion Radio-Fréquence Bande de Base

Ce module faisant partie de la voie descendante et le terminal mobile fonctionnant en MIMO, ce bloc est donc doublé. Comme dans la voie montante, ce bloc réalise des mesures et effectue des corrections au niveau de la partie amplification, correction et conversion (CAN) pour mettre en forme le signal et le transcrire en numérique pour la phase de décodage de la bande de base. Le contrôle automatique de gain ou AGC (Automatic Gain Control) ajuste la puissance des amplificateurs pour garantir que le signal en sortie des antennes soit dans la gamme de puissance dynamique des CAN. Ces amplitudes doivent être les plus grandes possibles pour s'affranchir des erreurs de quantification, ceci en respectant la non saturation des amplificateurs (variation des amplitudes du signal reçu en fonction de la charge des utilisateurs).

Ce bloc est en charge de la synchronisation du slot en détectant le symbole de synchronisation placé en début de slot. Il est également réalisé une correction de la dérive de fréquence d'échantillonnage ou encore appellée SFO (Sampling Frequency Offset) induite par le fait que les fréquences d'échantillonnage des CNA et CAN en émission et en réception sont différentes.

Ainsi, pour ce bloc, nous avons choisi de modéliser uniquement le flux en sortie car le flux de données provient directement des antennes et par conséquent aucune commu-

nication à travers le NoC n'est nécessaire. De plus, comme nous l'avons énoncé ci-dessus, la modélisation de ce bloc est relativement complexe donnant lieu à des flux de contrôle plus ou moins important suivant la charge de la communication. Ainsi, les caractéristiques de la modélisation de la conversion Radio-Fréquence vers la partie Bande de Base sont données par les paramètres ci-dessous :

- Taille des données en entrée : 0 FLIT
- Temps de traitement : 0
- Taille des données en sortie : 1280 FLIT

Remarque : Pour nos simulations en SystemC pour la validation fonctionnelle de l'application, nous avons modélisé ce bloc comme une mémoire fournissant les symboles OFDM en provenance de la partie RF après passage des CAN.

2.2.4.2 La démodulation FFT

La fonction de démodulation FFT s'effectue en deux étapes. La première consiste à retirer l'intervalle de garde ($N_{\triangle} = 256$) inséré dans la modulation OFDM de la voie montante ayant pour fonction d'éviter les interférences inter-symbole. Ainsi, le nombre de symboles complexes passe de $N_{FFT} + N_{\triangle} = 1280$ à $N_{FFT} = 1024$. Ensuite, la seconde étape réalise la transformée de Fourier inverse des $N_{FFT} = 1024$ complexes représentant le symbole OFDM. Durant cette FFT, les bits de bourrage à zéro sont retirés pour ne conserver que les informations de pilotes et données. On a ainsi en sortie de cette modulation OFDM un symbole OFDM constitué de $N_{donnees} + N_{pilotes\_continus} = 695$ FLIT ou complexes. $N_{donnees} = 672$ représente le nombre de symboles complexes de données contenant les informations de l'utilisateur, et $N_{pilotes\_continus} = 23$ représente les symboles pilotes continus. Par conséquent, les caractéristiques de la modélisation de la modulation OFDM sont données par les paramètres ci-dessous :

- Taille des données en entrée : 1280 FLIT (1 symbole OFDM)
- Temps de traitement : 2620 cycles
- Taille des données en sortie : 695 FLIT (1 symbole OFDM)

2.2.4.3 Le CFO

Le CFO (Carrier Frequency Offset) ou correction de dérive de fréquence porteuse calcule un terme de correction utilisé par le bloc ROTOR afin de corriger cette dérive de fréquence porteuse. En effet, compte tenu du fait que nous sommes dans un équipement mobile et autonome, les fréquences d'oscillateurs en émission et réception ne sont pas exactement identiques. Ceci est d'autant plus vrai qu'une émission est réalisée de la station de base vers un terminal mobile ou bien d'un terminal mobile vers une station de base. On comprend facilement que les conditions d'utilisation et les tolérances sur les oscillateurs entraînent "quasi" systématiquement une dérive de fréquence. C'est pour cette raison que ce bloc fonctionnel CFO prend en entrée les pilotes continus d'un symbole OFDM afin d'estimer la dérive en fréquence subie par celui-ci et de donner une valeur de correction qui sera transmise au ROTOR. Ainsi, les caractéristiques de la modélisation du CFO sont données par les paramètres ci-dessous :

- Taille des données en entrée : 23 FLIT (pilotes continus d'un symbole OFDM)
- Temps de traitement : 23 cycles

- Taille des données en sortie : 1 FLIT (1 valeur de correction de CFO pour le symbole OFDM)

Cette valeur de correction sera utilisée pour corriger aussi bien les symboles OFDM de données mais également les symboles OFDM pilotes qui seront employés par le bloc d'estimation de canal.

2.2.4.4 Le ROTOR

Le module ROTOR a pour objectif de corriger la dérive en fréquence pour chaque symbole OFDM. Ainsi, il prend en entrée deux flux provenant de la démodulation OFDM et du CFO. La démodulation OFDM fournissant les $N_{donnees} = 672$ complexes des symboles OFDM et le CFO fournissant une valeur de correction associée au symbole courant. Ainsi, en sortie de ce bloc, on retrouve les symboles pilotes et de données corrigés qui vont ensuite être traités par les blocs d'estimation de canal pour les pilotes et par le décodeur MIMO pour les données. On a donc les paramètres de modélisation de ce module qui sont listés ci-après :

- Taille des données en entrée 1 : 672 FLIT (symbole OFDM)
- Taille des données en entrée 2 : 1 FLIT (valeur de correction du CFO)
- Temps de traitement : 672 cycles
- Taille des données en sortie : 672 FLIT (symbole OFDM : pilotes ou données)

Ce bloc requiert donc deux flux simultanés en entrée provenant de deux blocs différents, impliquant la nécessité d'avoir deux FIFO en entrée de ce bloc pour l'implantation matérielle.

2.2.4.5 L'estimateur de canal MIMO

Ce bloc réalise l'estimation d'un canal MIMO pour chaque antenne en réception contre deux antennes en émission (station de base). Comme nous l'avons vu dans la section 2.1.1 figure 2.2, la trame OFDM en voie descendante est composée de paquets de 8 symboles de données précédés de paires de symboles pilotes. Ainsi, l'estimation de canal est réalisée pour chaque paire de symboles OFDM pilotes. Lors de cette phase d'estimation, des coefficients de canal sont calculés pour chaque sous-porteuse. Ces coefficients permettent d'évaluer les caractéristiques du canal de transmission dans le sens ou certaines porteuses ont pu être fortement atténuées (contexte multi-trajets). D'autre part, la diversité spaciale obtenue par la méthode de [77] permet de réaliser un recoupement de ces coefficients calculés pour chacune des antennes et seront pris en compte dans l'étape d'égalisation présente au sein du bloc de décodage MIMO. Il y a donc deux coefficients calculés pour chaque symbole complexe d'un symbole OFDM. Ainsi, pour un symbole OFDM d'une antenne, on aura 1344 coefficients de correction.

Il faut noter que ces coefficients vont être également pris en compte dans le bloc fonctionnel réalisant la transposition symbole à bit ou CBS^{-1} (Conversion Binaire Symbole inverse). Ces coefficients vont permettre de calculer une probabilité de maximum de vraisemblance lors de la CBS^{-1}. Si la valeur des coefficients du symbole indique une mauvaise qualité de canal, alors une probabilité d'erreur ou LLR (Log Likelihood Ratio) pourra être calculée lors de l'attribution d'une valeur binaire sur la constellation employée témoignant d'une plus ou moins grande certitude dans la transcription symbole à bit. Les

paramètres nous permettant de modéliser ce bloc fonctionnel d'estimation de canal sont donnés ci-dessous :

- Taille des données en entrée : 1344 FLIT (2 symboles pilotes de 672 complexes)
- Temps de traitement : 1344 cycles
- Taille des données en sortie : 10752 FLIT ($8 \times 672 \times 2$ coefficients de canal)

2.2.4.6 Le décodeur MIMO

Le décodeur MIMO reçoit 4 flux de données en entrée : deux flux représentant chacun les symboles OFDM de données des ROTOR sur chaque antenne, soit 2×24 symboles OFDM pour un slot, deux autres flux correspondant aux coefficients de canal de chacune des antennes. Au sein de ce bloc, deux traitements sont effectués l'un portant sur l'égalisation des symboles de données l'autre correspondant au décodage de Alamouti. Ce dernier impose la nécessité de recevoir les symboles OFDM par paire sur chaque flux de symboles OFDM des antennes. En effet, comme nous l'avons montré dans la section 2.2.3.5, le codage de Alamouti s'effectue sur des paires de symboles ce qui implique les mêmes conditions lors du décodage. On peut donc représenter le modèle simplifié de ce bloc fonctionnel de la manière suivante :

- Taille des données en entrée 1 : 2×1344 FLIT (2 paires de symboles OFDM : 1 paire par antenne)
- Taille des données en entrée 2 : 2×2688 FLIT (2 paires de coefficients de canal : 1 paire par antenne)
- Temps de traitement : 1344 cycles
- Taille des données en sortie : 2×672 FLIT (2 symboles OFDM de données)

2.2.4.7 L'AMRC inverse

Le bloc de désétalement consiste à retrouver les informations de données de l'utilisateur dans les complexes des symboles OFDM de données. Contrairement à ce que l'on a vu dans le bloc d'étalement de la voie montante, ici un seul code d'étalement est attribué par utilisateur. Ainsi, le facteur d'étalement S_f appliqué permettra d'avoir plus ou moins de données utilisateur par symbole. Pour la voie descendante, le S_f peut varier de 8 à 32. Or pour notre modélisation des blocs fonctionnels (le but étant de modéliser une application 4G pour réaliser des explorations architecturales), nous choisissons les conditions pire cas générant le plus de besoins en bande passante. Ainsi, nous allons nous placer dans les conditions où $S_f = 8$ correspondant à un nombre de complexes de données par utilisateur de 84 contre 21 pour $S_f = 32$. Par conséquent, la modélisation de ce bloc peut être représentée de la manière suivante :

- Taille des données en entrée : 672 FLIT
- Temps de traitement : 4032 cycles
- Taille des données en sortie : 84 FLIT

2.2.4.8 Le décodage symbole à bit

Fondamentalement, le décodage symbole à bit ou encore CBS^{-1} réalise l'opération inverse de la fonction de mapping (CBS), c'est à dire qu'elle convertit les M-aire symboles com-

plexes reçus en valeur binaire en respectant les règles de mapping utilisées lors de la CBS (fonction de la constellation employée : QPSK, 16QAM, BPSK, 64QAM). Ceci implique nécessairement de prendre des décisions dures sur les M-aire symboles avec le risque d'avoir un décodage erroné si le critère de décision dans la constellation est mauvais. De plus, on sait que les décodeurs canal (Viterbi, Turbo Codes, LDPC) ont de meilleurs performances lorsqu'ils opèrent avec des "Soft bits" incluant un niveau de confiance. En effet, comme nous l'avons vu dans la section 2.2.4.5, des valeurs de maximum de vraisemblance, appelées aussi LLR, sont calculées au moyen des symboles pilotes introduits dans la trame. Ceux-ci permettent de donner une indication d'atténuation dans le canal de propagation sur les symboles de données.

Ainsi, le but du CBS^{-1} est de produire des "Soft bits" en prenant en compte les valeurs de LLR calculées dans le bloc de décodage MIMO, ceci afin de prendre une décision de valeur binaire sur la constellation avec une probabilité de plus ou moins grande certitude relative à une plus ou moins grande atténuation du symbole lors de sa propagation dans le canal de transmission. Ces "Soft bits" sont ensuite envoyés à l'entrelaceur et au décodeur canal afin de restituer les valeurs binaires d'origine en tenant compte de ces incertitudes de décodage. Ces "Soft bits" sont codés sur 4 bits au lieu de un pour un décodage matériel dur. Comme énoncé précédemment, nous nous plaçons dans les conditions pire cas pour les contraintes de bande passante au niveau de l'adéquation avec l'architecture NoC FAUST. Ainsi, la modélisation de ce bloc va se faire pour une constellation de 64 QAM comme dans la voie montante, cas où pour un symbole complexe nous allons avoir 6 Soft bits, soit 24 bits pour la représentation binaire. On constate que ceci n'est pas un multiple de la taille d'un FLIT (32 bits). C'est pourquoi la modélisation de ce bloc va s'effectuer pour un traitement par bloc de 12 symboles complexes correspondant à 72 Soft bits ($\frac{72\times[4bits]}{32bits} = 9FLIT$) soit 9 FLIT. Par conséquent, la modélisation de ce bloc peut être représentée de la manière suivante :

- Taille des données en entrée : 12 FLIT
- Temps de traitement : 72 cycles
- Taille des données en sortie : 9 FLIT

2.2.4.9 Le désentrelacement

La fonction de désentrelacement du canal est l'opération inverse de l'entrelacement effectuée lors de l'émission. La différence majeure réside dans le fait qu'à l'émission l'entrelacement est effectué au niveau de l'information binaire matérielle (1 bit par donnée binaire) alors que le désentrelacement s'effectue au niveau de l'information binaire logicielle (4 bits pour un Soft bit). Comme dans la voie montante, le désentrelacement s'opère sur la totalité du slot. Ainsi, les éléments permettant de modéliser le comportement de ce bloc sont décrits ci-dessous :

- Taille des données en entrée : 2016 FLIT
- Temps de traitement : 2016 cycles
- Taille des données en sortie : 2016 FLIT

2.2.4.10 Le décodeur canal

Le décodage canal permet de restituer l'information binaire utile B_{info} encodée lors de l'émission. Ce codage augmente la robustesse de la transmission au moyen de bits de

parité permettant de corriger les éventuelles erreurs de transmission (mauvaise qualité du canal de transmission). Plus le rendement de code est élevé et moins le codage décodage est robuste aux erreurs. Ainsi, comme vu dans l'encodeur canal de la voie montante (2.2.3.1), le décodeur canal possède trois rendements de code possible : $\frac{1}{2}$, $\frac{2}{3}$ et $\frac{3}{4}$. Il faut noter ici que ce décodeur canal prend en entrée des Soft bits. Ceux-ci possèdent une information de probabilité de maximum de vraisemblance calculée lors de la CBS^{-1} qui va être utilisée conjointement au décodage pour prendre une décision au niveau bit la plus cohérente possible (la valeur la plus vraisemblable aux vues de la valeur de LLR et du décodage des bits de parité). Ainsi, les données en entrée au format FLIT vont être des "Soft bits" codés sur 4 bits qui seront transcrits en valeur binaire après retrait des bits de codage.

Nous nous plaçons dans un rendement de code de $\frac{1}{2}$, paramètre pire cas générant le plus de données en sortie du bloc. Les paramètres du codeur canal sont donc modélisés par les paramètres figurant ci-après :

- Taille des données en entrée : 8 FLIT (64 Soft bits)
- Temps de traitement : 32 cycles
- Taille des données en sortie : 1 FLIT (32 bits pour un rendement de $\frac{1}{2}$)

2.2.5 Paramètres de modélisation de l'application 4MORE

Nous avons décrit les différents blocs fonctionnels qui composent les algorithmes des voies montantes et descendantes. Afin de récapituler les différents modèles équivalents de ces blocs fonctionnels en vue des phases d'explorations architecturales sur NoC, nous listons ci-après les descriptifs des voies montante et descendante. Le tableau 2.3 référence les valeurs des paramètres des blocs de la voie montante et le tableau 2.4 ceux de la voie descendante. Nous nous référerons à ces tableaux dans le chapitre 3 pour décrire notre graphe d'application pour réaliser notre AAA dans le cadre de notre contribution au sein du WP4 du projet 4MORE.

TX	**Input Data**	**Compute Time**	**Output Data**
Codeur Canal	1	64	2
L'entrelacement	96	6144	96
CBS	1	6	6
L'AMRC	8	48	8
L'encodeur MIMO	48	50	96
La modulation FFT	24	2620	1280
Bande de base vers RF	1280	0	0

TAB. 2.3 – Tableau récapitulatif des modélisations des blocs fonctionnels de la voie montante

RX	**Input Data**	**Compute Time**	**Output Data**
RF vers Bande de Base	0	0	1280
La démodulation FFT	1280	2620	695
CFO	23	23	1
ROTOR	672, 1	672	672
L'estimateur de canal	1344	1344	10752
Le décodeur MIMO	2×1344, 2×2688	1344	1344
$L'AMRC^{-1}$	672	4032	84
CBS^{-1}	12	72	9
Désentrelacement	2016	2016	2016
Le décodage canal	8	32	1

TAB. 2.4 – Tableau récapitulatif des modélisations des blocs fonctionnels de la voie descendante

2.3 Conclusion

Dans ce chapitre nous avons situé le contexte du projet 4MORE dans lequel nous avons été impliqué dans la cadre du WP4 (Work Package 4). Ce WP4 a pour mission de proposer des solutions de CoDesign pour la réalisation du SoC final du terminal mobile. Dans ce projet, le CEA LETI a proposé son NoC FAUST comme média de communication entre les blocs IP de la voie montante et de la voie descendante du terminal mobile de 4MORE pour la réalisation du SoC final. Dans ce contexte, l'exploration architecturale du NoC FAUST pour vérifier la faisabilité de l'implantation au niveau respect des contraintes temps réel a été réalisée au moyen de simulations en SystemC. Le modèle SystemC du NoC FAUST est en TLM ce qui permet de s'affranchir de la nécessité de disposer du bloc IP fonctionnel réel. De ce fait, une modélisation simple des blocs IP a pu être envisagée afin de vérifier le respect des contraintes temps réel de l'application. C'est pour cette raison que dans ce chapitre nous avons présenté le schéma de modélisation de tous les blocs IP de la chaîne pour la simulation de ceux-ci sur le réseau. Tous les blocs IP des voies montante et descendante ont leur modélisation simple associée qui possède des caractéristiques équivalentes aux blocs IP VHDL utilisés pour l'implantation finale.

De plus, le NoC employé est de type "Wormhole"' "packet switching" avec une qualité de service en BE impliquant nécessairement la non garantie de bande passante pour les communications au sein du réseau. Ce genre de réseau implique le besoin de simulations pour valider les choix de topologie, de chemins de communications ou encore le dimensionnement des FIFO au sein des NI. Ceci permet de dimensionner au plus juste l'architecture pour garantir les contraintes temps réel de l'application dans le pire cas d'utilisation. C'est pour cette raison que nous allons voir dans le chapitre suivant les caractéristiques du NoC FAUST utilisé avec son flot de conception associé.

Chapitre 3

Le réseau FAUST et sa méthodologie de conception associée

Sommaire

Dans ce chapitre, nous allons voir les caractéristiques du réseau FAUST et sa méthodologie de conception associée. Ce NoC intègre le schéma de modélisation simplifié des blocs de traitement (cf. chapitre 2 section 2.4) dans le flot de conception du NoC. Le but étant de valider des choix architecturaux parmi l'espace d'exploration architectural, en réalisant des simulations SystemC TLM précis au niveau cycle. Ces simulations ont pour objectif de valider les différents paramètres majeurs qui caractérisent les NoC afin d'améliorer l'adéquation de l'application en cours d'exploration avec les choix architecturaux proposés par la structure du NoC. L'ensemble de ces paramètres doit être exploré séquentiellement ou conjointement selon les cas. L'ensemble de ces différentes simulations vont nous permettre de vérifier le respect des contraintes temps réel de l'application et la faisabilité du SoC.

Nous allons présenter, dans ce chapitre, l'architecture du réseau FAUST et ses caractéristiques. Ce réseau FAUST a été développé au sein du laboratoire LETI du CEA de Grenoble. Ensuite, nous verrons la gestion du flot de données au sein de ce réseau avec notamment sa méthodologie de configuration. Enfin, nous détaillerons le modèle SystemC de celui-ci et son flot d'exploration associé afin de réaliser des explorations architecturales du NoC avec une application associée.

3.1 Présentation

Comme nous l'avons vu dans le chapitre 1, les réseaux sur puce offrent des alternatives aux problèmes rencontrés actuellement par les topologies bus employées dans les SoC. Ces limitations étant principalement induites par des applications aux besoins grandissant en bande passante (communications à forts besoins en bande passante en parallèle) mais également induites par l'accroissement de la taille des SoC qui intègrent de plus en plus de standards. En contrepartie, ces NoC souffrent d'un inconvénient majeur qui est une complexité de mise en œuvre importante.

En effet, plusieurs paramètres au sein du NoC peuvent être modifiés afin d'adapter l'architecture à l'application utilisée. Ces paramètres ont un impact direct en terme de besoins en ressources matérielles (Surface de silicium, besoin en mémoire, fréquence de fonctionnement, ...) mais également en terme de performances globales offertes à l'application. C'est pour ces raisons que la nécessité d'avoir un outil d'aide à la conception s'impose.

Cet outil va permettre d'aider le concepteur dans ses choix architecturaux pour garantir le respect des contraintes imposées par l'application lors de la réalisation du SoC final sur architecture cible dédiée (ASIC) ou reprogrammable (FPGA). L'outil va permettre d'explorer l'espace des solutions architecturales au moyen de simulations. Nous avons fait le choix de réaliser ces simulations en SystemC TLM précis au niveau cycle afin de bénéficier de la souplesse de ce langage, mais aussi de sa rapidité de simulation (contrairement à une représentation en VHDL où les simulations peuvent être longues). Ainsi, pour pouvoir intégrer les contraintes de l'application pour laquelle on souhaite trouver la meilleure adéquation avec l'architecture cible visée, deux possibilités sont offertes pour effectuer ces explorations. Soit les algorithmes des blocs IP sont décrits de manière complète en SystemC, soit ils sont décrits selon un modèle simplifié permettant de s'affranchir de la disponibilité ou non du code complet de l'algorithme. Nous avons choisi la deuxième solution car d'une part l'outil AAA va pouvoir être utilisé pour différentes applications rapidement et d'autre part parce que le but recherché dans ces simulations est le respect des contraintes temporelles de l'application et non une validation fonctionnelle des algorithmes (cette étude est supposée être effectuée en amont).

Ce modèle générique de représentation des IP permet de simuler rapidement et simplement l'application mais également de s'affranchir de la disponibilité des "vrais" blocs IP en SystemC, ou VHDL pour l'implantation. De plus, ce formalisme de représentation donne l'avantage de ne se focaliser que sur les contraintes de l'application au niveau temporel et non pas sur les aspects fonctionnels qui bien souvent sont une étape coûteuse en temps dans le développement d'une application.

Cette modélisation présente également un intérêt majeur dans le cas d'explorations par émulation, car la possibilité offerte par la généricité et la paramétrisation des blocs

IP de manière logicielle permet d'émuler différents choix architecturaux sans avoir à recommencer le flot de conception. Celui-ci est similaire à celui employé dans l'outil EDK (Embedded Development Kit) de Xilinx [78] où le bitstream (le bitstream étant un fichier contenant des informations binaires matériel permettant de configurer un FPGA pour un design donné) de configuration du FPGA contient des informations binaires de configuration matérielle (plan de câblage de la puce réalisant les fonctions logiques du design) et logicielle (initialisation du contenu des mémoires ou BlockRAM). Lorsqu'une modification logicielle est nécessaire, le flot de conception matériel ne nécessite pas d'être recommencé, une simple mise à jour (assemblage) du bitstream est effectuée afin de remplacer uniquement la séquence de bits ayant attrait à la partie logicielle.

Cette méthode offre un gain de temps substantiel pour l'émulation sur carte, car les blocs prennent des comportements différents uniquement par paramétrage logiciel. Nous verrons par la suite dans le chapitre 4 des résultats d'émulations effectués sur des plateformes de développement de la société Nallatech [79] à base de FPGA virtex4 [80] Xilinx.

Nous allons donc voir dans ce chapitre l'architecture du NoC FAUST [81] et ses caractéristiques. Ensuite, nous verrons en détail le flot de conception que nous avons établi. Et enfin, nous étudierons dans quelles conditions nous validerons les applications pour les simulations ou les émulations.

3.2 L'architecture du NoC FAUST

Le réseau présenté ici utilise le mode de commutations par paquets ("Packet switching") car il offre une plus grande flexibilité que la commutation de circuit ("Circuit switching"). De plus, le mécanisme de gestion de flux des communications employé repose sur la technique "Wormhole" car elle offre le plus faible temps de traversée possible à travers les routeurs et avec un coût en mémorisation faible au sein de ceux-ci.

Ce NoC possède un mécanisme de requête avec accusé de réception lors des communications ce qui, par conséquent, garantit la non destruction de FLIT, ceci étant conditionné par une indication non erronée du chemin des communications dans l'en-tête du paquet. Le mécanisme de "poignée de mains" est utilisé conjointement avec un contrôle de flux basé sur des paquets de crédits permettant de s'assurer qu'il y ait assez de places dans la FIFO de destination avant d'émettre. Les paquets qui peuvent rencontrer une contention dans le réseau ne seront pas détruits, mais cette contention engendrera inévitablement une latence dans l'acheminement des données de la communication, influençant directement les performances de l'application. La QoS apportée aux communications est de type BE.

3.2.1 Le routeur

Le routeur du NoC FAUST possède cinq ports réseaux. Quatre d'entre eux sont réservés à la connexion des routeurs entre eux, le cinquième permettant de connecter un bloc fonctionnel représentant une tâche de l'application. Ce bloc peut être un bloc IP dédié réalisé en VHDL ou bien un processeur matériel ou logiciel (DSP, ARM ou Microblaze par exemple). Le technique de gestion de flux des communications utilisée est la technique "Wormhole" impliquant le besoin de buffers pour chaque lien entrant du routeur. Chacun de ces buffers a une capacité de 1 FLIT plus deux bits d'information, un de début et un de fin de paquet, soit une largeur de 34 bits. Sur la figure 3.1 représentant le routeur,

les buffers nécessaires à la technique “Wormhole” sont intégrés dans les ports d’entrées du routeur (Input Port ou IP).

Comme nous l’avons vu dans le chapitre 1, les communications au sein du réseau se font par paquets de FLIT pour lesquels est ajouté un FLIT d’en-tête comportant des informations de chemin de routage vers la cible, l’identifiant de la cible, la commande, et des paramètres de contrôle. Ainsi ce FLIT d’en-tête également appelé “header” est lu et décodé par l’arbitre du routeur afin de prendre les décisions de routage adéquates et gérer le niveau de priorité de l’information (cf. 3.2.4). Le contrôle de ces flux d’information en entrée et en sortie de ces liens est réalisé au moyen d’un acquittement entre émetteur et récepteur. Le routeur possède deux canaux virtuels pour lesquels des signaux d’acquittement (send + accept) sont dédiés pour chacun de ces canaux. L’architecture du routeur est représentée sur la figure 3.1.

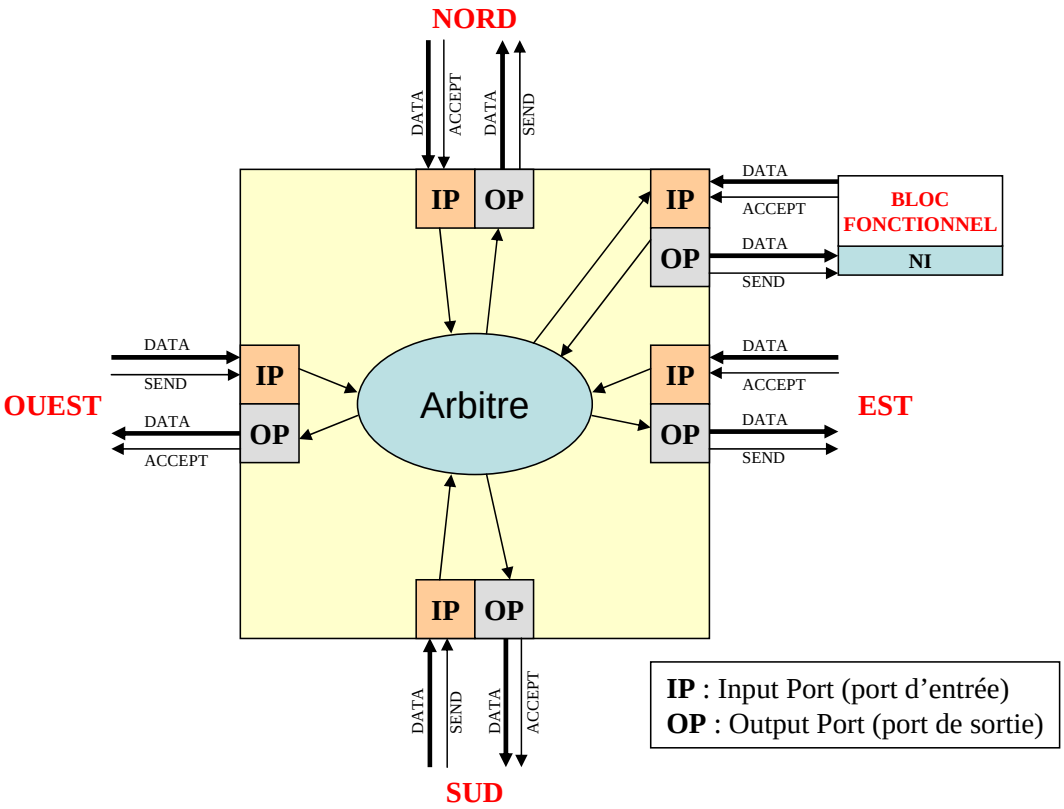

Fig. 3.1 – Architecture du routeur de FAUST

Les étapes de décodage de l’en-tête du paquet, d’arbitrage et de commande du multiplexeur pour aiguiller les FLIT vers le bon port de sortie sont effectuées en un seul temps de cycle. Ce routeur offre donc une latence d’un seul temps de cycle d’horloge pour le passage d’un routeur vers un autre ou d’un routeur vers un bloc fonctionnel. Ainsi, pour un réseau opérant à une fréquence de 100MHz, la bande passante théorique maximale disponible sur ce lien sera de 3200 Mb\s (800 Mo\s). Il faut noter, dans ce cas précis, que pour chaque paquet de données émis dans le réseau un FLIT d’en-tête est ajouté à celui-ci. Ce FLIT

d'information de paquet réduit la bande passante utile offerte pour l'application. Pour des tailles de paquets de données fixées à 8 FLIT, un FLIT d'en-tête est ajouté au paquet, engendrant un rendement de 89% environ. Les mécanismes de routage des communications au sein du réseau engendrent systématiquement une perte de bande passante pour l'application. C'est un critère qu'il faut garder en mémoire lorsque l'on souhaite réduire la fréquence de fonctionnement du NoC pour entrer dans des conditions basse consommation où indirectement on réduit les bandes passantes disponibles sur les liens avec le risque de ne plus garantir les contraintes temps réel [82, 83, 42, 35].

3.2.2 Les canaux virtuels

Les canaux virtuels ont pour objectif de permettre de donner des niveaux de priorité aux communications à travers le réseau. Ici le réseau employé possède deux canaux virtuels ayant la même qualité de service (QoS) à savoir le BE. Les canaux virtuels ont donc un ordre de priorité qui correspond à leur numéro. Le canal numéro 1 est celui de niveau le plus prioritaire. Ainsi, lorsqu'une requête est effectuée sur le canal numéro 1 celui-ci est le plus prioritaire pour accéder au lien demandé et spécifié dans les champs du FLIT d'en-tête. Bien-entendu, un canal virtuel n'est considéré candidat que s'il a au minimum un FLIT à envoyer et qu'il possède également des crédits pour émettre sur ce lien et plus particulièrement vers la cible souhaitée. C'est donc l'arbitre du routeur qui va administrer ces niveaux de services en prenant en compte le niveau de priorité en cours sur chaque lien pour que les autres requêtes ou communications en cours sur l'autre canal soient gelées. La QoS de type BE offre une exploitation maximale de la bande passante permettant, lorsque des niveaux de priorité sont spécifiés, de rendre une ou plusieurs communications plus prioritaire par rapport aux autres. Ainsi, le canal virtuel numéro 1 va être de QoS BE prioritaire et permettra de mettre l'accent sur la priorité de certaines communications.

3.2.3 La politique d'arbitrage

L'arbitrage mis en place au sein du réseau est effectué conjointement avec les signaux d'acquittement des ports d'entrée du routeur. En effet, chaque port d'entrée du routeur possède différents signaux et bus permettant d'effectuer les acquittements de réception des FLIT. Les différents signaux d'une communication entre deux routeurs sont listés sur la figure 3.2.

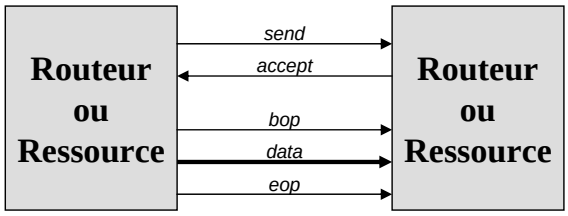

FIG. 3.2 – Signaux d'acquittement de données entre deux routeurs ou un routeur et une ressource

Les rôles remplient par chacun de ces signaux dans les mécanismes d'acquittement d'échange de données entre routeurs ou entre routeur et ressource sont détaillés ci-dessous :

- *Accept* : signal permettant de signifier que le récepteur peut recevoir des informations (reste à 1 tant que le récepteur peut recevoir des FLIT)
- *Send* : signal permettant de signifier que l'émetteur est en train d'envoyer des données (mis à 1 lorsqu'un FLIT est envoyé)
- *data* : bus unidirectionnel de données de largeur 32 bits
- *bop* : signal marqueur de début de paquet (actif à 1)
- *eop* : signal marqueur de fin de paquet (actif à 1)

Pour chaque canal virtuel, une paire de signaux d'acquittement *send* et *Accept* est associé. Ainsi, par l'intermédiaire de ces signaux, l'arbitre du routeur prend en compte la disponibilité du lien de sortie visé, mentionné dans le "header" pour prendre sa décision d'arbitrage. La politique d'arbitrage mise en place au sein de ce routeur est à priorité tournante (tourniquet ou round-robin) [29] permettant d'éviter les famines. La figure 3.3 montre un exemple de chronogramme des signaux d'acquittement pour un échange de données entre deux routeurs ou bien entre routeur et ressource et vice et versa.

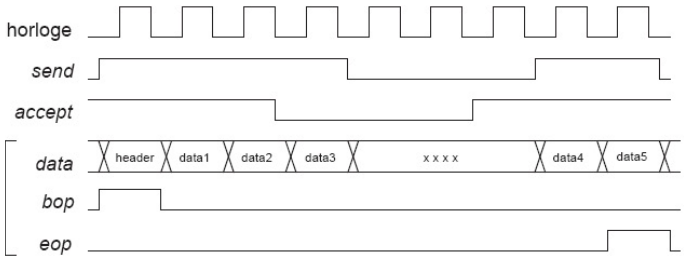

FIG. 3.3 – Chronogramme d'acquittement de données au sein du NoC FAUST

Nous allons maintenant voir plus en détail la structure du FLIT nommé "header" qui encapsule les paquets de données lors de leur trajet au sein du NoC.

3.2.4 La structure du paquet

Conformément à la technique de gestion des communications employée ici, le "Wormhole", chaque paquet de données émis à travers le réseau se voit automatiquement greffé d'un FLIT supplémentaire renseignant le paquet lui-même. Ces informations sont nécessaires pour pouvoir spécifier au routeur mais également à la ressource cible les caractéristiques des données et également celles de la communication.

Le paquet est identifié par le signal *bop* à l'état 1 et la fin de celui-ci est identifiée par le signal *eop* à l'état 1. Le dernier FLIT identifié par le signal *eop* est soit un FLIT de données lorsqu'il s'agit d'un paquet de données, soit le "header" lui-même s'il s'agit d'un paquet de crédit. Les signaux de début et fin de paquet permettent de garder une cohésion des FLIT appartenant au même paquet lors de leur arbitrage au sein du routeur. Le FLIT d'en-tête possède donc différents champs caractérisant la transmission et la ressource réceptrice. Ils sont détaillés sur la figure 3.4. La structure des FLIT de données suivant le FLIT d'en-tête est donnée sur la figure 3.5.

La topologie maillée à deux dimensions offre cinq directions possibles : Nord, Sud, Est, Ouest et Ressource. Chaque direction au sein du réseau est codée sur 2 bits avec la particularité de ne pas pouvoir effectuer de demi-tour. C'est cette exception qui permet

d'adresser la ressource (vers le 5^{e} port de sortie du routeur où est connectée la ressource de traitement) en retournant le paquet sur la direction dont il provient.

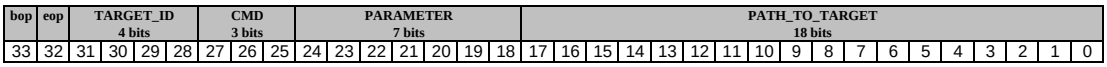

FIG. 3.4 – Structure du FLIT d'en-tête ("header")

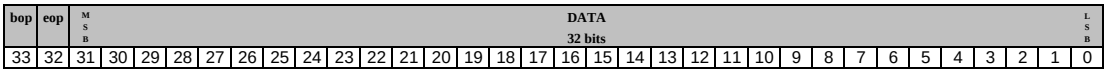

FIG. 3.5 – Structure d'un FLIT de données

Par cette astuce de codage, on économise 1 bit de codage pour les 5 directions possibles qui en auraient nécessité 3 bits en théorie avec une sous exploitation des possibilités de codage. Les valeurs de codage attribuées à chacune des 4 directions sont données dans le tableau 3.1.

Direction	**B1**	**B0**
Nord	0	0
Est	0	1
Sud	1	0
Ouest	1	1

TAB. 3.1 – Codage des directions

Le chemin vers la cible (PATH_TO_TARGET) permet de réaliser 9 changements de direction incluant le demi-tour permettant d'adresser la ressource. Les ressources de traitement dépendantes de données ne peuvent pas être distantes de plus de 8 liens. Les données du message qu'une ressource veut émettre vers une autre sont découpées en paquet de taille raisonnable de façon à éviter un blocage du réseau. Généralement ces paquets ont une taille de 8 FLIT mais cette taille est paramétrable au sein des interfaces réseau de chaque ressource de traitement. Ainsi, les deux ressources s'échangeant des paquets de données et de contrôle représentés par un FLIT d'en-tête et des FLIT de données. Pour pouvoir réguler la communication, le NoC FAUST dispose d'un mécanisme de gestion de flux par crédit. L'en-tête du paquet contient des paramètres intervenant dans la régulation de la communication. Ainsi, le champ "TARGET_ID" (cf. Tableau 3.2) renseigne la destination du paquet dans la ressource. Le champ "CMD" (cf. Tableau 3.3) spécifie la nature du paquet. Celui-ci pouvant être un paquet de données pour lequel on souhaite faire une action particulière (Lecture ou écriture) ou bien un paquet de crédits. Dans le cas du paquet de crédits, le paquet est réduit à un seul FLIT correspondant au FLIT d'en-tête dans lequel le champ "PARAMETER" est renseigné par le nombre de crédits (nombre de places disponibles dans la FIFO entrante de la ressource cible). Ce paquet de crédits est envoyé

à partir d'un seuil de FLIT dans la FIFO qui a été paramétré dans l'interface réseau de la ressource.

Valeur	Destination
0000	Registres de configuration
0001 à 1111	Mémoire FIFO

TAB. 3.2 – Spécification du champ TARGET_ID

Valeur	Commande
000	READ
001	WRITE
010	INITIAL WRITE
011	SEND CREDIT

TAB. 3.3 – Spécification du champ CMD

Ainsi, la modélisation de l'émission d'un paquet entre un émetteur et un récepteur au sein du réseau peut être représenté sur la figure 3.6.

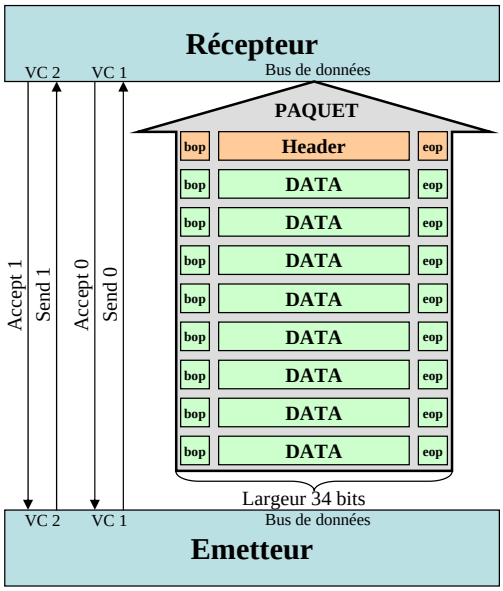

FIG. 3.6 – La transmission d'un paquet sur un lien unidirectionnel

Nous allons maintenant voir la composition de l'interface réseau permettant de faire l'adaptation de protocole réseau avec le bloc fonctionnel. En effet, pour le cas du NoC FAUST, aucun standard d'encapsulation des blocs IP n'a été employé. La norme d'encapsulation des IP est propriétaire du NoC et dédiée à celui-ci.

3.2.5 L'interface réseau

Le réseau FAUST a été conçu comme une plate-forme de communication offrant la possibilité d'intégrer plusieurs unités de traitement de nature hétérogène. Afin de pouvoir intégrer ces unités de traitement, des adaptateurs de protocole aussi appelés "wrapper" sont nécessaires pour simplifier leur intégration dans le NoC. Ces adaptateurs de protocole ou interface réseau (NI) ont une double fonctionnalité. D'une part, elles permettent de gérer les communications entrantes et sortantes du bloc fonctionnel avec le réseau (gestion des paquets de crédits et de données). Et d'autre part, elles permettent de prendre en charge les configurations de traitement des données entrantes et leurs enchaînements.

Ainsi, l'architecture de cette NI est constituée de différents éléments matériels qui sont assemblés et configurés en fonction des besoins de la ressource de traitement. L'ensemble de ces éléments paramétrables le sont au moyen de registres permettant de spécifier les caractéristiques des communications de données et de crédits mais également le comportement du bloc fonctionnel pour les traitements. La figure 3.7 présente un exemple d'architecture de l'interface réseau de FAUST.

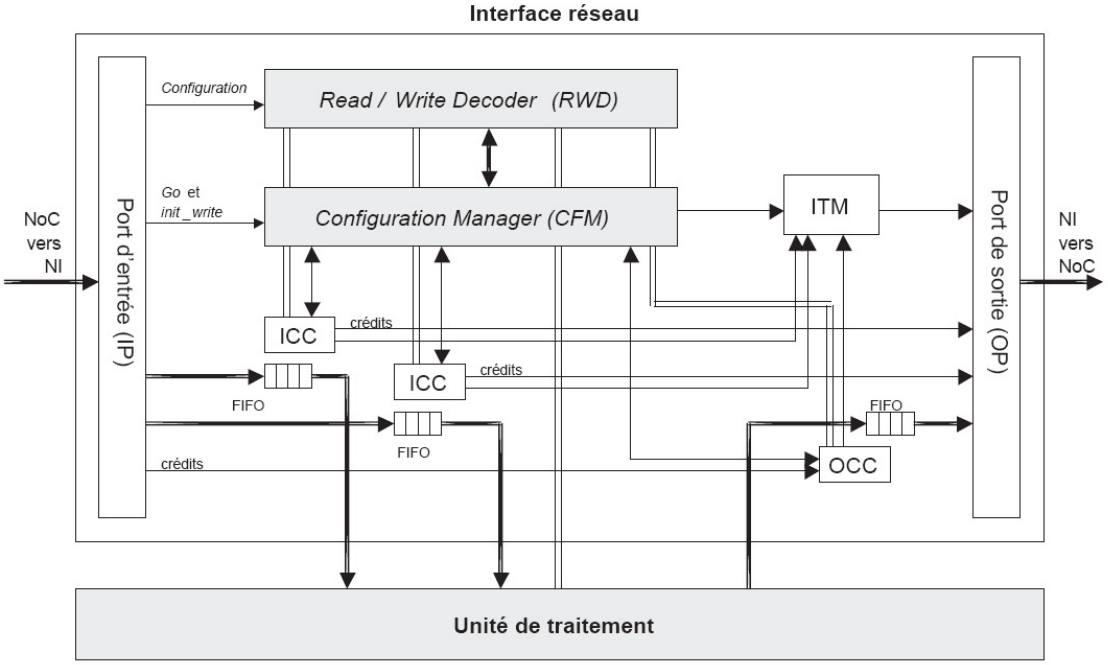

FIG. 3.7 – Structure de l'interface réseau

Nous allons voir maintenant plus en détail chacun des éléments principaux qui constituent cette NI.

3.2.5.1 Les ports d'entrée et de sortie (IP et OP)

Les ports d'entrée et de sortie sont le premier lien entre la ressource et le réseau. Ils ont pour rôle de gérer les communications entrantes et sortantes du bloc fonctionnel au niveau FLIT et paquets. Cette gestion s'effectue au moyen des signaux *bop* et *eop* pour la gestion

des paquets et au moyen des signaux *send* et *accept* pour la gestion des FLIT ainsi que des canaux virtuels.

Dans le cas d'une réception, le port d'entrée reçoit le FLIT d'en-tête et le décode pour ensuite aiguiller les données du paquet dans les autres éléments composant l'interface au moyen des champs CMD et TARGET_ID (cf. Figure 3.4). Dans le cas d'une émission, le port de sortie arbitre les différents modules qui souhaitent émettre des paquets de données ou de service à un autre bloc fonctionnel.

3.2.5.2 Les contrôleurs de communications entrantes et sortantes

La réception et l'émission de paquets de données entre la ressource de traitement et le réseau sont gérées par l'intermédiaire des ports d'entrée et de sortie vu précédemment. Or les données reçues ou émises par ces ports doivent être liées avec la ressource de traitement. Ainsi, la réception et l'émission de ces données au niveau du cœur de traitement s'effectue au moyen de FIFO. La gestion des communications du réseau est basée sur une gestion par crédits. Or ceux-ci signalent à un émetteur s'il y a suffisamment de places mémoires disponibles au niveau du récepteur pour qu'il puisse émettre ses données. Ces crédits sont donc l'image de l'occupation de la FIFO entrante du récepteur.

Ainsi, l'interface réseau possède un contrôleur de communications pour chaque FIFO entrante ou sortante. Le nombre de FIFO, et indirectement de contrôleurs de communication, dépend directement du nombre de flux d'entrée et de sortie que requiert l'unité de traitement. Lors de la réception des données, l'ICC (Input Communication Controller) assure l'envoi des FLIT de crédits aux émetteurs en fonction de l'état de remplissage de la FIFO dont il a la charge. Dans le cas de l'émission de données, l'OCC envoie les paquets de données produits par l'unité de traitement aux récepteurs sous réserve que celui-ci est reçu suffisamment de crédits de la part de la ressource de destination.

3.2.5.3 Le Read/Write decoder

Le RWD (Read/Write decoder) contient les registres de configuration des éléments de contrôle des communications et de configuration de l'unité de traitement. Le RWD est donc spécifique pour chaque unité de traitement car les éléments matériels de contrôle des communications sont assemblés et configurés en fonction des besoins de la ressource de traitement. Il décode et exécute les ordres d'écriture des registres de configuration qui sont envoyés par le réseau via un processeur de contrôle. Une fois ces registres initialisés, les différents paramètres mémorisés ayant attrait aux différents blocs matériels composant la NI sont accessibles par leurs destinataires respectifs.

3.2.5.4 Le gestionnaire d'interruption

Le gestionnaire d'interruption ou ITM (Interrupt Manager) va créer des paquets permettant de rapporter des informations ou évènements au processeur contrôlant l'application. Ces paquets rendent compte de la fin d'un traitement, de la fin d'une configuration de traitement, d'une erreur de changement de traitement ou encore une erreur de traitement de l'unité de traitement. Ce gestionnaire d'interruption permet de réaliser un asservissement de certains blocs fonctionnels paramétrables où l'on peut par exemple modifier un bloc de filtrage ou encore une CBS lors d'une communication.

3.2.5.5 Le gestionnaire de configuration

Nous avons présenté dans les sections ci-dessus les différents éléments qui permettent de gérer les flux de données et de crédit au sein de l'interface réseau. Or le fonctionnement de ces contrôleurs de communication et de l'unité de traitement nécessite un certain nombre de paramètres pour opérer correctement. En effet, on comprend facilement que l'émission de données ou de crédits nécessite le besoin de spécifier un certain nombre de paramètres dans l'en-tête du paquet afin de cibler le bon récepteur ou le bon émetteur. Il est donc nécessaire d'avoir un module matériel qui prend en compte le chargement des configurations vers les éléments de contrôle de communication concernés. C'est le rôle du gestionnaire de configuration appelé CFM (Configuration manager).

Nous avons vu dans la section 3.2.5.3 que les différents contrôleurs de communication étaient configurés par une série de paramètres qui sont mémorisés au sein du bloc RWD. Il faut donc charger les paramètres de configuration aux différents éléments de contrôle de communications concernés en fonction du traitement réalisé par la ressource. Le CFM commande l'exécution et l'enchaînement de ces configurations aux contrôleurs lorsque ceux-ci signalent qu'ils sont prêts à recevoir une nouvelle configuration. Pour mieux comprendre le rôle du CFM, nous allons détailler le contenu des registres de configuration de l'ICC puis de l'OCC.

L'ICC a pour rôle de gérer le flux de données entrant dans l'interface réseau vers la FIFO de stockage, élément permettant de fournir les données à la ressource de traitement. L'ICC va donc envoyer des paquets de crédits à la ressource émettrice dès lors que des éléments mémoires sont disponibles dans cette FIFO. Les paquets de crédits ont la particularité d'être composés d'un seul FLIT qui est l'en-tête dans lequel on spécifie le nombre de crédits disponibles dans la FIFO. Par conséquent, il est nécessaire de définir un seuil de crédit à partir duquel on souhaite émettre un paquet de crédits. Sous peine de quoi on va se trouver dans une situation paradoxale dans laquelle on va envoyer un paquet de crédits à chaque fois qu'un élément mémoire dans la FIFO se libère. Ainsi, les différents registres permettant de caractériser le fonctionnement de l'ICC sont présentés dans la figure 3.8.

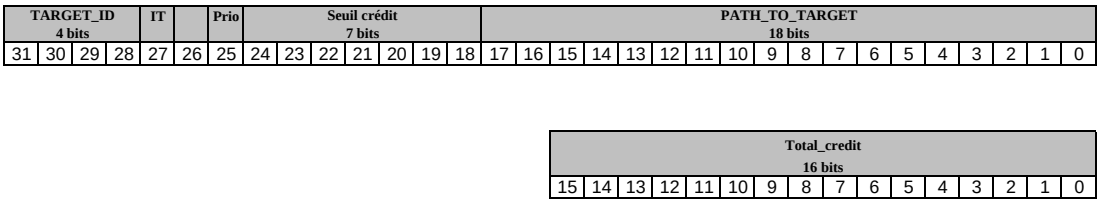

FIG. 3.8 – Registres de configuration de l'ICC

Le gestionnaire de configuration ou CFM (Configuration Manager) réalise le lien entre tous les autres éléments de la NI afin de distribuer les éléments de configuration nécessaires à chacun d'entre eux. Ainsi, les gestionnaires des communications (IP et OP) lorsqu'ils reçoivent et émettent des données doivent connaître la configuration en cours, le chemin des paquets de données et de crédits, ainsi que les niveaux de FLIT sur les FIFO entrantes et sortantes pour lesquels il faut générer des paquets de crédits ou émettre des paquets de données.

L'ICC envoie donc un paquet de crédits dès que le nombre d'éléments mémoires disponibles dans la FIFO dépasse la valeur *seuil_ credit*. De plus, l'ICC est configurée pour envoyer un certain nombre de credits (*Total_ credit*) qui est en réalité la quantité de FLIT de données que va recevoir l'unité de traitement pour cette configuration. Le chemin de la destination des crédits est mentionné par *PATH_ TO_ TARGET* et le canal virtuel employé est mentionné par le champ *prio*. Quant au champ *TARGET_ ID*, celui-ci correspond au registre de stockage des crédits dans la ressource réceptrice au niveau de l'OCC. Enfin, le bit *IT* permet de valider ou non la possibilité d'émettre une interruption vers le processeur de contrôle à la fin d'un traitement.

En ce qui concerne les registres de configuration de l'OCC, ceux-ci sont présentés sur la figure 3.9. L'OCC est configuré pour envoyer un certain nombre de FLIT de données mentionné par le champ *Total_ donnée*. Ce montant total de données à envoyer pour cette configuration correspond au nombre de FLIT de données qui vont être produits par la ressource de traitement.

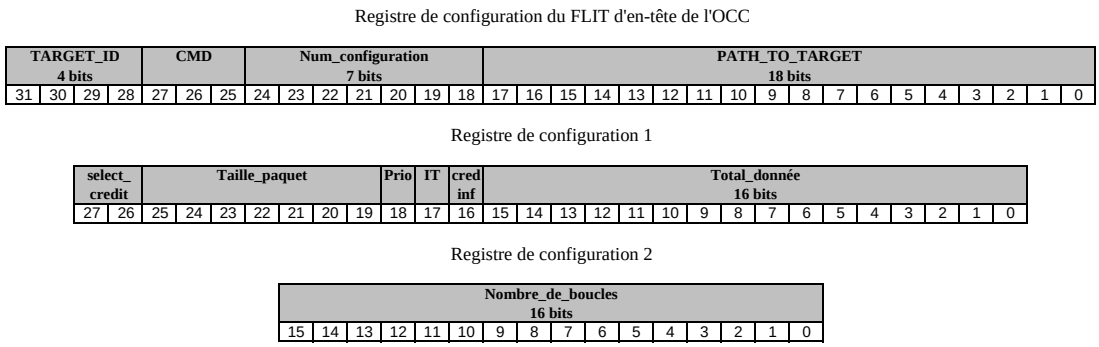

FIG. 3.9 – Registres de configuration de l'OCC

Par conséquent, la configuration de ce registre devra tenir compte des caractéristiques de la ressource de traitement (chaque unité de traitement ayant un comportement différent). Il faut souligner la nécessité d'avoir une égalité entre les champs *Total_ donnée* (ressource émettrice) et *Total_ credit* (ressource réceptrice) afin d'avoir une cohérence du montant total de données qui devra être échangé entre les deux parties. De la même manière que pour l'ICC, la taille des paquets de données est fixée par le champ *Taille_ paquet* (correspondant au seuil minimum de FLIT de données présents dans la FIFO à partir duquel on souhaite émettre un paquet de données). Cette taille de paquet ne doit pas être trop importante sous peine d'accroître les risques de blocage mais aussi la latence des communications. Généralement cette valeur est fixée à 8 mais nous verrons dans le chapitre 4 une étude portant sur l'impact de ces tailles de paquets sur les performances d'une application.

La destination des données est spécifiée par le champ *PATH_ TO_ TARGET* et le canal virtuel employé par le champ *prio*. Le champ *TARGET_ ID* correspond au numéro de la FIFO ciblée dans la ressource réceptrice (information prise en compte par le port d'entrée de la ressource ciblée lors du décodage du FLIT d'en-tête du paquet de données). L'émission de ces paquets de données est bien entendu conditionnée par la présence de crédits dans le compteur *select_ credit* à moins que le bit *cred_ inf* (crédits infinis) vaille 1. Ces crédits

sont ceux émis par la ressource émettrice lorsque le nombre d'éléments disponibles dans la FIFO dépasse la valeur contenue dans le champ *seuil_ credit* du registre de l'ICC.

Par conséquent, il y a autant de compteurs de crédits que de configurations possibles et valides pour chaque OCC. Ces compteurs de crédits sont distincts les uns des autres pour permettre de gérer simultanément plusieurs destinataires (différents flux de données) sans mélanger leurs crédits. Les paquets de données émis ont une commande *CMD* et un numéro de configuration *Num_ configuration* qui sont spécifiés dans le FLIT d'en-tête du paquet.

3.3 La gestion du flot de données dans FAUST

Nous avons vu ci-dessus qu'il existe différents éléments matériels permettant de paramétrer l'interface réseau en fonction des besoins de l'unité de traitement. Les différents registres de configurations des ICC et OCC permettent de spécifier les quantités de données consommées et produites par la ressource de traitement. Or au niveau des ressources de traitement, certaines peuvent recevoir différents flux de données en entrée et également émettre des données vers différentes ressources de calcul. Par conséquent, dans un contexte de réceptions ou d'émissions multiples de données, il est nécessaire d'attribuer une configuration par flux de données (quantité de donnée, chemin de routage, registre cible à la réception, ...). Nous allons donc détailler dans cette section les précautions à prendre pour pouvoir remplir les conditions imposées par l'application au niveau de l'enchaînement des communications dans FAUST.

3.3.1 Gestion en réception

Dans le cas d'une réception, chaque flot de données est contrôlé par l'ICC associé à la FIFO concernée. Le CFM permet de gérer deux configurations par ICC. Une commande d'initialisation provenant de l'ICC permet de signaler au CFM le besoin de charger un scénario de configuration. En effet, le CFM peut gérer deux configurations par ICC, mais offre aussi la possibilité d'enchaîner ces configurations, ces enchaînements pouvant s'effectuer de deux manières différentes.

La première possibilité d'enchaînement des configurations consiste en une exécution séquentielle des configurations sans répétition. Si les deux configurations sont validées, le CFM chargera en premier la configuration 1, puis la configuration 2. Cet enchaînement permet d'avoir deux flux de données bien distincts provenant de ressources différentes.

La deuxième possibilité d'enchaînement des configurations consiste en un enchaînement répétitif des configurations où le nombre d'itérations est fixé dans l'ICC. Dans ce cas, si les deux configurations sont validées, le CFM chargera en alternance la configuration 1, puis la configuration 2, et ce jusqu'à épuisement du nombre de boucles initialement prévu. Ainsi, pour un nombre de boucles fixé à 7, on aura le séquencement des configurations qui sera le suivant : 1 2 1 2 1 2 1.

Ce mécanisme de séquencement est particulièrement utile pour aller chercher des données alternativement à deux endroits différents en les réceptionnant dans une seule et même FIFO. Ceci est rendu possible au moyen du mécanisme de crédit qui permet de sélectionner l'origine des données. Nous verrons par la suite que nous nous servirons de ce séquencement pour pouvoir respecter les contraintes de la trame 4G de 4MORE où

l'on doit séquentiellement traiter des blocs de pilotes intercalés entre des blocs de données symétriques et de même taille. La figure 3.10 présente trois exemples de séquencement de configuration au niveau d'un ICC.

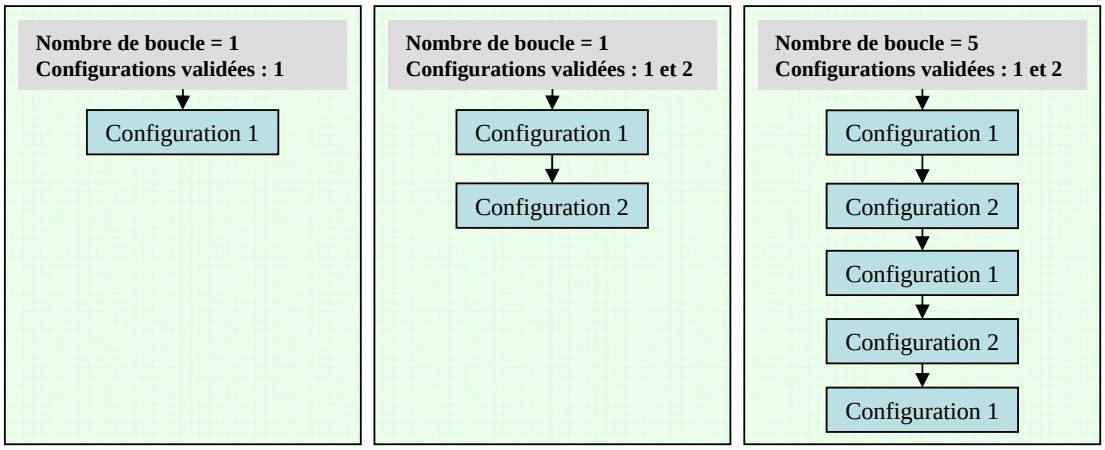

FIG. 3.10 – Scénarii de configurations pour les ICC

3.3.2 Gestion en émission

Dans le cas d'une émission, le flux de données produit par l'unité de traitement et mémorisé dans la ou les FIFO est géré directement par les contrôleurs de communication de sortie (OCC). L'OCC permet de gérer jusqu'à 4 configurations par FIFO, offrant la possibilité d'émettre des données vers quatre autres unités de traitement différentes. Ces configurations sont chargées dans le RWD et l'écriture d'un registre permet de les valider. Ici, contrairement à l'ICC, le séquencement des configurations n'est pas effectué par le CFM mais par les données elle-même (via le FLIT d'en-tête encapsulant le paquet de données). En effet, lorsqu'une configuration est chargée, celle-ci est exécutée jusqu'à l'envoi de toutes les données dont le montant total a été mentionné dans le champ *Total_donnée* du registre de configuration de l'OCC. Il est également possible de spécifier un nombre de boucles pour chaque configuration, mais celui-ci ne concerne qu'une seule et même configuration. Dans ce cas précis, une configuration ne sera terminée que si le nombre total de données a été envoyé, mais également si le nombre d'itérations a été effectué.

Comme nous l'avons vu dans la tableau 3.3, il existe 4 commandes différentes pour les paquets de données ou de crédits. La commande INIT_WRITE permet d'opérer des changements de configuration dans les OCC par l'intermédiaire du CFM. Ainsi, la réception d'un paquet de données comportant la commande INIT_WRITE associée avec le numéro de configuration (CONFIG_ID) dans son en-tête permet de spécifier au CFM le chargement d'une nouvelle configuration d'OCC. Si une commande INIT_WRITE est reçue lorsqu'une configuration dans l'OCC n'est pas terminée, le CFM va mettre en attente la commande pour la traiter par la suite. Cette gestion d'émission de donnée n'offre pas la possibilité de programmer le séquencement des configurations de chaque OCC contrairement à l'ICC. Ce séquencement va dépendre du découpage des blocs de données entrant en les associant avec la commande INIT_WRITE.

Nous allons maintenant voir de quelle façon l'ensemble de ces éléments matériels est paramétré au sein du réseau afin de configurer correctement chaque interface réseau de chaque unité de traitement constituant l'application.

3.3.3 Méthodologie de configuration du NoC

Nous l'avons vu dans les chapitres précédents, les réseaux sur puce offrent des performances supérieures à celles proposées par les topologies bus actuelles. En contre-partie, leur complexité de mise en œuvre est accrue compte-tenu des différents paramètres de dimensionnement et de spécification de l'architecture par rapport à l'application. Lorsque la phase d'exploration de l'espace des solutions architecturales est effectuée (cf. 4.2), le réseau est dimensionné avec des contraintes de placement des unités de traitement et des contraintes de chemins de communication. Ainsi, l'architecture est dimensionnée et caractérisée mais il est nécessaire de paramétrer correctement chacune des interfaces réseau des blocs fonctionnels de l'application.

Comme nous l'avons mentionné dans les sections 3.3.1 et 3.3.2, les registres des contrôleurs de communication doivent être initialisés afin que chaque unité de traitement puisse recevoir et émettre des données par l'intermédiaire de sa NI. Ces registres ont pour objectif de spécifier des contraintes de gestion de flux aux contrôleurs de communication pour être conforme au mode de commutation par paquets ("packet switching") ainsi qu'à la gestion de flux des communications de type "Wormhole". Ces paramètres d'initialisation prennent également en compte les spécifications de l'unité de traitement concernée, notamment son rendement en terme de consommation et de production de données.

L'ensemble de ces paramètres mémorisés dans les registres des ICC et des OCC de chaque NI doit être spécifié pour toutes les ressources. Cette ressource spécifique est constituée d'un processeur avec une mémoire associée dans laquelle se trouve toutes les instructions à envoyer à chaque NI du réseau. L'initialisation de ces registres et la validation des configurations jouées dans chaque NI se fait en 5 étapes. La figure 3.11 montre les différentes étapes successives permettant de configurer les NI des unités de traitement placées sur le NoC.

Les phases 1 et 2 permettent d'initialiser les registres des ICC et des OCC de chaque NI. Les fichiers de configuration des ICC et OCC sont lus et interprétés par le CPU et envoyés à la ressource concernée par le réseau. Chaque ressource est identifiée par un numéro sur la matrice 2D du réseau. Ce numéro d'identifiant est associé à un chemin de communication dans la matrice permettant au CPU d'envoyer ces paquets de configuration aux registres des unités de traitement concernées.

La phase 3 permet de spécifier le nombre de boucles que l'on souhaite faire au niveau de chaque ICC (cf. 3.3.1). Si rien n'est spécifié lors de cette phase, par défaut, le nombre de boucles est fixé à 1.

Pour finir, les phases 4 et 5 ont en charge de la validation des configurations à jouer en entrée et en sortie de chaque bloc de traitement. En effet, on peut configurer les registres d'un ICC pour deux configurations différentes mais ne choisir d'en utiliser qu'une seule pour l'application. Cette validation permet de sélectionner et valider les configurations et leurs enchaînements éventuels. Une fois ces deux étapes effectuées, le réseau devient autonome et les communications démarrent dès lors que des paquets de crédits parviennent aux émetteurs.

Les transactions peuvent démarrer et le réseau reste autonome jusqu'à épuisement du montant total de données et de crédits. Il peut être éventuellement stoppé par le processeur de contrôle dans les cas où une interruption est émise afin de changer un contexte de traitement par exemple.

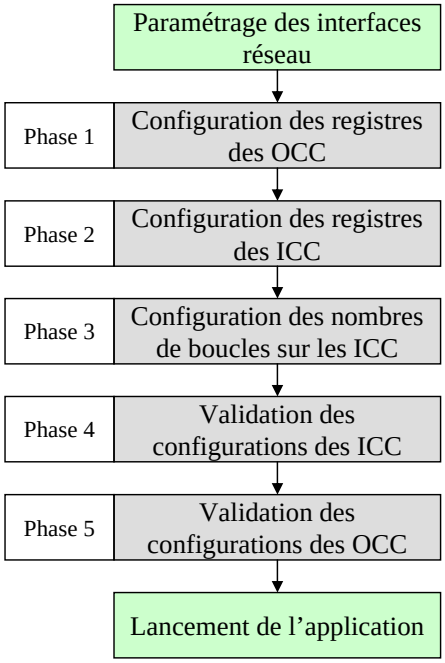

FIG. 3.11 – Flot de configuration des interfaces réseau du NoC

Le processeur (CPU) permet de configurer chaque NI du réseau pour initialiser les communications mais aussi pour faire de la supervision d'application. En effet, chaque bloc fonctionnel peut effectuer des interruptions pour executer une tâche particulière ou signaler une erreur éventuelle à l'OS gérant la couche physique.

Nous allons donc voir dans la section suivante comment le modèle SystemC permet de mettre en œuvre un réseau et son application associée. Le modèle et le flot d'implantation présentés sont ceux qui nous ont été livrés par le LETI. Nous verrons dans le chapitre 4 le flot de conception que nous avons modifié afin d'apporter des solutions de placement routage automatique (AAA) mais également une plus grande souplesse de mise œuvre pour les cas nécessitant des spécifications manuelles.

3.4 Le modèle TLM/SystemC du NoC FAUST

Ce modèle a pour but de réaliser des scénarii de simulations et d'évaluer les performances du réseau. Il permet donc de simuler des communications au niveau routeur mais également au niveau des interfaces réseau (lien indispensable pour pouvoir communiquer entre les unités de traitement). Le langage employé est le SystemC associé à la librairie TLM de STMicroelectronics.

Ce modèle a pour objectif de fournir les éléments de base au concepteur afin de réaliser un réseau avec une application basée sur les mécanismes de FAUST (packet switching, wormhole et BE). Ce modèle offre la possibilité au concepteur d'évaluer le réseau tout en ayant la garantie d'avoir les mêmes performances qu'une simulation en VHDL qui serait plus coûteuse en temps. Il est donc fortement utile d'avoir un outil d'aide à la conception basé sur ce modèle pour pouvoir aider le concepteur dans son implantation réelle sur silicium. Les réseaux étant complexes, réaliser des simulations VHDL afin de trouver la ou les solutions architecturales qui permettent de garantir les contraintes temporelles de l'application sont longues et fastidieuses.

Nous allons donc voir dans cette section les différents éléments de bases qui vont nous permettre de construire notre réseau ainsi que son application associée.

3.4.1 Modélisation du routeur

Le module *anoc_ node* (dérivé du *sc_ module* en SystemC) représente le routeur du réseau. Il contient des ports d'entrées *node_ in[nb_ ports]* et des ports de sorties *node_ out[nb_ ports]*. La valeur node_wait _state renseigne la notion de temps du routeur. Elle permet de caractériser le délai qui doit s'écouler entre deux transactions sur les ports de sorties, ce délai correspondant au temps de propagation pour passer d'un routeur à un autre. Cette valeur renseigne donc la fréquence à laquelle le réseau va opérer.

Le réseau opérant en transfert bi-directionnel entre routeurs, un protocole de synchronisation a été adopté. Ainsi les signaux d'acquittement de données data et accept sont modélisés par l'exécution d'une fonction transport unique définie dans l'API TLM/SystemC.

Enfin l'interconnexion des routeurs s'effectue au moyen du mécanisme de binding du SystemC. L'exemple ci-dessous montre une connexion bi-directionnelle ouest est :

nodeA->node_ out[WEST]->bind(nodeB->node_ in[EAST]) ;

3.4.2 Modélisation de l'interface réseau

Le module *anoc_ ni* représente la description en SystemC de l'interface réseau. Comme nous l'avons vu précédemment (cf. 3.2.5) l'interface réseau est composée de différents éléments de contrôle permettant de gérer les flux de données entrants et sortants destinés à l'unité de traitement. Nous avons également vu que l'échange des données entre l'interface réseau et l'unité de traitement s'effectuait au moyen de FIFO (*sc_ fifo*). Par conséquent, l'interface réseau est paramétrable afin de suivre les besoins de l'unité de traitement. La taille et le nombre de FIFO ainsi que le nombre de configurations disponibles sont renseignés au moyen d'un fichier de structure propre à la NI. Ce modèle implante aussi la fonction *transport* afin de réceptionner toutes les transactions et gérer les compteurs de crédits et de données dans les ICC et OCC.

3.4.3 Modélisation des unités de traitement

Le modèle d'unité de traitement a été développé afin de représenter les blocs fonctionnels connectés au routeur. Il exploite le modèle d'interface réseau *anoc_ ni* présenté ci-dessus (cf. 3.4.2).

Nous avons ici pour ce réseau trois modèles d'unité de traitement qui peuvent être soit un processeur contrôle (configuration du réseau), soit une mémoire ou soit une ressource de traitement. Nous allons voir brièvement chacune de ces ressources.

3.4.3.1 Modélisation du processeur de contrôle

Ce modèle *anoc_ressource_cpu* correspond à la ressource spécifique du réseau qui permet d'adminisitrer les routeurs du réseau et donc indirectement les ressources de traitement. Il a pour rôle d'interpréter les données des fichiers de configuration pour chaque ressource de traitement et de les envoyer aux interfaces réseaux. Ces fichiers de configuration sont les contenus des registres des ICC et des OCC de chaque NI. Une fois ces configurations envoyées, il envoie les commandes de validation des configurations qui vont être activée dans chaque NI en entrée et en sortie.

3.4.3.2 Modélisation de la mémoire

Le modèle *anoc_ressource_gen* représente les éléments mémoires qui exploitent le modèle d'interface réseau (*anoc_ni*). Ici il s'agit plus particulièrement de modéliser des générateurs de trafic pour le réseau. Ces générateurs de données sont programmables et offrent la possibilité de séquencer plusieurs configurations au niveau de l'interface réseau pout orienter les flux de données vers différents blocs fonctionnels du réseau. En effet, nous savons que la seule manière de changer une configuration d'OCC dans les NI était d'utiliser la commande INIT_WRITE. Or cette commande ne peut être exécutée dans les blocs de ressources de traitement pur. Par conséquent, ce sont les générateurs de trafic qui vont permettre d'envoyer cette commande lors des changements de destination des données.

3.4.3.3 Modélisation de la ressource de traitement

Le modèle *anoc_ressource_tb* représente les éléments de ressources de traitement pur. Ils exploitent également le modèle d'interface réseau (*anoc_ni*). L'unité de traitement est reliée à la NI au moyen de FIFO (*sc_fifo*). La synchronisation des données lues ou écrites dans ces FIFO s'effectue au moyen de lectures et d'écritures bloquantes. Le but recherché ici n'est pas de réaliser des traitements de calcul mais de modéliser le comportement du bloc fonctionnel, à savoir échanger des données, qui n'ont aucune signification.

Cette ressource est générique et est caractérisée par trois paramètres : la taille des messages en entrée et en sortie, et un temps de traitement spécifié au niveau cycle d'horloge. La structure de la modélisation de la ressource de traitement est bien celle présentée dans la section 2.2.2. La ressource de traitement a pour objectif de simuler le comportement de n'importe quel bloc fonctionnel matériel d'une application.

3.4.4 La configuration du NoC

L'ensemble des modèles présentés doivent respecter un flot de mise en œuvre qui permet de dimensionner et de paramétrer correctement le réseau et son application associée. La construction de la partie matérielle, à savoir les routeurs et leurs interconnexions pour former la matrice 2D du réseau peut s'effectuer en parallèle avec la construction de la partie logicielle.

On a donc une structure en arbre où les deux branches se rejoignent au moment du placement des blocs fonctionnels (CPU, RAM et unité de traitement) sur les routeurs de la matrice. Cette étape appelée aussi "mapping", est suivie du processus de configuration du réseau par le processeur spécifique dont la figure 3.11 présente le flot permettant cette configuration.

La figure 3.12 présente le flot de configuration du réseau mis en œuvre.

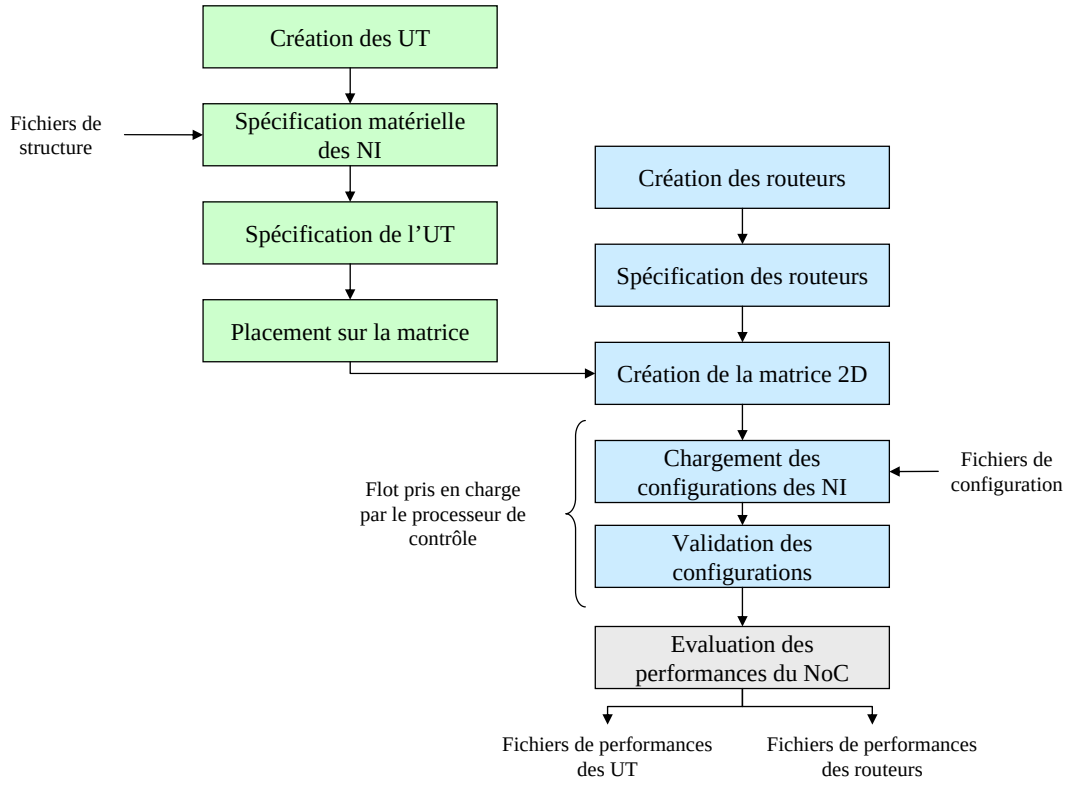

FIG. 3.12 – Flot de conception pour simulation SystemC TLM du NoC

Ce flot est celui qui a été proposé dans le modèle SystemC de FAUST. Nous verrons dans le chapitre 4 les modifications et contributions que nous avons apportées dans le cadre de ces travaux de thèse afin d'apporter plus de souplesse mais également la possibilité de réaliser une AAA automatique. On peut constater que ce flot possède des limites dans la description des modèles fonctionnels (routeur et unité de traitement) composant celui-ci où leur description est faite manuellement dans le code SystemC. Le nombre de routeurs ainsi que la dimension de la matrice sont également figés dans le code SystemC, et enfin, la phase de mapping est également faite manuellement dans le code SystemC.

Ce flot souffre d'un manque de flexibilité notamment dans les cas d'explorations architecturales multiples où l'on souhaite modifier rapidement la topologie et le mapping des unités de traitement. Sans fichier de contraintes et de description, il est nécessaire de modifier tous les fichiers sources et de recompiler l'ensemble avant de relancer une simulation. Ceci est d'autant plus vrai lorsque l'on est en limite de respect de contraintes temps

réel de l'application et que l'on souhaite affiner les paramètres de l'architecture. Le flot présenté ici peut donc devenir fastidieux, coûteux en temps et source d'erreurs.

3.4.5 Exemple d'une application simple

Nous présentons dans cette section un exemple simple d'un réseau avec son flot de mise en œuvre associé. Cette section vise à illustrer les mécanismes du modèle SystemC associé au flot de conception. Nous prenons comme exemple un réseau de topologie 2D de dimension 4×4 sur lequel sont connectées des IP numérotées de 1 à 13. Les autres ressources connectées aux routeurs sont occupées par le CPU en charge de la configuration et la gestion du réseau avec ses mémoires associées. La figure 3.13 montre la structure du NoC de cet exemple.

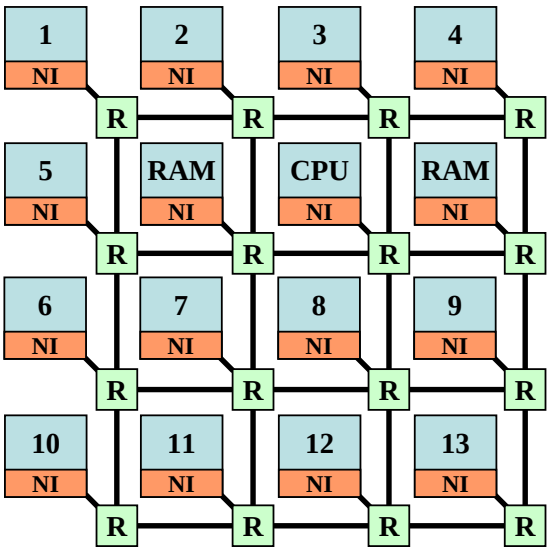

FIG. 3.13 – Exemple d'une application connectée à un NoC 4×4

Pour simplifier notre démarche de description du flot de conception pour la mise en place des simulations, nous allons nous focaliser sur les unités de traitement 2 et 3 qui pour l'exemple, ont une dépendance de données. Nous allons détailler le flot de conception du NoC uniquement pour ces deux éléments. Les caractéristiques de nos deux UT (Unité de Traitement) suivent le schéma de modélisation simplifié présenté précédemment, et leurs caractéristiques sont mentionnées dans le tableau 3.4 ci-dessous.

Numéro UT	Données entrantes	Nombre de cycles	Données sortantes
2	10	100	100
3	20	10	40

TAB. 3.4 – Caractéristiques des blocs fonctionnels de l'exemple

Comme nous l'avons vu dans le flot de conception, il est nécessaire de dimensionner les éléments matériels constituant la NI. De cette façon, nous avons associé un fichier de structure renseignant le nombre de FIFO en entrée et en sortie ainsi que leur taille et le nombre de configurations qui va être utilisé pour chaque ICC de chaque NI. A titre d'indication, les fichiers de structure associés à ces deux ressources (2 et 3) sont présentés sur la figure 3.14. Ces fichiers de structure de ces deux NI sont identiques. On aura donc une FIFO en entrée et une FIFO en sortie d'une capacité respective de 16 FLIT pour lesquelles on ne jouera qu'une seule configuration sur chaque ICC.

```
// this file is the structure file
// first line : nb_fifo_in, nb_fifo_out, nb_config
// second line : size_fifo_in
// third line size_fifo_out
1 1 1
16
16
```

FIG. 3.14 – Fichier de structure des ressources 2 et 3

Les étapes présentées ci-dessus constituent donc la phase de description matérielle. Une fois celle-ci effectuée, la phase de configuration logicielle est prise en charge par le processeur de contrôle qui va configurer les registres des éléments matériels de la NI. Il s'agit principalement de la configuration des registres des ICC et OCC et de les valider afin de pouvoir lancer les simulations.

Concernant le code SystemC, les paramètres de configuration des registres des ICC et des OCC sont spécifiés dans des fichiers de configuration. Ces fichiers de configuration sont lus et interprétés afin de les mettre en forme. Cette mise en forme est nécessaire afin de respecter le format des trames de configuration que le CPU va envoyer au travers du réseau. Un exemple de fichier de configuration est présenté sur la figure 3.15.

```
// the config file is composed as follows, for each configuration :
// first line : icc configuration  tot_credit/size_credit/tgtid/prio/maskIT
// second line : lgth_path/path 1/path 2/...
// third line : occ configuration tot_data/size_packet/tgtid/prio/cmd/configid/maskIT/sel_credit
//
// fourth line : lgth_path/path 1/path 2/...
48 8 1 1 0
2 SOUTH NORTH
720 8 1 1 INIT_WRITE 0 0 1
1
2 WEST EAST
```

FIG. 3.15 – Fichier de configuration

Par conséquent, lorsque le CPU a connaissance de l'ensemble des configurations de chaque unité de traitement connectée et utilisée dans le réseau, celui-ci envoie les données de configuration par le biais de commandes de configuration. Ces commandes doivent renseigner le numéro d'identification de la ressource ainsi que l'adresse des registres dans la NI. Il faut noter que le numéro d'identification de la ressource est associé à un fichier mentionnant le chemin d'accès des ressources dans la matrice par le CPU.

La figure 3.16 montre les commandes de chargement et validation des configurations des registres ICC et OCC d'unité de traitement. Ces commandes représentent successivement les étapes 1 à 5 du flot de configuration des interfaces réseau présenté précédemment sur la figure 3.11.

Les champs Adr_ICC et Adr_OCC renseignent les adresses des registres des ICC et des OCC dans la NI. Par conséquent, chacun des éléments matériels composant la NI sont donc accessibles par l'espace d'adressage. Celui-ci est présenté dans le tableau 3.17. Ainsi, les adresses allant de 0 à 128 sont réservées pour la configuration de la NI. Les adresses supérieures allant jusqu'à 256 sont laissées disponibles pour la configuration du bloc fonctionnel de traitement. En effet, certaines unités de traitement ont besoin de paramètres de configuration afin d'ajuster les traitements au contexte des données de l'application. Nous verrons dans le chapitre 5, lors de nos implantations matérielles, que ces espaces d'adressage seront utilisés pour configurer notre bloc fonctionnel générique.

En ce qui concerne notre exemple, les besoins de notre application ne requièrent qu'une seule configuration valide par ICC et par OCC. Par conséquent, les adresses passées en argument des fonctions de configuration utilisées par le CPU dans les phases 1 et 2 seront de l'adresse 16 pour Adr_ICC et l'adresse 64 pour Adr_OCC.

```
// Configuration des registres des ICC et OCC
write_config_icc(0,adr_ressource,Adr_ICC,&send_config_tb[i_TB]);
write_config(0,adr_ressource,Adr_OCC,&send_config_tb[i_TB]);

// Spécification du nombre de boucles sur les configurations de l'ICC
write_loop_icc(0, adr_ressource,8, Nb_loop_ICC);

// Validation des configurations des OCC et ICC
write_valid(0, adr_ressource , Valid_OCC );
write_go(0, adr_ressource, Valid_ICC);
```

FIG. 3.16 – Commande de configuration et de validation des registres des ICC et OCC d'une unité de traitement par le CPU de contrôle

Nous l'avons vu précédemment, il est possible de choisir l'ordre de séquencement des configurations dans le registre ICC de la NI. En fonction des besoins de l'application, si l'on souhaite spécifier le nombre de boucles à effectuer sur les configurations validées des ICC, le registre du CFM gérant l'ICC1, par exemple, sera accessible à l'adresse 8. Si aucune écriture n'est fait à cette adresse, par défaut, le nombre de boucles est fixé à 1.

Enfin, le lancement des simulations ne s'effectue que lorsque les configurations des ICC et des OCC ont été validées. Cette validation est représentée par les phases 4 et 5 du flot

	Address	TgtID	CfgID	Block		Content	Comment
	0					Not Used	
Common	1			ITM		ITM_Header	
	2			ITM		ITM Config	Prio et SRC_ID
	3			CFM		Go_CPU	ICC start
	4			CFM		CF_Valid	For OCC & CORE
	6-7					Not Used	
	8			CFM		NB boucle ICC1	
	9			CFM		NB boucle ICC2	
	10			CFM		NB boucle ICC3	reserved
	11			CFM		NB boucle ICC4	reserved
	12-15					Not Used	
ICC	16-17	1		Fifo1	ICC1	Header / Config1	
	18-19	1		Fifo1	ICC1	Header / Config2	
	20-21	1		Fifo1	ICC1	Header / Config3	
	22-23	1		Fifo1	ICC1	Header / Config4	
	24-25	2		Fifo2	ICC2	Header / Config1	
	26-27	2		Fifo2	ICC2	Header / Config2	
	28-29	2		Fifo2	ICC2	Header / Config3	
	30-31	2		Fifo2	ICC2	Header / Config4	
	32-33	3		Fifo3	ICC3	Header / Config1	
	34-35	3		Fifo3	ICC3	Header / Config2	
	36-37	3		Fifo3	ICC3	Header / Config3	
	38-39	3		Fifo3	ICC3	Header / Config4	
	40-41	4		Fifo4	ICC4	Header / Config1	
	42-43	4		Fifo4	ICC4	Header / Config2	
	44-45	4		Fifo4	ICC4	Header / Config3	
	46-47	4		Fifo4	ICC4	Header / Config4	
	48-63					Not used	
OCC	64-65-66		1	Fifo1	OCC1	Header / Config1 / N boucle	
	67-68-69		2	Fifo1	OCC1	Header / Config2 / N boucle	
	70-71-72		3	Fifo1	OCC1	Header / Config3 / N boucle	
	73-74-75		4	Fifo1	OCC1	Header / Config4 / N boucle	
	76-77-78		1	Fifo2	OCC2	Header / Config1 / N boucle	
	79-80-81		2	Fifo2	OCC2	Header / Config2 / N boucle	
	82-83-84		3	Fifo2	OCC2	Header / Config3 / N boucle	
	85-86-87		4	Fifo2	OCC2	Header / Config4 / N boucle	
	88-89-90		1	Fifo3	OCC3	Header / Config1 / N boucle	
	91-92-93		2	Fifo3	OCC3	Header / Config2 / N boucle	
	94-95-96		3	Fifo3	OCC3	Header / Config3 / N boucle	
	97-98-99		4	Fifo3	OCC3	Header / Config4 / N boucle	
	100-101-102		1	Fifo4	OCC4	Header / Config1 / N boucle	
	103-104-105		2	Fifo4	OCC4	Header / Config2 / N boucle	
	106-107-108		3	Fifo4	OCC4	Header / Config3 / N boucle	
	109-110-111		4	Fifo4	OCC4	Header / Config4 / N boucle	
	112-127					Not Used	
CORE	128		1	CORE		N mots	
	129						
	128 + N		2	CORE		N mots	
	129 + N						
	...						
	128 + 2N 255						

FIG. 3.17 – Espace d'adressage du NI

de configuration des interfaces réseau géré par le CPU. Cette validation de configuration est réalisée par un masquage de bit au sein du CFM. Rappelons que le CFM gère les chargements des configurations.

De ce fait, la validation des configurations des OCC se fera au moyen de la commande *write_valid* qui accédera au CFM par l'adresse 4. Et la validation des configurations des ICC s'effectuera à l'aide de la commande *write_go* qui accédera au CFM par l'adresse 3. Le mot de masquage de bit permettant de sélectionner les configurations à valider pour les ICC et les OCC sera calculé au moyen des tableaux 3.5 et 3.6.

FIFO 2		FIFO 1		
config 2	**config 1**	**config 2**	**config 1**	**validation**
0	0	0	1	config 1 sur FIFO 1
0	0	1	1	config 1 et 2 sur FIFO 1
0	1	0	1	config 1 sur FIFO 1 et 2
1	1	1	1	config 1 et 2 sur FIFO 1 et 2

TAB. 3.5 – Registre de validation des configurations des OCC pour une interface réseau du CFM

FIFO 2		FIFO 1		
config 2	**config 1**	**config 2**	**config 1**	**validation**
0	0	0	1	config 1 sur FIFO 1
0	0	1	1	config 1 et 2 sur FIFO 1
0	1	0	1	config 1 sur FIFO 1 et 2
1	1	1	1	config 1 et 2 sur FIFO 1 et 2

TAB. 3.6 – Registre de validation des configurations des ICC pour une interface réseau du CFM

A titre d'exemple, si l'on souhaite valider la configuration 1 de l'ICC1 et de l'ICC2, la valeur décimale correspondant au masquage de configuration sera 5 (0101 en binaire).

Nous avons donc présenté dans cette section les différentes phases qui composent le dimensionnement matériel et la configuration logicielle.

3.5 Conclusion

Nous avons vu dans ce chapitre les caractéristiques du réseau FAUST. Ce réseau est à gestion de flux de données de type "Wormhole" avec une seule qualité de service en meilleur effort. Rappelons que cette qualité de service n'offre aucune garantie sur le respect des contraintes temps réel imposées par l'application, mais offre par contre une meilleure exploitation des capacités des liens du réseau. Cette QoS offre la possibilité d'exploiter du mieux possible la bande passante théorique des liens.

Par conséquent, il est nécessaire de réaliser des simulations du NoC avec son application associée afin de dimensionner et paramétrer celui-ci afin de respecter les contraintes de l'application. Ainsi, pour pouvoir réaliser ces simulations à haut niveau, nous avons

présenté le modèle SystemC TLM du NoC FAUST. Ce modèle propose un flot de conception pour ces simulations qui est l'image de celui qui serait à adopter lors d'implantations physiques. Plusieurs paramètres entrent en ligne de compte dans la configuration et l'initialisation des différents registres des NI. Ces simulations haut niveau ont pour but de guider le concepteur dans ses choix architecturaux (topologie, nombre de routeurs, mapping, tailles mémoires, profondeur de FIFO, ...) et ces choix de configuration logicielle (chemin de routage, taille des paquets, ...). La multitude de choix parmi ces paramètres montre l'intérêt de posséder un outil d'aide à la conception de haut niveau.

Le flot de conception employé dans le modèle SystemC de FAUST possède des inconvénients. La modification des différents paramètres ayant un impact direct sur les performances globales du réseau étant nécessaire dans la phase d'exploration des solutions architecturales, il faut donc avoir un flot le plus souple possible. Or ce n'est pas le cas dans celui présenté dans ce chapitre. En effet, la modification de la topologie du réseau ainsi que les contraintes de placement des unités de traitement sur le réseau n'est pas automatisé. De plus, les simulations du réseau nécessitent un ajustement de ces critères à plusieurs reprises dans l'objectif de remplir le cahier des charges fixé par l'application.

C'est pour ces raisons que nous allons voir dans le chapitre suivant le flot de conception que nous avons mis en œuvre afin de proposer un outil de conception de ce NoC complètement automatique. Le flot de conception présenté dans le chapitre suivant montre les contributions des travaux qui ont été réalisées durant cette thèse.

Chapitre 4

Méthodologie de conception mise en œuvre pour la simulation de NoC

Sommaire

L'objectif de ce chapitre est de décrire le flot de conception que nous avons mis en œuvre dans notre outil d'aide à la conception de NoC. Cet outil a été développé autour du NoC FAUST mais il est également envisagé de donner la possibilité à d'autres NoC de pouvoir s'interfacer à l'outil. Celui-ci permet de réaliser des simulations d'une application en générant une architecture matérielle en SystemC TLM du NoC pour une application donnée.

L'outil proposé permet deux modes de fonctionnement, l'un complètement automatique, l'autre semi-automatique. Dans le mode automatique, l'application et l'architecture sont décrites sous forme de graphe pour lequel on réalise une AAA. A l'issue de cette phase, des contraintes d'architecture et de placement des éléments fonctionnels du graphe d'application sont renseignées dans des fichiers qui sont ensuite pris en compte dans la génération du flot de simulation sur le NoC. Cette deuxième phase du flot de conception prend différents paramètres permettant d'optimiser les performances du réseau sous forme de fichiers. Ces paramètres pourront être ajustés manuellement par la suite pour pouvoir réaliser des optimisations de performance sur l'application.

Le mode semi-automatique permet au concepteur de spécifier manuellement les contraintes architecturales voulues ainsi que les contraintes de l'application et son placement associé sur la topologie du NoC. Nous le verrons dans ce chapitre, cette phase s'est avérée intéressante dans le cadre de nos contributions dans le projet Européen 4MORE, où les simulations portaient sur des aspects mono et multi-composants avec des contraintes de placement, cas où l'AAA est difficile de mise en œuvre notamment en ce qui concerne les restrictions d'entrée/sortie et les chemins de routage entre les différents composants qui caractérisent l'architecture matérielle visée.

4.1 Introduction

Nous l'avons vu dans le chapitre 1, les NoC offrent de belles perspectives pour pallier aux limites des solutions de communications à base de bus dans les SoC actuels. Or ces paramètres sont nombreux et impliquent une difficulté de mise en œuvre accrue comparativement aux bus. L'espace des solutions architecturales devient important et il en est de même concernant la configuration des paramètres logiciels du réseau (chemins de routage, taille des paquets, ...).

Compte tenu de cette complexité de mise en œuvre, il est donc nécessaire d'avoir un outil d'aide à la conception pour les NoC. Cet outil peut être automatique ou semi-automatique afin de permettre au concepteur d'avoir la possibilité de forcer des contraintes dans le flot de conception.

De plus, cet outil permet de réaliser des simulations à haut niveau en SystemC TLM afin d'évaluer les performances de l'application placée/routée sur la matrice du NoC. Lorsque ces résultats de simulations garantissent le respect des contraintes de l'application, la phase d'implantation matérielle peut être réalisée. Certains flots de conception comme celui de Ætheral [36] ou encore celui de Arteris [84], génèrent le code VHDL associé afin de pouvoir réaliser directement l'implantation matérielle lorsque les résultats de simulation le permettent.

Or l'implantation matérielle des NoC est plus complexe de réalisation car il faut respecter les contraintes de topologie de la structure du réseau en fonction des contraintes temporelles (fréquence de fonctionnement du réseau). De plus, les blocs fonctionnels connectés au réseau ont des coûts en surface différents les uns des autres souvent liés à leur complexité algorithmique. Un bloc FFT par exemple, n'a pas le même coût en surface de silicium qu'un bloc de CBS. Cette contrainte de surface induite par les blocs fonctionnels rajoute un critère de contrainte plus ou moins fort dans les phases de synthèse et de placement routage qui caractérise une implantation matérielle sur FPGA par exemple. Les implantations matérielles sont souvent coûteuses en temps compte tenu de la spécificité de la cible architecturale voulue (ASIC ou FPGA). Mais dans le cas des NoC, cette étape peut s'avérer être encore plus longue du fait des contraintes de respect de la régularité de la topologie du NoC et de la distribution des horloges aux différents blocs fonctionnels.

On trouve dans la littérature plusieurs articles abordant ce problème d'implantation [28, 60, 85, 40, 27, 86, 87] où justement le placement des blocs fonctionnels sur la topologie du NoC devient un problème critique à part entière. Parmi les solutions les plus couramment rencontrées, une consiste à empêcher la connexion de bloc fonctionnel aux routeurs avoisinant un bloc fonctionnel à fort besoin en surface de silicium. La figure 4.1 montre une possibilité d'implantation d'un gros bloc fonctionnel tout en conservant la régularité

de la topologie du NoC. La ressource 14 a un coût en surface important comparativement aux autres (Figure 4.1 (a)). Ainsi, les liens vers les ressources 8, 9 et 13 sont détruits afin de pouvoir placer l'unité 14 sans avoir une irrégularité de topologie. Bien entendu, il s'agit ici d'un exemple, les ressources supprimées devront être reconnectées au réseau si elles font partie intégrante de l'application. Dans ce cas, il faudra accroître la matrice de routeurs du NoC.

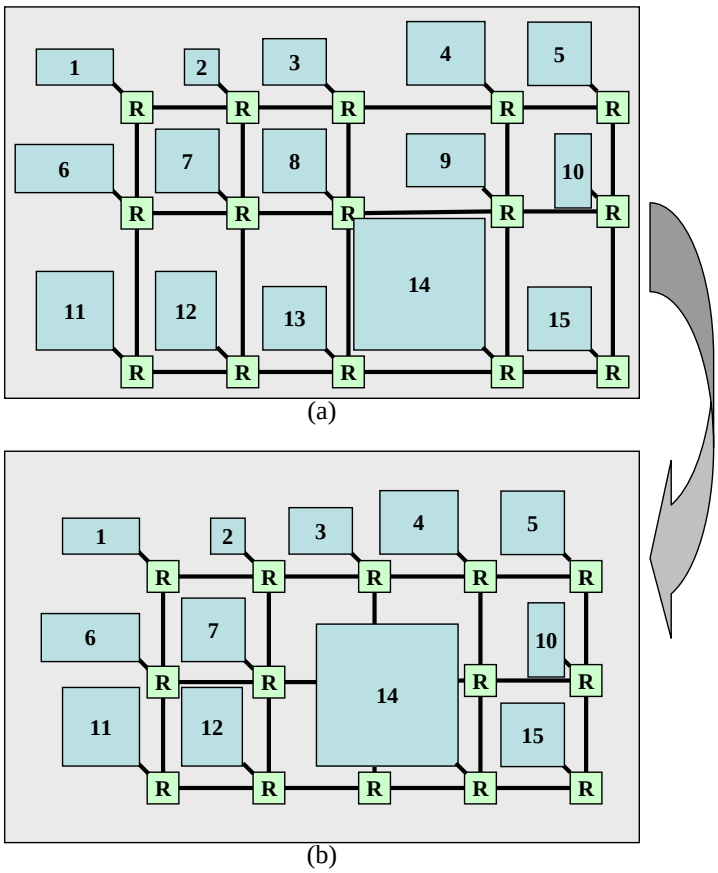

FIG. 4.1 – Solution de placement des blocs fonctionnels pour implantation matérielle : (a) topologie déformée , (b) topologie régulière avec 3 routeurs non disponibles pour connecter un bloc fonctionnel

Notre flot offre la possibilité de réaliser des implantations matérielles lorsque la phase de validation des performances de l'application par simulation est terminée. Pour l'instant, la génération du code n'est pas réalisée de manière automatique. Cette génération est actuellement effectuée manuellement pour une matrice de routeurs de faible dimension en vue d'être implantée sur FPGA. Nous le verrons dans le chapitre 5, nous avons mis en place une solution architecturale de modélisation des blocs fonctionnels permettant de s'affranchir du problème de surface de silicium requise mais aussi de la disponibilité ou non du bloc algorithmique en VHDL. De plus, l'intégration des véritables blocs fonction-

nels implique une simulation fonctionnelle en VHDL afin de valider les algorithmes avant implantation.

Nous envisageons d'utiliser un formalisme de description en langage XML pour générer automatiquement la topologie réseau en sources VHDL associées à la solution architecturale de modélisation des blocs de traitement que nous proposons. Ces blocs fonctionnels sont entièrement génériques et paramétrables logiciellement par le biais de commandes envoyées par le processeur de contrôle. Cette partie implantation sera détaillée plus amplement dans le chapitre 5.

Nous allons donc voir dans un premier temps le flot complet de conception que nous avons mis en œuvre. Ensuite, nous verrons quels critères et quelle méthode nous avons mis en place dans la phase AAA. Puis nous verrons la phase de génération du NoC en SystemC avec les paramètres d'entrée qui peuvent être issus de la phase AAA ou bien spécifiés manuellement par l'utilisateur. Pour finir, nous verrons de quelle manière nous exploitons les données issues des simulations afin de valider ou non les contraintes de l'application.

4.2 Présentation du flot de conception mis en œuvre

Comme nous l'avons vu dans le chapitre précédent (chapitre 3), le flot de conception mis en place dans les sources SystemC du NoC FAUST possède des limites en terme de souplesse de mise en œuvre (modification de la topologie du réseau, modifications des contraintes de placement des unités de traitement sur le réseau non automatisées).

Nous allons donc présenter dans cette section les contributions apportées durant ces travaux de thèse afin d'améliorer ce flot de conception et d'offrir la possibilité de réaliser une AAA à plus haut niveau d'abstraction en décrivant l'application et l'architecture sous forme de graphe.

L'outil développé offre la possibilité de spécifier l'application et l'architecture sous forme de graphe afin de trouver une solution d'adéquation de l'algorithme avec notre architecture NoC. Cette partie de notre flot parcourt l'espace des solutions du graphe au moyen d'un algorithme glouton que nous détaillerons dans la section 4.3. Cet algorithme est spécialisé à notre outil de conception de NoC dans le sens où plusieurs critères sont à prendre en compte conjointement pour le NoC. A l'issue de cette phase d'AAA, nous obtenons des critères de contrainte sur l'architecture (Taille des FIFO des NI, nombre de FIFO, topologie du NoC et contrainte de placement des UT de l'application sur la matrice) mais également des contraintes pour l'application (chemin de routage, quantité de données et de crédits en entrée et en sortie de chaque UT, commande de validation des UT pour le CPU).

Ces deux contraintes sont prises en considération dans la phase de génération du NoC. En effet, la modélisation en SystemC du NoC requiert plusieurs paramètres pour construire le réseau et valider l'application sur celui-ci. Nous le verrons dans la section 4.4, nous avons mis en place un élément de routage automatique de la matrice du NoC qui permet de réaliser des simulations dans un contexte mono-composant mais également dans un contexte multi-composants.

Ce critère multi-composant n'étant pour l'instant pas géré par la phase AAA, nous avons mis en place dans cet outil la possibilité de spécifier manuellement les contraintes de l'application sur l'architecture. Ainsi, dans notre flot, il est donné la possibilité à l'uti-

lisateur de renseigner toutes ses contraintes architecturales et applicatives dans un fichier Excel pour lequel une macro sera exécutée afin de générer automatiquement les fichiers de contraintes. Nous verrons plus en détail cette étape dans la section 4.4.3. Enfin, lorsque les simulations sont terminées, l'outil fournit deux informations de performance :

- des fichiers de performance des routeurs renseignant les variations de bande passante sur chaque lien du routeur et ce pour la durée de la simulation
- des fichiers de performance pour chaque UT renseignant les latences des données en entrée et en sortie ainsi que le temps de traitement des données

Ces données de simulation renseignent sur le respect ou non des contraintes temps réel de l'application. Mais elles donnent également une indication sur les critères matériel et/ou logiciel qui peuvent être modifiés afin de pouvoir respecter les contraintes de l'application. Enfin, nous prévoyons d'utiliser le formalisme de description XML afin de générer automatiquement les codes VHDL pour réaliser une implantation sur FPGA. Nous présentons dans le chapitre 5 des résultats d'implantation sur plate-forme à base de FPGA Xilinx Virtex4 pour lesquels les fichiers de description des ressources matérielle ont été spécifiés manuellement.

L'ensemble des différents éléments de notre flot de conception mis en place dans notre outil d'aide à la conception de NoC est présenté sur la figure 4.2.

Nous allons donc voir dans les sections suivantes chacun des différents éléments principaux que nous avons mis en place dans notre flot.

4.3 La Phase d'Adéquation Algorithme Architecture (AAA)

Nous l'avons vu précédemment, la phase AAA consiste à réaliser l'adéquation de l'algorithme spécifié par l'utilisateur sur une architecture ayant comme média de communication un NoC. Cette phase nécessite donc une description de l'application et de l'architecture sous forme de graphe. Ensuite, l'AAA s'effectue au moyen d'une méthode heuristique d'optimisation basée sur un algorithme glouton. Cette branche de notre flot de conception représente la partie dite "automatique" qui permet de générer automatiquement les différents fichiers de description des ressources matérielle de l'architecture (topologie, contraintes de placement) mais également les fichiers de configuration logicielle des éléments encapsulant les unités de traitement placées sur l'architecture (configuration des NI).

L'espace des solutions possibles nécessitent l'exploration d'un grand nombre de cas, or le choix d'une exploration au niveau algorithme n'est pas envisageable. En effet, une exploration au niveau algorithmique dans le cas d'un NoC nécessiterait un temps d'exploration trop long du fait des nombreux critères à prendre en compte engendrant de nombreuses combinaisons possibles. C'est pour cette raison que nous avons choisi d'utiliser une méthode basée sur une heuristique. On réduit la complexité moyenne en examinant d'abord les cas qui ont le plus de chance d'être une solution.

Nous allons donc voir dans un premier temps dans cette section le formalisme de description des graphes d'application et d'architecture. Ensuite, nous verrons la formulation du problème qui nous a permis de mettre en œuvre l'heuristique de l'algorithme des parcours de graphe pour l'AAA.

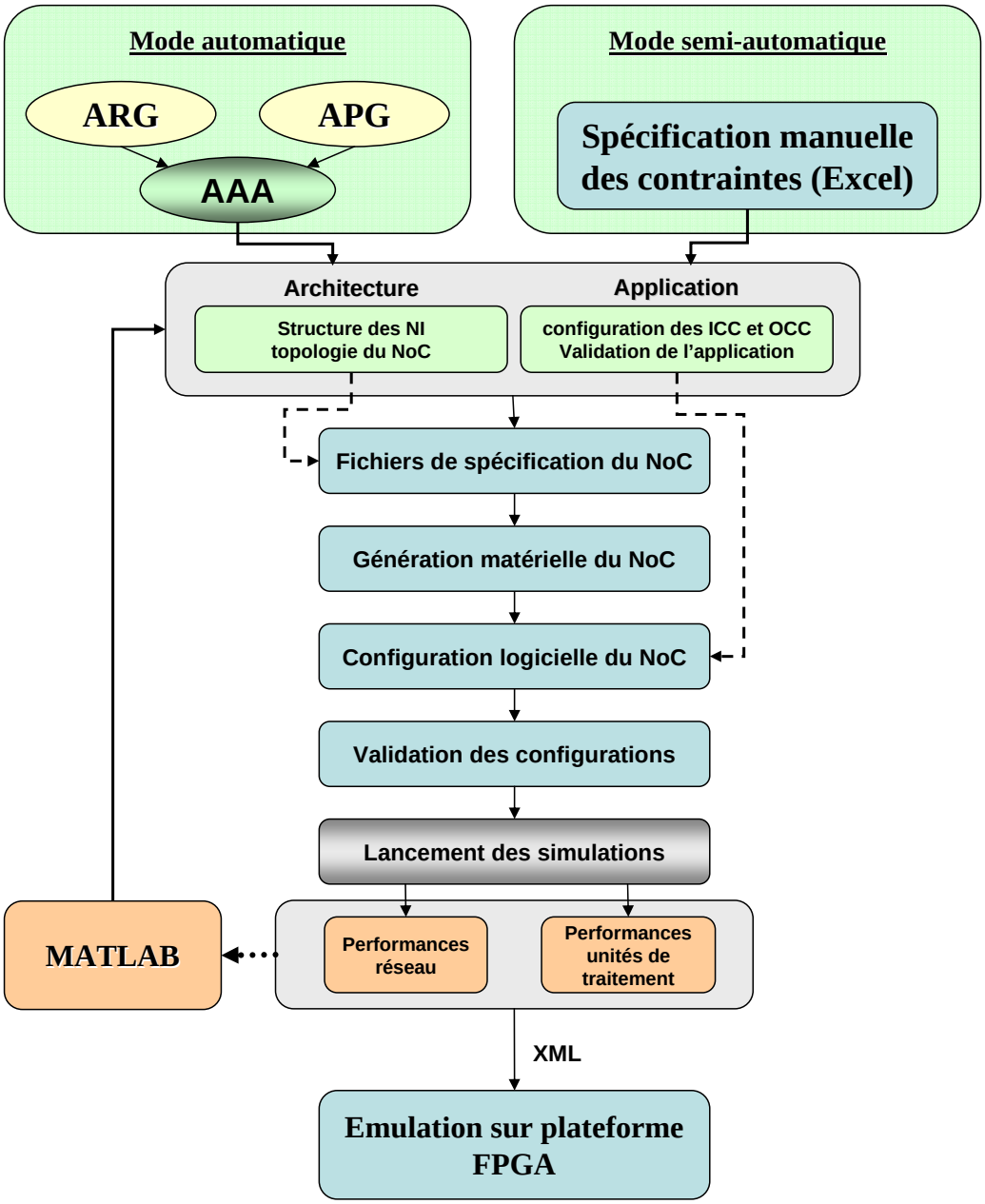

FIG. 4.2 – Flot de conception

4.3.1 Modélisation des graphes d'application et d'architecture

La modélisation sous forme de graphe de l'application et de l'architecture a pour objectif de décrire brièvement les contraintes de l'application ainsi que celles de l'architecture afin de donner des informations à l'algorithme d'optimisation de graphe utilisé pour l'AAA. En effet, une AAA sur NoC doit prendre en considération différents paramètres au niveau de l'application et de l'architecture pour que l'optimisation de l'application sur l'architecture soit optimale compte tenu de la spécificité de l'architecture de communication.

4.3.1.1 Le graphe d'application

Le graphe d'application permet au concepteur de renseigner les caractéristiques de l'application. Ces paramètres indiquent la notion de dépendance de données entre chaque cœur du graphe d'application. Le NoC FAUST étant en QoS BE, il est nécessaire d'avoir une information de contrainte de bande passante et de quantité de données. En effet, la QoS de type BE ne garantissant pas de bande passante entre blocs fonctionnels dépendants de données, il est nécessaire de connaître les contraintes de bande passante entre chaque cœur de l'application. Cette bande passante est calculée par rapport au montant de données à transférer en fonction des contraintes temporelles de l'application. Par conséquent les indications de bande passante sont des valeurs minimales permettant de garantir les contraintes temps réel de l'application, ceci sans la notion de latence qui va nécessairement être introduite par le réseau lui-même.

La figure 4.3 montre un exemple de graphe d'application.

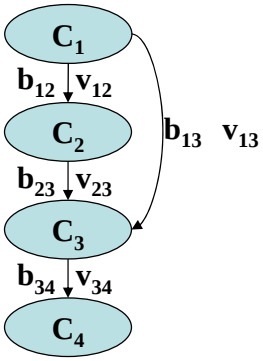

FIG. 4.3 – Exemple d'un graphe d'application (APG)

Les arcs de pondération entre chaque cœur du graphe renseignent les besoins en bande passante ainsi que le volume des communications entre les deux vertex. Cette pondération de besoin en bande passante est calculée par rapport aux contraintes temps réel de l'application. On peut donc définir le graphe d'application de la manière suivante :

Définition **:** Un graphe d'application (APG) est défini tel que APG = G(C,A) est un graphe maître où chaque vertex c_i représente un bloc fonctionnel sélectionné, et chaque arc directionnel $a_{i,j}$ représente une communication du vertex c_i au vertex c_j. L'arc $a_{i,j}$ est caractérisé par deux quantités :

- $v(a_{i,j})$: arc de volume du vertex c_i au c_j, qui représente le volume de la communication(bits) de c_i à c_j.
- $b(a_{i,j})$: arc de bande passante requise du vertex c_i au c_j, qui représente la bande passante minimale nécessaire(bits/sec.) entre c_i et c_j qui doit être réservée lors de la communication dans le réseau.

Le paramètre $v(a_{i,j})$ n'est pas utilisé dans l'AAA en elle-même, mais il est pris en compte à la fin de celle-ci lorsque l'on souhaite générer les fichiers de configuration. En effet, les fichiers de configuration de l'application (configuration des NI notamment) nécessitent les informations de montant total de données et de crédits pour les récepteurs et émetteurs respectivement.

Dans l'optique de générer les fichiers de configuration pour paramétrer le modèle SystemC du NoC après adéquation, les vertex c_i possèdent les informations du modèle équivalent simplifié des UT défini et présenté précédemment (cf. chapitre 2 section 2.2.2).

4.3.1.2 Le graphe d'architecture

Le graphe d'architecture permet au concepteur de renseigner les caractéristiques de l'architecture du NoC. Il a pour objectif de modéliser la topologie du réseau en apportant une notion de dépendance physique entre les routeurs (r_i). Celle-ci est modélisée par les liens ($p_{i,j}$) entre routeurs pour lesquels est spécifié une bande passante théorique maximale directement liée à la fréquence de fonctionnement du NoC. Ainsi, le graphe d'architecture peut être défini de la manière suivante :

Définition **:** Un graphe d'architecture (ARG) ARG = G(R,P) est un graphe maître ou chaque vertex r_i représente un routeur de l'architecture, et chaque arc directionnel $p_{i,j}$ représente le chemin de routage entre deux routeurs r_i et r_j. L'arc directionnel $p_{i,j}$ est caractérisé par deux quantités :

- $L(p_{i,j})$: arc de chemin du routeur r_i au r_j, qui représente les différents liens constituant le chemin le plus court pour aller de r_i à r_j
- $B(p_{i,j})$: arc de bande passante disponible du routeur r_i vers r_j, qui représente la bande passante disponible entre r_i et r_j

Un exemple de modélisation simple d'une topologie réseau sous forme d'ARG est présenté sur la figure 4.4.

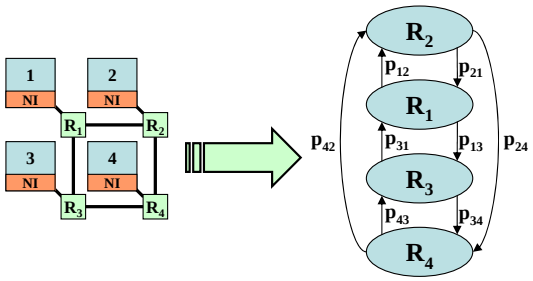

FIG. 4.4 – Exemple d'un graphe d'architecture (ARG)

Cette modélisation permet, lorsque l'adéquation est effectuée, de connaître les chemins des données et des crédits entre chaque UT dépendante de données placées sur un routeur. En effet, l'arc $L(p_{i,j})$ représente l'un des chemins les plus courts (dans les cas où il y en a plusieurs) entre deux routeurs, minimisant théoriquement la latence des communications.

Après avoir défini les modèles de représentation des graphes d'architecture et d'application, nous allons voir comment ceux-ci sont intégrés dans la phase AAA.

4.3.2 La phase AAA

La phase AAA consiste à placer l'application spécifiée sous forme de graphe sur l'architecture également décrite sous forme de graphe tout en considérant les contraintes de placement induite par le média de communication qui est le NoC.

4.3.2.1 Les contraintes de placement des UT sur un NoC

Le principe de l'adéquation de l'application sur une structure de type NoC consiste à placer chacun des vertex du graphe d'application sur un seul et unique routeur de la structure du réseau tout en respectant les contraintes de performances de l'application. La figure 4.5 illustre un exemple d'adéquation.

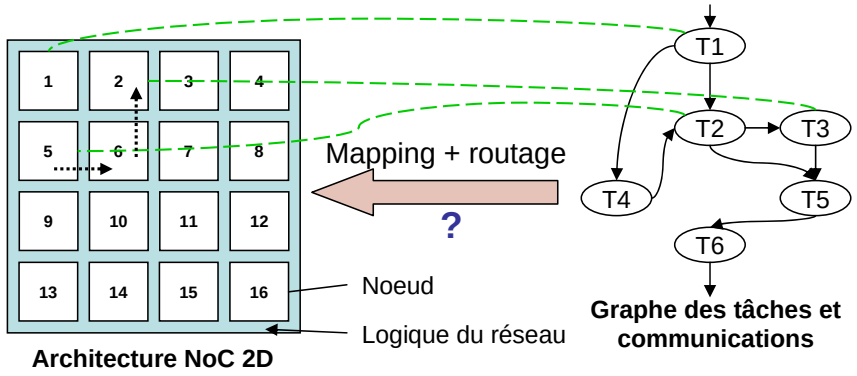

FIG. 4.5 – Exemple de mapping et routage d'un graphe d'application de taches

Ainsi, la technique la plus intuitive et la plus rapide consisterait à placer les IP de façon à ce que les chemins des communications qu'ils ont entre eux soient les plus courts possibles. Cette approche est envisageable lorsque l'on a un NoC en QoS de type GT où un ordonnancement des communications est effectué au préalable permettant de réserver des slots de temps pour chacune des communications. Cet ordonnancement est mentionné au sein des routeurs ou NI au moyen de tables d'allocation. Cette approche est celle employée dans le NoC Ætheral avec la technique UMARS [61, 63, 62, 31].

Celle-ci met en œuvre un placement et un routage en prenant en compte le respect des bandes passantes entre les ressources. Cette technique dite TDMA, apporte une QoS permettant de garantir des communications avec des performances maintenues sans perte de données, avec une bande passante minimum et une latence maximum. Ainsi, le NoC et les blocs IP peuvent être découplés ce qui signifie que le comportement d'une IP lors

de communications ne peut pas influer celui des autres IP présentes sur le même NoC. Les blocs IP peuvent être modélisés et validés indépendamment les uns des autres. En contrepartie, l'utilisation des liens de communication au sein du réseau n'est pas optimale et la complexité de l'interface réseau des UT ainsi que celle des routeurs se voit augmentée.

Or, dans le cas d'une QoS de type BE (cas du NoC FAUST utilisé), aucun ordonnancement des communications n'est effectué ce qui a pour effet de ne pouvoir garantir des contraintes de bande passante pour l'ensemble des communications du réseau. Ainsi, la technique consistant à placer les IP en ayant les chemins de communication les plus courts possibles n'est pas suffisante. En effet, les chemins de communication peuvent emprunter des liens communs engendrant des contraintes de bande passante plus fortes pour chaque lien physique. Par conséquent, les risques de violation de bande passante théorique pour chaque lien sont accrus.

Dans [33, 34], le placement des UT et le routage des communications sont réalisés conjointement afin d'avoir les chemins de communication les plus courts possibles tout en tenant compte au fur et à mesure du placement de la bande passante requise sur les liens par rapport à sa bande passante théorique. Si le routage final requiert une bande passante plus grande que celle disponible sur un lien, alors le placement et le routage sont réitérés afin de trouver une solution remplissant les conditions. De plus, cette technique prend également une contrainte de coût en énergie visant à minimiser le coût en consommation des communications à travers le réseau. Ce coût étant estimé par le volume de bits transmis et leur coût de passage dans un lien et dans un routeur. Une notion d'ordonnancement des tâches a été introduite dans [35].

Une amélioration a été apportée à la méthode UMARS en introduisant la technique NMAP présentée dans [42, 67, 37, 38]. Cette technique offre la possibilité de permettre à une ressource d'émettre ses données et/ou crédits en multi-trajets vers une autre ressource. La méthode de routage employée avant l'introduction de cette technique était de type XY. Ce routage peut avoir ses limites dans certains cas. Cette technique permet donc une meilleure utilisation des liens de communication. Mais en contrepartie, elle rend plus comlexe l'architecture des routeurs et des NI.

Dans ce contexte, et compte tenu des contraintes du NoC employé, notre approche est basée sur les travaux [33, 34]. Elle consiste à placer les tâches du graphe d'application sur les routeurs du réseau de façon à ce que les chemins de communication soient les plus courts tout en tenant compte du respect de la limite de bande passante théorique imposée par les liens entre routeurs. Cette bande passante est fonction de la fréquence de fonctionnement du réseau. Nous allons voir dans la section suivante la formulation du problème de notre méthode AAA.

4.3.2.2 Formulation du problème

La méthode AAA mise en place ici est basée sur une heuristique (technique consistant à apprendre petit à petit, en tenant compte de ce que l'on a fait précédemment pour tendre vers la solution d'un problème sans garantie d'arriver à une solution quelconque en un temps fini) pour les raisons énoncées précédemment.

Nous avons mis en place une méthode qui va placer les vertex du graphe d'application sur les vertex du graphe d'architecture en ayant pour condition de minimiser les chemins de communication tout en veillant à ne pas violer la limite de bande passante maximale

théorique de chaque lien du NoC. Cette limite de bande passante maximale est directement liée à la fréquence de fonctionnement du NoC (mentionné dans le graphe d'architecture).

Ainsi, la formulation du problème de l'AAA prise en charge par l'heuristique que nous avons mis en œuvre dans notre flot de conception peut être énoncée de la manière suivante :

$$Taille(APG) \leq Taille(ARG) \tag{4.1}$$

La fonction de placement *map()* des UT (APG) sur l'architecture (ARG) est donc définie selon les critères suivants :

$$\forall c_i \in C,\ map(c_i) \in R \tag{4.2}$$

$$\forall c_i \neq c_j \in C,\ map(c_i) \neq\ map(c_j) \tag{4.3}$$

$$B(l_k) \geq \sum_{\forall a_{i,j}} b(a_{i,j}) \times f(l_k, p_{(map(c_i),map(c_j))}) \tag{4.4}$$

Où $B(l_k)$ est la bande passante maximale sur un lien l_k tel que :

$$f(l_k, p_{m,n}) = \{ \begin{array}{l} 0 : l_k \notin L(p_{m,n}) \\ 1 : l_k \in L(p_{m,n}) \end{array} \tag{4.5}$$

La condition 4.1 permet de vérifier si le nombre de routeurs disponibles sur l'architecture est supérieur ou égal au nombre de blocs fonctionnels du graphe d'architecture avant de lancer l'AAA. Cette condition est importante car notre graphe d'application permet au concepteur de spécifier si des routeurs sont réservés (cas du processeur de contrôle du NoC) et par conséquent indisponibles pour la connection d'un bloc fonctionnel.

Les conditions (4.2) et (4.3) définissent la fonction de placement pour laquelle un routeur ne peut recevoir qu'un seul et unique bloc fonctionnel sur son 5^{e} port.

La condition (4.4), permet de remplir la condition énoncée précédemment qui consiste à vérifier si la charge de chaque lien dans l'architecture cible n'est pas supérieure à la bande passante théorique disponible. Cette vérification est effectuée à chaque fois que l'on cherche à placer une nouvelle tâche de l'APG.

Pour finir, la fonction définie dans (4.5) renvoie une valeur vraie ou fausse selon que le (ou les) chemin de communication entre les routeurs c_i et c_j contient ou non le lien l_k.

De cette manière, nous avons décrit la façon dont notre méthode AAA va placer les UT sur notre architecture avec les contraintes imposées par celle-ci (QoS de type BE). Nous allons donc voir dans la section suivante de quelle manière les résultats de cet adéquation sont interprétés pour générer le modèle SystemC du NoC. Ces paramètres d'entrée de la modélisation SystemC du NoC sont également ceux générés par la branche dite "semi-automatique" de notre flot. Dans ce mode "semi-automatique" les contraintes de l'application sur l'architecture sont spécifiées directement dans des feuilles d'un classeur Excel dont les informations qu'elles contiennent sont extraites par une macro qui génère l'ensemble des fichiers nécessaires à la génération du modèle SystemC du NoC.

4.4 Génération du modèle SystemC du NoC

Nous allons aborder dans cette section les modifications apportées au flot de conception dans la génération du modèle SystemC. Cette contribution a permis d'apporter une plus

grande souplesse et une plus grande flexibilité dans la création du modèle ainsi que pour les phases d'optimisation des performances. Les différents paramètres étaient auparavant renseignés manuellement directement dans le code source du NoC FAUST fournit par le CEA LETI. Cette méthode s'est avérée peu adaptée et source d'erreurs lors d'ajustements de paramètres pour l'optimisation de l'architecture et l'application dans l'objectif de respecter les contraintes temps réel de l'application.

Nous avons mis en place un formalisme de fichiers permettant de fixer les contraintes de dimensionnement de l'architecture mais également les contraintes de configuration logicielle. Ce formalisme permet au concepteur de modifier l'architecture et les contraintes de placement sans avoir à recompiler l'ensemble des sources comme c'était le cas auparavant. Ceci permet un gain de temps lors de la phase d'ajustement de l'AAA et de s'affranchir d'erreurs de compilation.

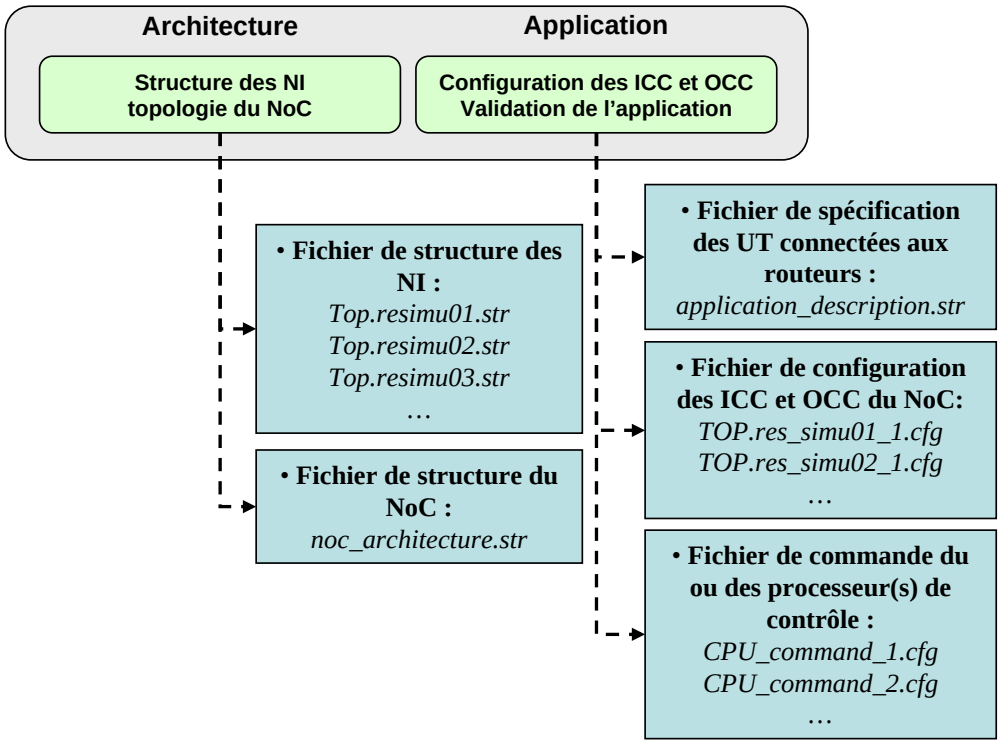

FIG. 4.6 – Générations des fichiers de contraintes du modèle SystemC du NoC

Ce formalisme mis en place est le même pour les deux modes de fonctionnement de notre flot offrant une portabilité suivant le cas dans lequel le concepteur se place : adéquation manuelle ou adéquation automatique. Par conséquent, les deux modes doivent s'adapter aux contraintes imposées par le modèle SystemC en fournissant les mêmes fichiers de contrainte. Notre flot AAA réalise donc une exploitation des résultats fournit par l'heuristique de placement routage afin de les adapter au formalisme de fichier de contraintes mis en œuvre.

Les différents formats de fichiers permettant de décrire les parties applicatives et architecturales qui caractérisent le NoC sont mentionnés sur la figure 4.6.

Par ce formalisme, l'utilisateur peut modifier à tout moment les paramètres caractérisant l'architecture ou ceux de l'application ayant un impact sur les performances globales de l'application. De ce fait, la topologie du réseau et les contraintes de placement des blocs fonctionnels ou encore les caractéristiques matérielles de chaque NI peuvent être modifiées séparément. Par la même occasion, les chemins de routage, la taille des paquets, les configurations d'ICC et d'OCC ainsi que les paramètres de modélisation des blocs fonctionnels peuvent être également modifiés séparément pour la partie applicative. Nous allons décrire brièvement ces différents fichiers de contraintes et les contributions que nous avons apportées lors de ces travaux de thèse afin d'optimiser cette génération du modèle NoC.

4.4.1 Formalisme des fichiers d'entrée

4.4.1.1 Paramètres structurels des NI

Nous l'avons vu dans le chapitre précédent, les éléments mémoires (FIFO) des NI sont tous paramétrables indépendamment les uns des autres. Il est possible de dimensionner les FIFO entrantes et sortantes dans chaque NI (*Top.resimu01.str*, *Top.resimu02.str*, ...). Nous le verrons dans le chapitre 5 lors de la présentation des résultats d'expérimentation obtenus durant ces travaux de thèse que ces paramètres ont un impact direct sur les performances de l'application. Ils offrent notamment la possibilité de réduire la latence des données lors des communications qui est un des défauts dans les NoC à QoS de type BE.

4.4.1.2 Paramètres de description de la topologie réseau

Les paramètres de description de la topologie réseau permettent au concepteur de décrire de manière simple la topologie du réseau voulu. Dans le flot de conception d'origine, cette description de l'architecture était réalisée directement dans le code source et ce de manière non paramétrable.

Le NoC FAUST utilisé ici possède une topologie de type 2D torus qui est une dérivation de la structure 2D mais avec un repliement des bords extérieurs entre eux. Ainsi, la topologie est décrite sous une forme matricielle. Le routage employé dans ce NoC est un routage par la source. Ce type de routage permet d'avoir des topologies irrégulières (suppression des routeurs n'ayant pas de blocs fonctionnels connectés) contrairement au mode XY qui lui impose une structure régulière au risque d'avoir des erreurs de routage. Durant ces travaux de thèse, nous avons souhaité rester dans des cas de structures régulières car le routage de ceux-ci est plus simple de réalisation et de mise en œuvre.

Nous avons mis en place un format de fichier de description de l'application permettant à l'algorithme mis en œuvre de générer automatiquement la topologie souhaitée. Le concepteur décrit la topologie du réseau souhaitée de manière simple et ces informations sont exploitées par l'algorithme. La particularité de cette méthode réside dans la possibilité de créer un NoC ayant une structure mono-composant ou multi-composants. En effet, dans le chapitre 5, nous présentons des résultats d'expérimentation que nous avons obtenus dans le cadre de notre contribution au projet 4MORE nécessitant la simulation et le test d'un contexte NoC multi-composants hétérogène. Un exemple de fichier de description d'une topologie NoC est donné sur le figure 4.7.

La description de la topologie s'effectue de manière hiérarchique. Dans un premier temps, le concepteur doit spécifier une matrice globale renseignant le nombre de routeurs

à créer avec la dimension de la matrice. Puis dans un deuxième temps, le concepteur liste les différents composants qui vont venir se placer sur la matrice globale de routeurs.

```
64 8 8
4
0 FAUST_1 3
1 FAUST_2 3
2 FPGA_1 6
3 FPGA_2 6
```

FIG. 4.7 – Exemple d'un fichier de description de topologie

Pour chacun de ces composants, un fichier est associé permettant de mentionner leur position relative dans la matrice globale ainsi que les ports d'entrée/sortie qui vont subir une exception dans le routage des routeurs entre eux. Ces paramètres vont permettre à l'algorithme d'interconnecter les entrées/sorties des différents composants entre eux afin de réaliser un NoC multi-composants. La figure 4.8 présente le concept de réalisation d'une topologie mono et multi-composants.

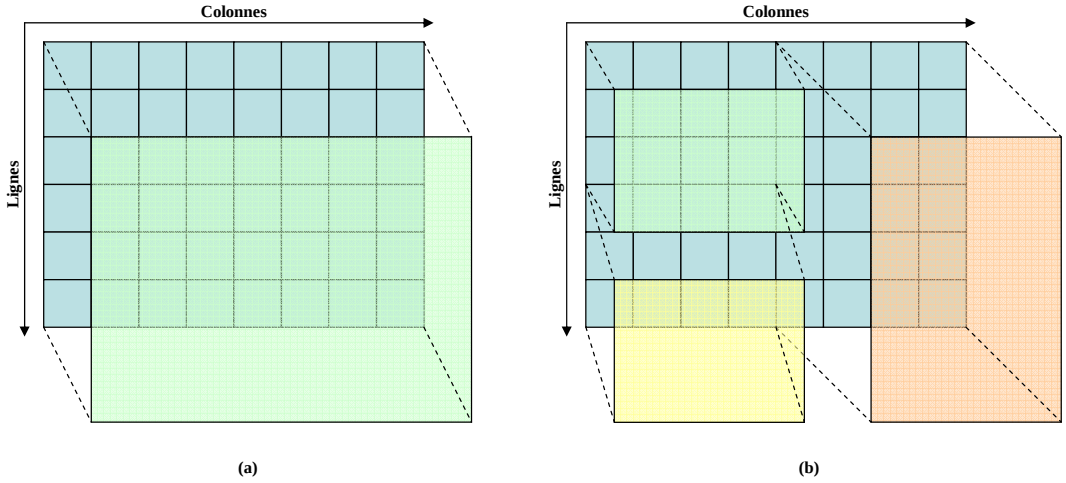

FIG. 4.8 – Structure de topologie en contexte mono-composant (a) et multi-composants (b)

Avec ces paramètres, l'algorithme réalise la création de la topologie en trois phases : le routage interne des composants, ensuite le routage des bords extérieurs des composants et enfin le routage des I/O entre les composants. Ces trois étapes prises en charge dans l'algorithme que nous avons mis en œuvre sont illustrées sur la figure 4.9.

4.4.1.3 Paramètres de description des unités de traitement

Auparavant, les paramètres permettant de décrire les caractéristiques de chaque unité de traitement étaient décrits directement dans le code SystemC. Nous avons donc mis en place un fichier de description de l'application pour lequel nous renseignons les différents

paramètres du schéma de modélisation simplifié des blocs fonctionnels décrit dans la section 2.2.2.

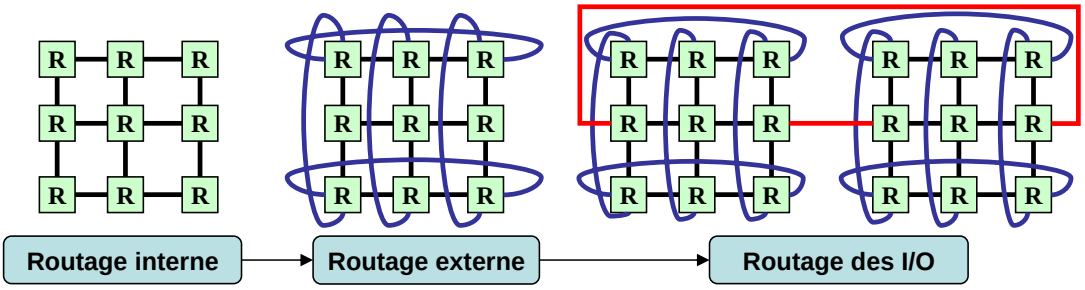

FIG. 4.9 – Étapes de routage de la topologie du réseau

De plus, nous avons rajouté la notion de contrainte de placement des UT sur les routeurs du composant auquel ils appartiennent. De cette manière, chaque UT qui constitue l'application possède une description de son schéma de modélisation simplifié ainsi qu'une contrainte de placement sur la matrice du NoC.

```
// **** IP description ****
// 0 = Test bench
// 1 = RAM
// 2 = CPU
 64
 2  res_simu00  0   1   10  CPU  0
 2  res_simu43  43   2   42  CPU  1
 1  res_simu08  8   1   9  ASIC_RAM_1_TX_RESOURCE_08  1   1   2   16   48   192
 1  res_simu10  10   1   11  ASIC_RAM_2_RESOURCE_10  1   1   2   2560   15360   38400
 1  res_simu28  28   3   15  RAM_RF_IF_TX_ANTENNA_1_RESOURCE_28  1   1   2   2560   15360   38400
 1  res_simu42  42   2   41  TX_DMA_ENGINE_RESOURCE_42  1   1   2   16   48   192
 1  res_simu44  44   2   43  ASIC_RAM_2_RESOURCE_44  1   1   2   2560   15360   38400
 1  res_simu48  48   4   47  RAM_RF_IF_TX_ANTENNA_2_RESOURCE_48  1   1   2   2560   15360   38400
 0  res_simu01  1   1   0  OFDM_MODULATION_1_RESOURCE_01  1   24   1   1280   0   2620
 0  res_simu02  2   1   1  MIMO_ENCODER_RESOURCE_02  1   48   1   96   0   50
 ...
```

FIG. 4.10 – Exemple de fichier de description de l'architecture

4.4.1.4 Paramètres de configuration logicielle des NI

Ces paramètres étaient déjà mis en place dans le flot de conception initial et n'ont donc pas reçu de modifications. Il faut noter qu'un fichier de configuration permet de renseigner une configuration d'ICC et une configuration d'OCC en même temps. Or, certains blocs fonctionnels nécessitant par exemple deux flots de données différents en entrée impliquent le chargement de deux configurations dans le registre de l'ICC. Ainsi, dans ces cas particuliers,

deux fichiers de configuration d'ICC seront à décrire. Ces fichiers, rappelons-le, seront lus et interprétés par le processeur de contrôle supervisant le NoC.

4.4.1.5 Paramètres de configuration des commandes du processeur de contrôle

Le processeur de contrôle, comme nous l'avons vu dans le chapitre 3, sert à initialiser, configurer et valider les différentes entités connectées aux routeurs du réseau. Ces différentes commandes exécutées par ce CPU étaient auparavant lancées directement dans le code SystemC. Dans l'objectif d'améliorer le flot de conception en lui apportant une plus grande souplesse de mise en œuvre, nous avons créé un fichier de commande par CPU (*CPU_command.cfg*) renseignant les trois étapes de la configuration du réseau. Les trois phases prises en charge par le CPU sont les suivantes :

1. chargement des chemins d'accès aux différentes entités du réseau
2. chargement des configurations des registres des ICC et OCC des différentes UT utilisées
3. validation des configurations d'ICC et d'OCC

La phase 1 utilise des fichiers de chargement, par exemple "load_simu01.cfg", renseignant le chemin à emprunter dans le réseau pour accéder à une ressource, la ressource 1 par exemple. Cette spécification de chemin est ensuite utilisée dans les phases 2 et 3 pour envoyer respectivement les configurations d'ICC et d'OCC (*TOP.res_simu01_1.cfg*) puis les commandes de validation des configurations d'ICC et d'OCC.

Ainsi, tous les critères de génération du modèle SystemC du NoC présentés ci-dessus vont être intégrés dans les deux modes de fonctionnement du flot de conception.

4.4.2 Flot en mode automatique

Dans ce flot de conception, lorsque l'heuristique de l'AAA est terminée, les contraintes de placement et de routage sur l'architecture (ARG) sont interprétées afin de générer l'ensemble des fichiers nécessaires à la génération du NoC et de son application.

Certains de ces paramètres ne peuvent être déterminés directement de la phase AAA, comme par exemple le dimensionnement des FIFO des NI. Ce dimensionnement des FIFO entrantes et sortantes de chaque NI est ajusté lors de l'exploitation des données de simulation montrant par exemple une latence trop importante dans certaines communications. Par conséquent, certaines FIFO nécessiteront une taille plus importante afin de diminuer des latences de communication qui entraînent un non respect des contraintes temporelles de l'application.

Dans ce mode automatique, nous avons fait le choix de fixer l'ensemble des tailles des FIFO entrantes et sortantes à une valeur de 16 FLIT sachant que la taille des paquets de données et de crédits est aussi fixé à une valeur par défaut de 8 FLIT. Un paquet de données ou de crédits sera émis sur le réseau dès lors que le seuil de remplissage de la FIFO dépassera ou descendra sous la barre des 8 FLIT.

Pour finir, les contraintes déterminées par l'heuristique de placement routage ne sont pas nécessairement optimales dans le sens où elles ne peuvent tenir compte des latences sur les communications induites par la congestion du réseau en régime établi (QoS de type BE). Cet outil permet d'aider le concepteur dans la phase d'adéquation de l'application

sur l'architecture avec les contraintes imposées par celle-ci. Mais les résultats de simulation du modèle SystemC généré vont permettre d'observer les performances obtenues sur l'architecture, et dans le cas du non respect des contraintes temporelles d'affiner certains paramètres indépendamment les uns des autres, afin de tendre vers le respect des contraintes temporelles de l'application.

4.4.3 Flot en mode semi-automatique

Dans le cas d'une utilisation du flot de conception en mode semi-automatique, la description des contraintes de placement de l'application sur l'architecture est réalisée manuellement. Par conséquent, le concepteur doit décrire manuellement l'ensemble des contraintes de placement de l'application sur l'architecture avec ses fichiers associés permettant de générer automatiquement le modèle SystemC du NoC.

Cette étape peut s'avérer longue, fastidieuse et source d'erreurs particulièrement lors de la génération d'un NoC multi-composants. C'est pour cette raison que nous avons fait le choix d'utiliser le tableur Excel afin de spécifier les différents paramètres de l'application ainsi que ceux de l'architecture. Une macro (programme codé en Visual Basic permettant

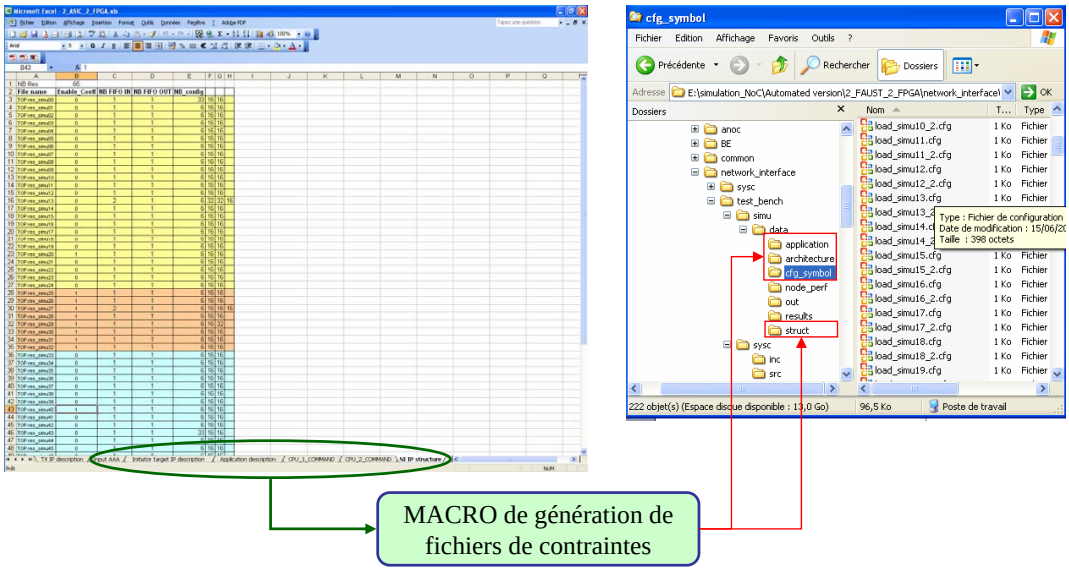

FIG. 4.11 – Exemple de génération des fichiers de création du modèle SystemC du NoC à partir d'une description sous Excel

d'opérer sur les différentes cellules de données des différentes feuilles que contient le fichier Excel) parcourant les différentes feuilles du fichier Excel vont générer automatiquement les différents fichiers représentant les 5 critères pris en compte dans la génération du modèle SystemC. Le concepteur va donc renseigner l'ensemble des critères nécessaires pour décrire l'application, l'architecture et les contraintes des deux entre elles de manière graphique, la génération des différents fichiers nécessaires à la création du modèle du NoC en SystemC étant effectué automatiquement par la macro que nous avons mis en œuvre. Cette génération est rapide et inférieure à une seconde comparée à la méthode entièrement

manuelle où la génération de ces fichiers pouvait prendre plusieurs jours. Il faut également noté que dans le cas d'un contexte multi-composants, l'utilisateur devra veiller à instancier un nombre de processeurs de contrôle suffisant pour accéder à toutes les UT de la matrice globale. Ceci s'explique par un nombre de bits réservé au codage du chemin de routage dans l'en-tête des paquets limitant le nombre maximal de chemins à 8. La figure 4.11 montre le principe de la génération de ces fichiers au moyen d'une macro exploitant les données fournit dans un fichier Excel.

Cette description offre donc une plus grande souplesse dans la description manuelle du contexte de simulation mais également lors des phases d'ajustement de paramètres après exploitation des données issues des résultats de simulation. On obtient un gain de temps substantiel dans la phase d'ajustement des différents paramètres pouvant influer directement ou indirectement sur les performances globales de l'application. A titre d'exemple, le contexte multi-composants dont nous présentons les résultats obtenus dans le chapitre 5 nécessite la génération de 282 fichiers pour construire le NoC avec son application.

A terme, nous souhaitons également générer un fichier Excel lors d'une utilisation du flot de conception en mode automatique. Ainsi, le concepteur pourra bénéficier des contraintes de placement et de routage fournit par l'heuristique de la phase AAA tout en ayant la possibilité d'affiner ces paramètres par une description graphique sous Excel.

Nous allons voir maintenant l'environnement de simulation dans lequel sont effectuées les simulations et notamment quelles données sont fournies à l'utilisateur afin d'observer les performances du réseau et de son application.

4.5 Environnement de simulation

Dans cette section, nous allons voir quelles données sont fournies au concepteur à la fin des simulations pour évaluer les performances du réseau. Les simulations sont effectuées sous l'OS Linux red Hat 7.2. Ce système est également portable sur plate-forme UNIX. Nous allons voir dans cette section les deux possibilités offertes au concepteur pour valider leur application sur une architecture à base de NoC : simulation et/ou émulation.

4.5.1 Validation par simulation

Nous l'avons vu dans le flot de conception présenté précédemment le modèle du NoC en SystemC généré fournit des résultats de simulation au niveau des ressources de traitement mais également au niveau des routeurs. Nous allons voir chacun de ces éléments d'analyse de performances.

4.5.1.1 Performances des routeurs

Compte tenu de la QoS de type BE offerte par le NoC FAUST que nous utilisons, nous avons souhaité mettre en place un outil d'analyse de performances réseau. Cet outil a pour objectif de donner au concepteur des informations de bande passante sur l'ensemble des liens de chaque routeur du réseau.

Nous avons donc mis en œuvre un système d'analyse de performances du réseau afin de fournir au concepteur une image des bandes passantes occupées pour chaque lien de chaque routeur à intervalle de temps réguliers. A la fin de chaque simulation de l'application sur le réseau, un fichier de performances est associé pour chaque routeur (*TOP.node16.txt* pour

le routeur numéro 16 du NoC) renseignant l'occupation de la bande passante sur chaque lien de celui-ci. La figure 4.12 donne un exemple de statistique de bande passante sur les liens du routeur 16 d'une application.

Par ces analyses de performances au niveau des routeurs et plus particulièrement leurs liens, le concepteur peut vérifier pendant la totalité de la durée de la simulation si certains liens subissent une congestion. Il est ainsi possible de modifier des chemins de routage ou modifier les contraintes de placement des blocs fonctionnels sur la structure du NoC afin de soulager les liens en défaut.

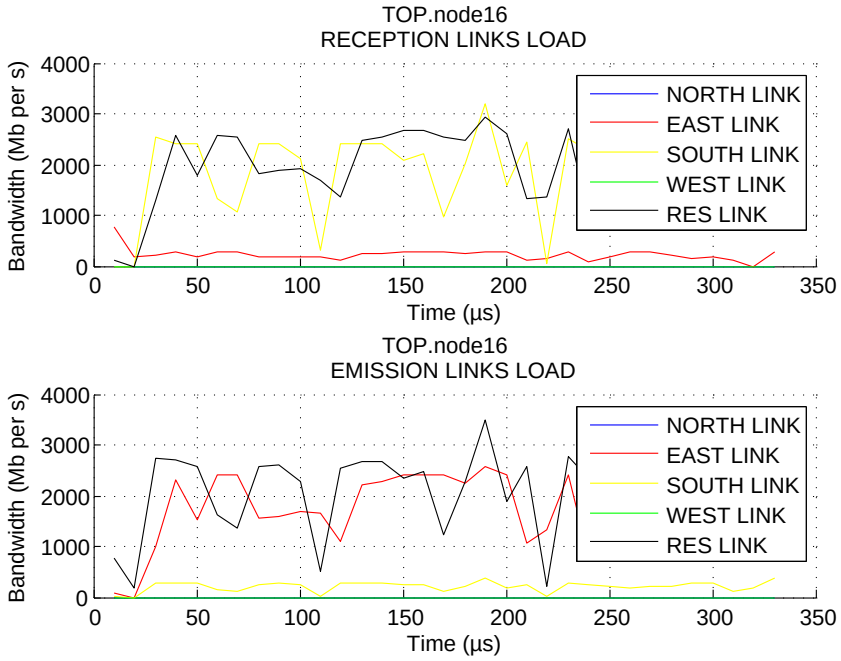

FIG. 4.12 – Exemple de statistiques de bande passante sur les liens du routeur 16

4.5.1.2 Performances des ressources de traitement

L'objectif des simulations du modèle NoC généré par notre flot de conception est de valider les contraintes architecturales et applicatives afin de vérifier le respect des contraintes temps réel de l'application. Dans cet objectif, nous avons intégré à l'outil de conception un élément d'analyse de contraintes de temps sur chaque entité connectée aux routeurs du réseau.

De part la structure même du réseau et de son mécanisme d'acquittement et de formatage des données des communications, nous avons découpé l'analyse temporelle des flots de données dans chaque UT en trois parties distinctes. En effet, le schéma de modélisation simplifié des blocs fonctionnels que nous avons mis en place dans le modèle SystemC est composé de trois parties : taille des données en entrée, temps de traitement et taille des données en sortie.

Ainsi, ces trois étapes correspondent à une latence dans le flot de données. La somme de ces trois latences constituent la latence globale T de l'UT définit par :

$$T = T_i + T_t + T_o \tag{4.6}$$

Où T_i représente le temps d'attente en entrée du bloc pour obtenir le montant de données nécessaire avant de commencer le traitement, et T_t représente le temps de traitement des données, celui-ci restant constant et équivalent à celui renseigner dans le modèle. Enfin T_o représente le temps d'attente en sortie du bloc pour envoyer les données produites par le bloc de traitement.

Ces trois contraintes temporelles sont renseignées dans un fichier (*resource29.res* pour la ressource 29) pour toutes les ressources de traitement du NoC (élément mémoire ou UT). La figure 4.13 donne un exemple des différentes latences présentées dans le paragraphe précédent pour chaque cycle de traitement effectué au sein d'une ressource.

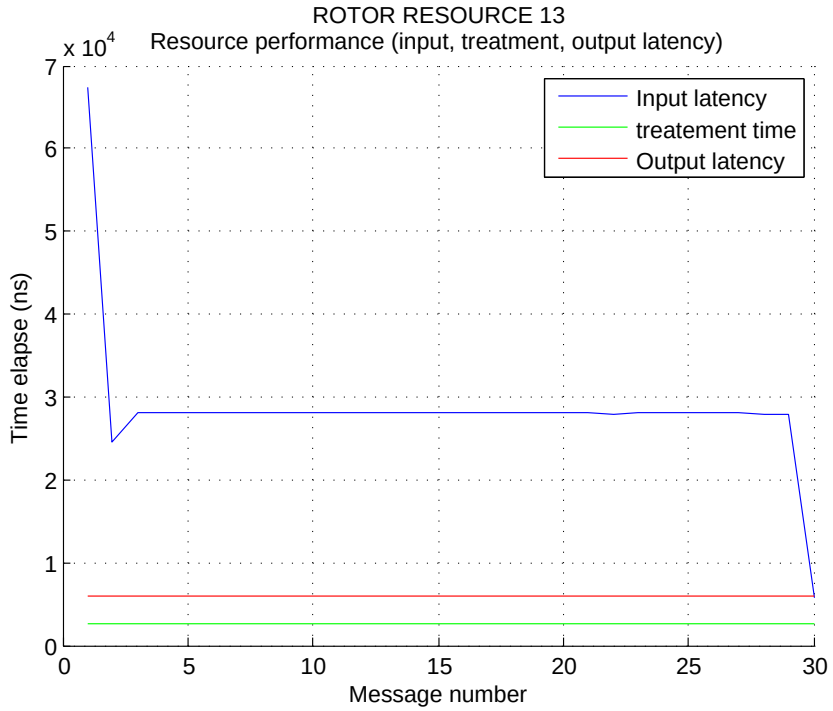

FIG. 4.13 – Tracé des composantes de la latence globale d'une ressource de traitement du type ROTOR

Les informations de latence en entrée ou en sortie permettent de montrer qu'il y a une congestion en entrée ou en sortie de ce bloc (trafic important sur ce routeur) qui implique une modification éventuelle des chemins de routage ayant en commun ce routeur. Mais cette latence peut également être induite par les UT nécessitant de grandes quantités de données en entrée pour effectuer des traitements (une FFT sur 1024 point par exemple). Dans ce cas, une augmentation de la taille des FIFO entrante et/ou sortante dans cette NI permet de mémoriser les données en entrée ou en sortie pendant qu'un traitement est

en cours. On donne la possibilité de réaliser un "pipeline" des communications réduisant les latences en entrée et en sortie des UT.

Par contre, le coût en surface mémoire engendré par les profondeurs des FIFO dans les NI en fonction du nombre de routeurs instanciés dans le réseau peuvent grandir linéairement. La figure 4.14 montre le coût de ces tailles de FIFO en fonction du nombre de routeurs composant la structure du NoC.

L'ajustement des tailles de FIFO permet d'accroître les gains en performance de l'application sur le réseau mais le concepteur doit également prendre en considération le coût de celles-ci pour l'implantation matérielle.

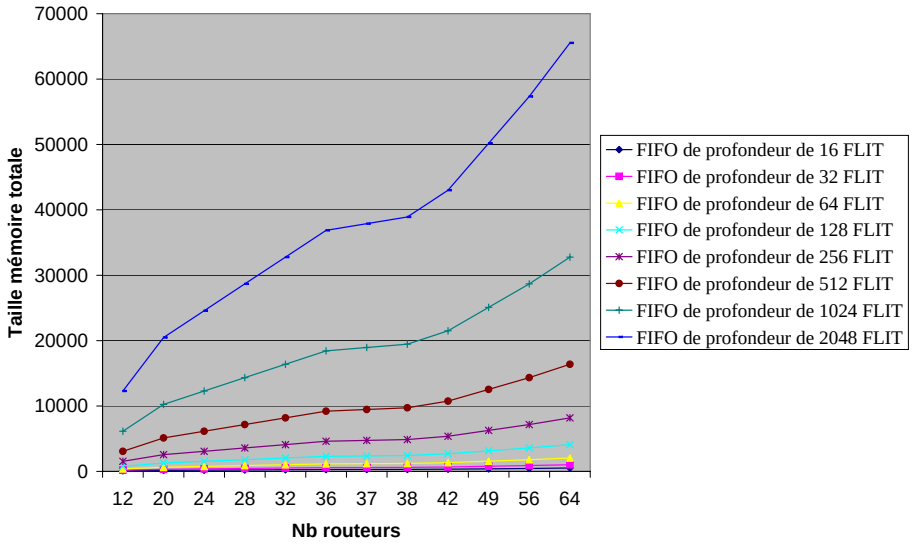

FIG. 4.14 – Coût des profondeurs de FIFO des NI des unités de traitement connectées aux routeurs en fonction de la taille du réseau

Pour finir, nous allons voir dans la dernière section, la dernière étape de notre flot de conception permettant de valider l'AAA par émulation sur plate-forme basée sur une architecture de type FPGA.

4.5.2 Validation par Emulation

Dans le flot de conception, nous avons souhaité avoir la possibilité de générer le code VHDL équivalent au modèle SystemC simulé par une description XML. Celle-ci n'est pas encore mise en œuvre de manière complètement automatique dans ce flot de conception mais elle fait partie des perspectives à court terme de ces travaux de thèse. L'idée de ce formalisme repose sur la possibilité de générer automatiquement et rapidement le modèle SystemC simulé et validé par le concepteur afin de pouvoir valider l'AAA par une émulation sur plate-forme reconfigurable. Ceci dans le but de proposer au concepteur une autre possibilité d'exploration architecturale. Ainsi, on permet d'accélérer le temps de simulation par l'émulation mais on augmente également le nombre de simulations possibles pour valider l'application sur une architecture NoC (simulations SystemC et par émulation).

Or nous l'avons vu en début de ce chapitre (section 4.1) l'intégration de blocs VHDL au sein d'une structure NoC s'avère complexe, particulièrement dans les cas de blocs de traitement à taille hétérogène. Une autre notion importante à prendre en compte dans cette implantation est la validation algorithmique de ces blocs lors d'une encapsulation NoC. En effet, certains blocs de traitement d'une chaîne algorithmique nécessitent des paramètres de configuration afin de pouvoir traiter correctement les informations de données. Ainsi, une implantation réelle peut s'avérer longue et coûteuse en temps du fait de la complexité de la phase de conception. D'autre part, les paramètres obtenus en simulation peuvent conduire le concepteur à des choix de contraintes de placement ou de fréquence de fonctionnement du NoC qui dans certains cas ne seront pas réalisable sur plate-forme reconfigurable par l'intermédiaire des outils de synthèse associés. Ces limitations peuvent être induites par la contrainte de surface de l'architecture visée mais aussi par l'impossibilité d'obtenir un design remplissant les contraintes de temps exigées par le concepteur.

Ainsi, la phase d'implantation en conditions réelles avec les véritables blocs VHDL peut s'avérer impossible car les contraintes de l'architecture sont trop importantes pour que l'outil de synthèse puisse respecter les contraintes temporelles de l'application.

C'est pour ces raisons de complexité d'implantation que nous avons choisi de mettre en place un formalisme de modélisation générique des blocs fonctionnels connectés aux routeurs de manière à offrir une plus grande souplesse lors de l'émulation. Ce formalisme a pour objectif de permettre au concepteur de réaliser une implantation matérielle rapide et automatique en s'affranchissant de tous les problèmes induits par une implantation réelle.

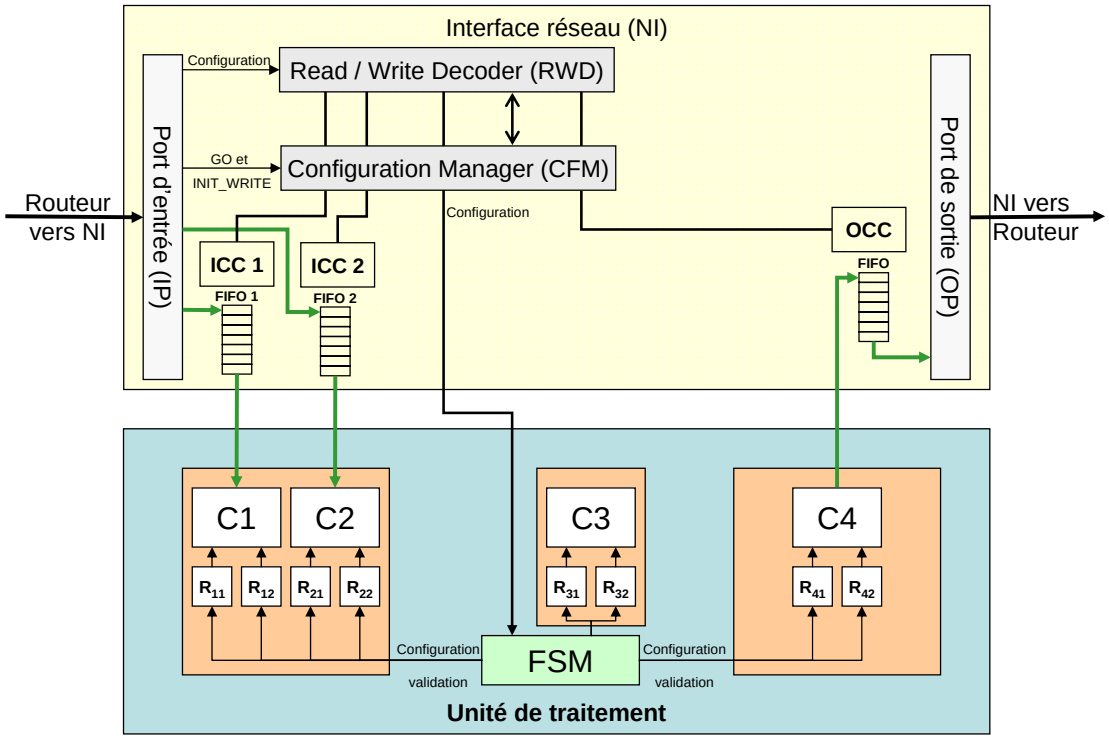

FIG. 4.15 – Modélisation de l'unité de traitement pour une généricité lors de l'émulation

Ces blocs vont avoir une structure matérielle identique, de faible taille et entièrement paramétrable logiciellement de façon à leur rendre individuellement un comportement différent et propre aux contraintes de l'application. Cette modélisation des blocs fonctionnels repose sur les mêmes bases que le schéma de modélisation simplifié du modèle SystemC. Nous modélisons par des compteurs les flux de lecture et écriture sur les FIFO entrantes et sortantes de la NI des blocs fonctionnels. Ces compteurs vont donc générer le trafic de données sur le réseau. Ceux-ci vont lire ou écrire des FLIT à chaque cycle d'horloge dans la FIFO qu'ils ont en charge suivant un montant maximal renseigné dans des registres associés. La latence exprimée en nombre de cycles représentant le temps de traitement des données, est également modélisée par un compteur synchrone ayant une limite de comptage maximale renseignée dans un registre.

L'ensemble des trois étapes qui caractérise le comportement d'un bloc fonctionnel au sein du NoC (lecture, traitement, écriture) est régie par une machine d'état (FSM : Finite State Machine). L'architecture générique décrite et mise en œuvre est présentée sur la figure 4.15.

La FSM va réaliser la synchronisation des traitements de l'unité en ayant un lien avec le CFM de la NI qui a la charge d'enchaîner les configurations des registres des ICC et OCC. En effet, suivant les configurations qui ont été validées au sein du CFM, l'unité de traitement peut avoir un traitement des données différent (commande INIT_WRITE par exemple). Par conséquent, le bloc de traitement ne va pas nécessairement opérer sur les mêmes blocs de données d'une configuration à l'autre. Nous avons donc mis en place deux registres par compteur afin de renseigner le nombre de FLIT maximal à lire ou à écrire dans la FIFO pour laquelle il est affecté en fonction de la configuration jouée par le CFM. Cette modélisation a été restreinte à deux registres car seulement deux configurations par FIFO peuvent être validées actuellement. D'autre part, lorsqu'un nombre de boucles a été mentionné pour l'enchaînement des configurations validées au niveau de l'ICC, la FSM opère le basculement entre les deux registres.

Ce modèle générique de l'UT permet de donner un comportement différent à chaque bloc fonctionnel tout en ayant une architecture identique à faible coût en surface. Ces registres sont donc paramétrés logiciellement par le processeur de contrôle. Ces paramètres sont envoyés par le processeur de contrôle en même temps que le chargement des registres de la NI. Les registres des compteurs C1, C2, C3, C4 de notre modélisation de l'UT sont accessibles par des adresses hautes supérieures à 128 (cf. figure 3.17).

Notre objectif à court terme est de pouvoir générer automatiquement le code VHDL associé au modèle SystemC simulé par le biais d'une description en langage XML. Celle-ci n'est pas encore mise en place mais nous présentons dans le chapitre 5 (section 5.3) des résultats d'implantation sur plate-forme reconfigurable à base de FPGA Xilinx de la famille de Virtex4 en ayant un code VHDL produit manuellement.

4.6 Conclusion

Nous avons présenté dans ce chapitre les différentes contributions apportées dans ces travaux de thèse au flot de conception. Ces différentes contributions permettent au concepteur de réaliser un Co-Design en mode complètement automatique (phase AAA) ou en mode semi-automatique (description des contraintes sous le logiciel Excel).

Ce flot de conception modifié permet d'offrir une plus grande souplesse de mise en œuvre notamment dans les phases d'optimisation manuelle des différents paramètres du réseau pouvant influer sur les performances globales de l'application (chemin de routage, contraintes de placement, taille des FIFO, taille des paquets de crédits et de données, fréquence de fonctionnement du réseau, ...). Le concepteur peut ainsi optimiser son adéquation plus rapidement en utilisant des fichiers de scripts permettant de lancer plusieurs simulations en modifiant les orientations de l'AAA de façon à privilégier les performances ou bien restreindre les coûts des ressources matérielles.

Pour finir, nous avons mis en place un modèle générique d'UT permettant de s'affranchir des problèmes de conception qui sont coûteux en temps lors d'une implantation réelle qui peut s'avérer impossible dans certains cas (fréquence de fonctionnement non obtenue après synthèse et placement routage par exemple). Ce modèle permet donc de réaliser rapidement et simplement une implantation matérielle sur plate-forme reprogrammable à base de FPGA en s'affranchissant des problèmes induits par une implantation avec des blocs fonctionnels réels. Le but de cette modélisation est de valider matériellement les performances obtenues lors des simulations du modèle SystemC. Ce flot permettra à terme de générer automatiquement les sources VHDL à chaque nouvelle génération du modèle SystemC par le biais d'une description en langage XML. Nous allons présenter dans le chapitre suivant les résultats obtenus au moyen de ce flot de conception pour nos contributions dans le cadre du projet 4MORE mais également lors de nos implantations matérielles.

Chapitre 5

Résultats d'expérimentation

Sommaire

Ce chapitre présente les différents résultats obtenus lors de ces différents travaux de thèse notamment dans le cadre du projet 4MORE dans lequel nous étions impliqués. Cette contribution s'inscrivait dans le cadre du WP4 ayant en charge de proposer des solutions de co-design pour la conception d'un SoC permettant de réaliser un démonstrateur final d'un terminal mobile de 4^{e} génération. Dans cet objectif, les différents partenaires du projet ont validé l'utilisation d'un média de communication basé sur une structure NoC pour la réalisation du SoC final afin d'intégrer l'ensemble des blocs fonctionnels constituant les différents éléments de la chaîne algorithmique développés et validés dans le cadre du WP1.

Compte tenu des avantages liés aux reseaux sur puce mentionnés dans le chapitre 1 mais également des inconvénients liés à la complexité de mise en œuvre de ce nouveau média de communication émergeant pour les SoC, il a été planifié des phases d'exploration afin de trouver les meilleurs compromis possibles pour l'AAA. C'est dans ce cadre précis que le WP4 a eu la charge d'explorer l'espace des solutions architecturales afin de proposer des solutions d'intégration au WP6 chargé de l'intégration des blocs VHDL dans le SoC final servant de démonstrateur matériel pour le projet.

Ainsi, notre contribution au sein du projet Européen 4MORE s'est déroulée en trois étapes ayant donné lieu à un rapport interne [88, 89, 90] pour chacune d'elle.

La première étape consistait à spécifier les blocs fonctionnels décrits par le WP1, afin de pouvoir intégrer ces paramètres dans le modèle SystemC du NoC FAUST. Cette étape a été décrite dans le cadre du chapitre 2 situant le contexte du projet et détaillant chacun des blocs fonctionnels constituant la chaîne algorithmique suivant le modèle simplifié des blocs fonctionnels présentés dans ce même chapitre.

Les deuxième et troisième étapes ont consisté à réaliser des explorations architecturales en utilisant le modèle SystemC du NoC FAUST dans un contexte mono-composant d'une part (SoC final pour une production réelle du terminal mobile 4G), et dans un contexte multi-composants d'autre part (SoC du démonstrateur réalisé dans le projet).

Par ailleurs, durant ces travaux de thèse, nous avons souhaité proposer une approche complémentaire des NoC en utilisant le bus PCI (Peripheral Component Interconnect) d'un ordinateur de type PC afin d'y connecter une carte de prototypage à base de composants reprogrammables de type FPGA. Le concept de ces travaux étaient d'utiliser ce bus dans un contexte multi-ponts afin qu'il s'apparente à un réseau sur puce (notion de bus hiérarchique étendu).

Enfin, les derniers travaux menés durant cette thèse ont porté sur la transcription, du modèle SystemC simulé obtenu par le flot de conception, en VHDL de manière à pouvoir réaliser une émulation rapide sur composant de type FPGA. Comme présenté dans le chapitre 4, nous souhaitons à court terme pouvoir générer automatiquement le code VHDL du modèle SystemC simulé au sein de notre flot de conception par le biais d'une description XML. Nous présentons ici une implantation matériel du réseau par une description manuelle du code VHDL pour un NoC de dimension 4×4. Cette implantation du NoC nécessitant un processeur de contrôle afin de le configurer et le gérer, nous avons connecté le cœur de processeur MicroBlaze de Xilinx au NoC FAUST par l'intermédiaire de ses ports FSL (Fast Simplex Link).

Nous allons donc voir dans ce chapitre les différents résultats dans ces 4 contextes matériels présentés brièvement ci-desus.

5.1 Contribution dans le contexte du projet 4MORE

L'objectif du projet Européen 4MORE était d'investiguer parmi les techniques de communications actuelles les plus innovantes afin de trouver les plus pertinentes permettant de garantir les critères imposés par le cahier des charges. Ces investigations effectuées au sein du WP1 ont conduit au choix de la technique MC-CDMA pour la voie descendante et de la technique SS-MC-MA pour la voie montante. Concernant les simulations NoC en SystemC, nous avons fait le choix d'un modèle équivalent simplifié de chaque bloc fonctionnel caractérisant les éléments algorithmiques des voies montante et descendante. Le détail de chacun de ces blocs a fait l'objet d'études réalisées durant la première étape de notre contribution au projet [88]. L'ensemble des paramètres caractérisant les blocs des voies montante et descendante sont décrits et présentés dans le chapitre 2.

Les deuxième et troisième étapes de notre contribution au projet ont concerné des études sur NoC dans un contexte mono-composant d'une part, et dans un contexte multi-composants d'autre part. Dans les deux contextes l'application visée est un algorithme de traitement du signal d'un terminal mobile de 4^{e} génération. Nous allons donc voir dans les sections suivantes chacune des contributions apportées dans ces deux contextes.

5.1.1 Contexte mono-composant

Le contexte mono-composant avait pour objectif de valider la faisabilité d'une intégration d'une application 4G sur un média de communication de type NoC sachant que le NoC utilisé est en "Wormhole" "packet switching" avec une QoS de type BE. L'interrogation majeure à propos de ce média concernait la qualité de service proposée car nous l'avons vu dans les chapitres précédents, celle-ci ne donne aucune garantie concernant les performances de l'application (conditions temps réel).

Ainsi le contexte mono-composant consistait à vérifier si l'intégration d'une telle application était possible. Auquel cas, quelles conditions sont à remplir afin de garantir les contraintes temps réel de l'application. L'application présentée dans le chapitre 2 montre qu'une cadence symbole doit être respectée au sein de la trame exigeant une contrainte d'émission d'un symbole OFDM tous les 20.8μs. Par conséquent, lors de l'exploitation des résultats de simulation fournis par notre flot de conception nous porterons notre attention sur les blocs opérant sur des symboles OFDM entiers. Typiquement, dans notre application il s'agira d'observer les cadences symboles au niveau des modulations et démodulations OFDM. Nous allons donc voir dans les sous-sections suivantes, les caractéristiques de l'application, puis nous verrons les orientations données lors de nos explorations afin de garantir les contraintes temporelles de l'application dans les pire cas, et enfin, nous présenterons les résultats obtenus dans ce contexte mono-composant.

5.1.1.1 Contexte de l'application

Les caractéristiques des voies montante et descendante sont présentées respectivement sur la figure 5.1.

D'un point de vu fonctionnel, ces deux algorithmes sont indépendants l'un de l'autre. Par contre, d'un point de vue architectural, lors de nos simulations, ces deux voies ne peuvent fonctionner en même temps. En effet, les antennes étant les mêmes en émission et en réception et compte-tenu de la trame de transmission, le TX et le RX fonctionnent alternativement. D'autre part, d'un point de vue NoC, les communications du TX peuvent avoir des liens de communication en commun avec ceux du RX influant directement sur les performances de l'un par rapport à l'autre. Pour ces raisons, les résultats présentés pour le TX et le RX sont des simulations effectuées avec seulement les blocs fonctionnels de la voie simulée et validée. Les simulations n'étant effectuées que sur un seul slot de 32 symboles OFDM.

L'étude du contexte mono-composant ayant été réalisé avant la modification de notre flot de conception, les explorations architecturales ont été effectuées manuellement. Par conséquent, le placement des blocs fonctionnels sur la matrice du NoC a été effectué en minimisant les chemins des communications.

D'autre part, le dimensionnement de la matrice de routeurs constituant le NoC a été établi en dressant un bilan des contraintes de l'application en terme de besoins en blocs fonctionnels. Rappelons qu'un routeur de la matrice du NoC ne peut recevoir qu'un seul bloc fonctionnel. Le bilan est donc le suivant :

- 19 blocs fonctionnels pour les chaînes algorithmiques des voies montante et descendante
- 3 blocs mémoires dont 1 pour la voie montante et 2 pour la voie descendante
- 1 processeur de contrôle pour la gestion du réseau.

Ces besoins nous ont donc conduit à dimensionner la matrice de routeurs à 24 éléments avec une dimension de 4×6. Les choix de placement des différents blocs fonctionnels sur la matrice NoC de 24 éléments sont présentés sur la figure 5.2.

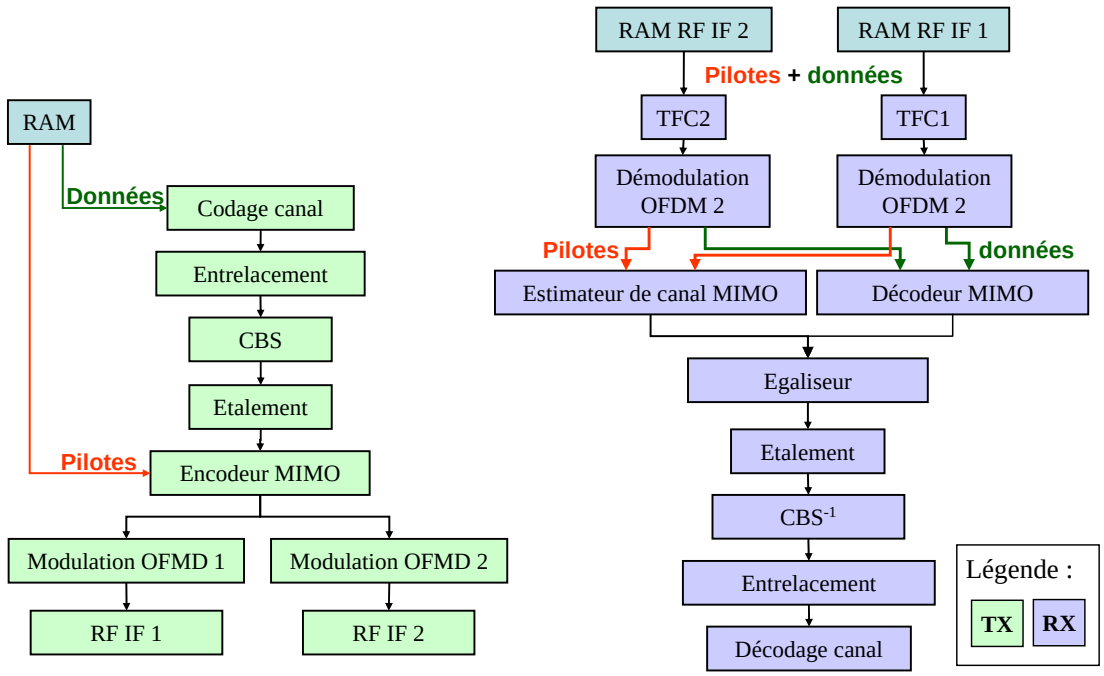

FIG. 5.1 – Diagramme fonctionnel des voie montante et descendante dans le contexte mono-composant

Nous allons donc voir dans la section suivante les différents paramètres que nous avons modifiés afin d'améliorer les performances globales de l'application toujours dans l'objectif de garantir les contraintes temps réel de l'application qui sont de 20.8μs par symbole OFDM.

5.1.1.2 Exploration de la voie montante

Nous présentons dans cette partie les résultats d'exploration obtenus lors de modifications de paramètres du réseau. Pour une topologie figée en ayant pris soin de minimiser les chemins de communication, les critères permettant d'augmenter les performances de l'application sont : l'augmentation des tailles des FIFO dans les NI, l'augmentation de la taille des paquets de données et de crédits et enfin, l'augmentation de la fréquence du réseau.

Nous avons souhaité observer dans un premier temps, les performances des blocs en charge de transmettre les symboles OFDM aux antennes (BB to RF 1 et 2) pour des tailles de FIFO fixées à 16 FLIT pour tous les NI et des tailles de paquets de crédits et de données fixées à 8 FLIT. Les performances obtenues pour une fréquence réseau variant de 125 à 333MHz sont présentées sur les figures 5.3.

On constate que les performances obtenues permettent de garantir les contraintes temps réel de l'application pour une fréquence de fonctionnement du réseau supérieure ou égale

à 166MHz. Afin de compléter cette étude, nous avons souhaité augmenter les tailles des FIFO des NI en tenant compte des tailles de messages que reçoivent chacun des blocs fonctionnels. Pour cela, nous avons fixé une fréquence de fonctionnement du réseau à 142.8MHz et nous avons étudié différents cas de taille de FIFO au niveau des NI. Ces cas d'étude sont présentés sur la figure 5.4. Le cas 1 sert de référence de performance pour lequel on ne respectait pas les contraintes temps réel de l'application. Nous présentons les performances obtenues pour ces différents cas d'étude pour une fréquence fixée à 142.8MHz sur la figure 5.5 sur laquelle nous avons ajouté les performances obtenues auparavant à 200MHz.

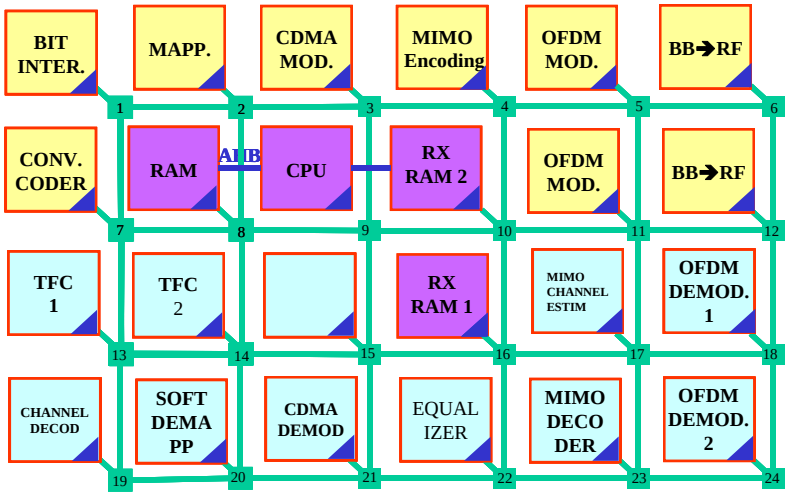

FIG. 5.2 – Topologie du réseau dans le contexte d'étude mono-composant

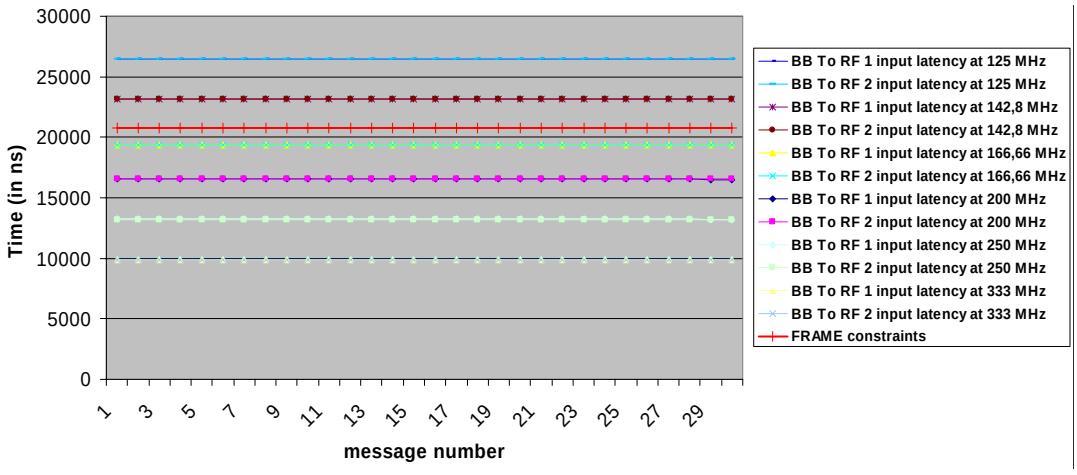

FIG. 5.3 – Latence globale des blocs fonctionnels BB vers RF en fonction de la fréquence réseau pour la voie montante dans le contexte d'étude mono-composant

On observe ainsi que dans le cas 2 où les FIFO ont été sur-dimensionnées, que les performances obtenues pour une fréquence de 142.8MHz sont similaires à celles obtenues dans à une fréquence de 200MHz pour des FIFO de taille faible (16 FLIT).

	MIMO		OFDM 1		OFDM 2		BB to RF 1		BB to RF 2	
CAS	Entrée	Sortie	Entrée	Sortie	Entrée	Sortie	Entrée	Sortie	Entrée	Sortie
Cas 1	16	16	16	16	16	16	16	16	16	16
Cas 2	16	16	16	**1280**	16	**1280**	**1280**	16	**1280**	16
Cas 3	16	16	16	**640**	16	**640**	**1280**	16	**1280**	16
Cas 4	16	16	16	**640**	16	**640**	**640**	16	**640**	16
Cas 5	16	16	16	**640**	16	**640**	16	16	16	16
Cas 6	16	16	16	**320**	16	**320**	**320**	16	**320**	16

FIG. 5.4 – Cas d'étude des tailles de FIFO de NI pour la voie montante dans le contexte d'étude mono-composant

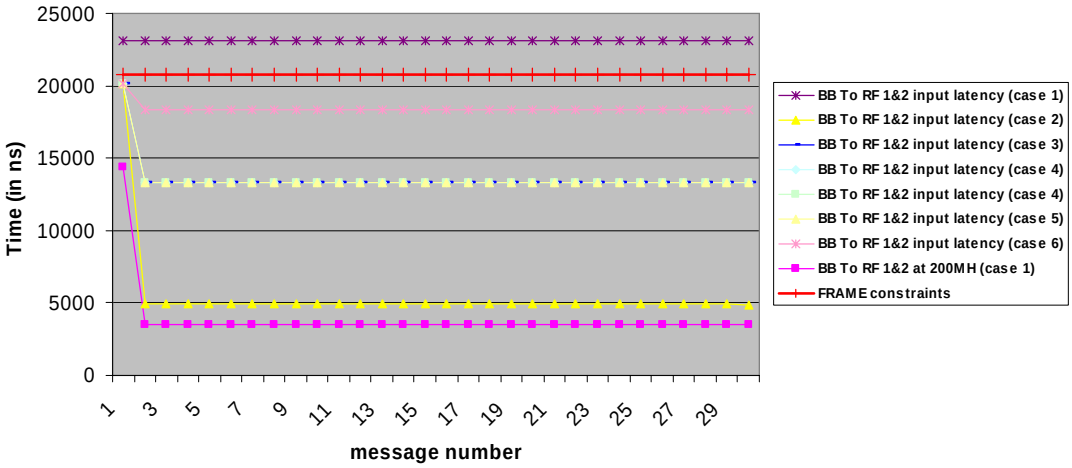

FIG. 5.5 – Performances observées sur les blocs BB vers RF 1 et 2 pour les différents cas d'étude de taille de FIFO des NI pour une fréquence réseau fixée à 142.8MHz pour la voie montante dans le contexte d'étude mono-composant

Ces études montrent qu'à ressources équivalentes on est obligé d'augmenter la fréquence de fonctionnement du réseau. Il s'agit là de réaliser un compromis entre ressource matérielle et fréquence d'utilisation (sous réserve que celle-ci puisse être atteinte lors de l'implantation). Or, nous l'avons dit, la synthèse et le placement routage d'un réseau lors d'une implantation matérielle restent des phases complexes qui ne permettent pas forcément d'atteindre les fréquences permettant de respecter les contraintes temps réel. Ainsi, l'augmentation des profondeurs de FIFO des NI permettent de proposer une alternative à ce problème de conception. On peut ainsi obtenir des performances à fréquence moindre avec des profondeurs de FIFO majorées par rapport à un réseau à fréquence plus élevée et des profondeurs de FIFO faibles.

5.1.1.3 Exploration de la voie descendante

Compte tenu des performances obtenues précédemment nous avons observé les performances au niveau des blocs fonctionnels TFC 1 et 2 (Time and Frequency Correction) pour une fréquence réseau fixée à 200MHz. Les résultats sont présentés sur la figure 5.6.

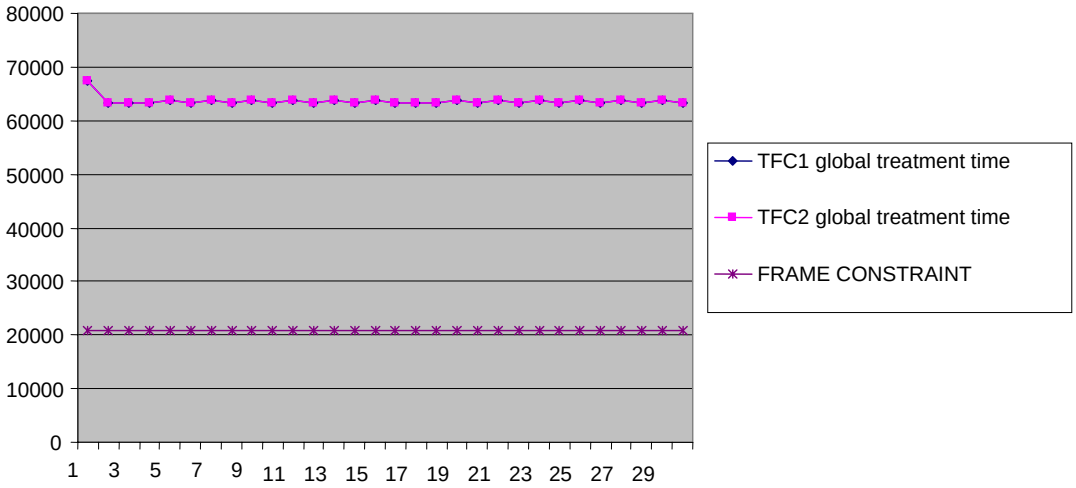

FIG. 5.6 – Latence globale des blocs fonctionnels TFC 1 et 2 pour une fréquence réseau de 200MHz pour la voie descendante dans le contexte d'étude mono-composant

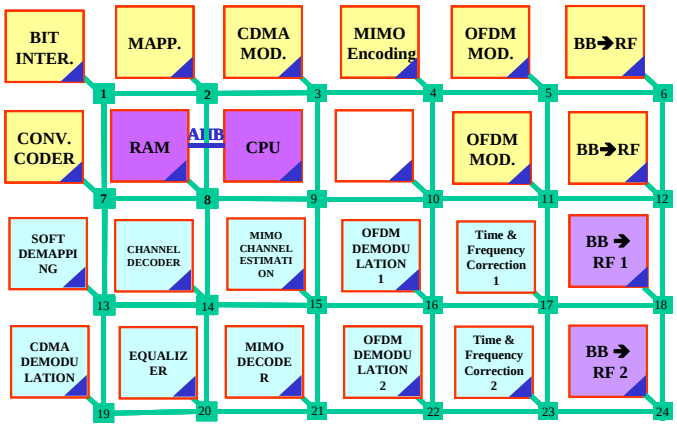

FIG. 5.7 – Topologie du réseau modifiée au niveau de la voie descendante dans le contexte d'étude mono-composant

On constate que les performances obtenues sont loin de pouvoir remplir les conditions imposées par le cahier des charges. Nous avons souhaité présenter cet exemple pour montrer l'impact du placement des blocs fonctionnels sur la structure du réseau. En effet, si l'on regarde la dépendance de données entre les blocs TFC1 et 2 et les blocs OFDM1 et 2, on se rend compte que les chemins de données parcourent toute la largeur de la matrice,

engendrant de grandes latences. Nous avons donc modifié la topologie du réseau et le placement des UT sur celui-ci afin d'obtenir les chemins de communication les plus courts et ainsi minimiser la latence des communications. La figure 5.7 montre la nouvelle structure du réseau.

En augmentant judicieusement les profondeurs des FIFO des NI (cf. figure 5.8) des blocs fonctionnels appartenant à la voie descendante mais également en augmentant la taille des paquets de données et de crédits (cf. figure 5.9), on peut accroître les performances globales de la voie descendante. L'augmentation de la taille des paquets circulant sur le réseau n'est possible que pour les cas où les profondeurs de FIFO des NI ont été augmentées, sinon un blocage des communications apparaît immédiatement.

TFC1		**TFC2**		**Dem OFDM1**		**Dem OFDM2**		**MIMO ch est**		**MIMO dec**		**Egaliseur**		**CDMA**	
Entrée	Sortie	Entrée	Sortie	Entrée	Sortie	Entrée	Sortie	Entrée	Sortie	Entrée	Sortie	Entrée	Sortie	Entrée	Sortie
1280	1280	1280	1280	1280	672	1280	672	672	672	672	672	64	64	672	16

FIG. 5.8 – Profondeurs de FIFO modifiées pour la voie descendante dans le contexte d'étude mono-composant

TFC1		**TFC2**		**Dem OFDM1**		**Dem OFDM2**		**MIMO ch est**		**MIMO dec**		**Egaliseur**		**CDMA**	
Entrée	Sortie	Entrée	Sortie	Entrée	Sortie	Entrée	Sortie	Entrée	Sortie	Entrée	Sortie	Entrée	Sortie	Entrée	Sortie
8	64	8	64	64	64	64	64	64	32	64	32	32	16	16	8

FIG. 5.9 – Tailles des paquets modifiées dans les NI des blocs fonctionnels concernés pour la voie descendante dans le contexte d'étude mono-composant

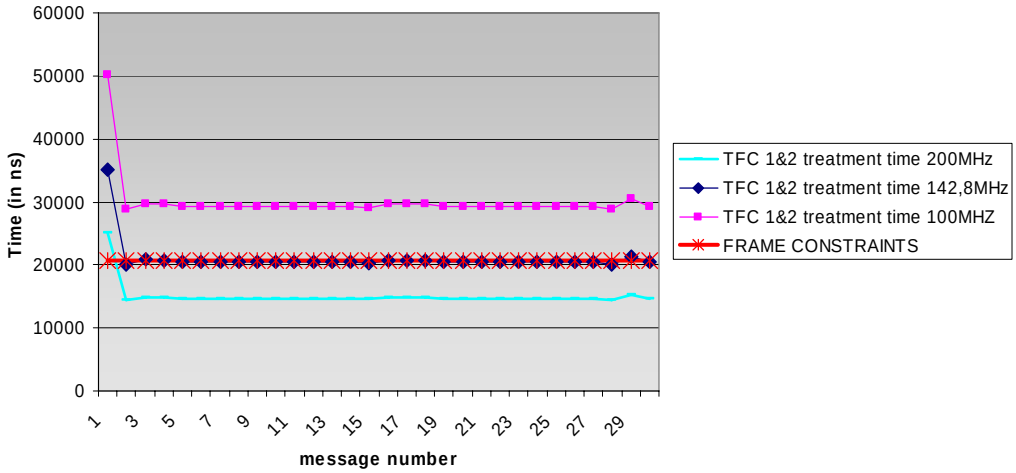

FIG. 5.10 – Latence globale des blocs fonctionnels TFC 1 et 2 pour une fréquence réseau variant avec des profondeurs de FIFO maximales et des tailles de paquets augmentées pour la voie descendante dans le contexte d'étude mono-composant

Les modifications d'architecture et de configuration des ICC et OCC des NI proposées ici, ont un impact direct sur les performances globales du réseau. La figure 5.10 présente les performances obtenues avec ces modifications pour des fréquences variant de 100MHz à 200MHz. On constate ainsi que l'on peut garantir les contraintes temps réel de l'application pour des fréquences supérieures ou égales à environ 150MHz.

Dans cette étude, nous avons montré les différentes étapes de nos explorations architecturales en modifiant différents paramètres du réseau (architecture et configuration du réseau) ayant un impact direct sur les performances globales. Ainsi, pour une fréquence donnée (par exemple 150MHz) on peut garantir les contraintes temps réel de l'application aussi bien dans le cas de la voie montante que de la voie descendante. Nous le verrons dans la section suivante traitant du contexte multi-composants, un ASIC intégrant un NoC a été conçu et réalisé par le CEA LETI. Or ces puces gravées dans une technologie en 0.13μm ne peuvent fonctionner à des fréquences supérieures à environ 175MHz (fréquence variant selon les lots issus de la fabrication). Par conséquent, nos études effectuées dans le contexte mono-composant se sont centrées sur des fréquences proches des 150MHz. L'ensemble de ces résultats ont fait l'objet d'un rapport technique interne au projet 4MORE permettant de guider les choix du WP6 en vue de la réalisation finale du SoC [89].

5.1.2 Contexte multi-composants

L'exploration NoC dans un contexte multi-composants a été nécessaire au sein du projet 4MORE car nous l'avons dit précédemment, un composant de type ASIC a été réalisé. Or, la réalisation d'un tel composant nécessite plusieurs mois dans sa phase de conception, dans sa phase de réalisation en fonderie et également dans les phases de tests et de validation des échantillons. Par conséquent, ces contraintes de temps ont engendrées des contraintes fortes concernant la réalisation des blocs VHDL des voies montante et descendante. Très tôt dans le projet, des choix de topologie et de contraintes de placement au sein de l'ASIC ont été effectués. Ainsi, les blocs déjà disponibles à ce moment ont été intégrés dans la puce afin de pouvoir lancer la fabrication le plus tôt possible. Il faut rappeler que le projet 4MORE s'inscrivait dans la continuité du projet MATRICE pour lequel des blocs VHDL avaient déjà été réalisés.

4MORE s'est différencié du projet MATRICE dans son cahier des charges mais également par la mise en œuvre de la technique MIMO. Compte tenu des investigations au niveau algorithmique du WP1 et des contraintes de délai imposées par la fabrication de l'ASIC, certains blocs n'ont pu être intégrés dans l'ASIC. Par conséquent, les blocs non disponibles ou pas encore réalisés ont été intégrés dans des composants reprogrammables de type FPGA afin d'étendre le réseau en dehors de l'ASIC et de pouvoir valider l'application complète à la fin de la fabrication. Cette plate-forme a pour objectif de servir de démonstrateur final à la fin du projet. Celle-ci est composée de deux ASIC et de deux FPGA. Afin de la mettre en œuvre correctement avec son média de communication NoC, des explorations architecturales ont été nécessaires afin de proposer des solutions de placement et de routage notamment concernant les blocs fonctionnels intégrés dans les FPGA.

Nous allons donc voir dans cette section le contexte de l'application pour lequel les contraintes du modèle SystemC ont dû être adaptées, puis nous verrons les résultats d'explorations effectuées au niveau des voies montante et descendante.

5.1.2.1 Contexte de l'application

Compte tenu des contraintes énoncées précédemment, certains blocs fonctionnels des voies montante et descendante se retrouvent intégrés dans les ASIC et d'autres dans les FPGA. Ces simulations au niveau NoC ont donc pour objectif de valider le partitionnement entre les deux familles de composants afin de guider les concepteurs du WP6 pour l'implantation finale. La chaîne algorithmique étant sous les investigations du WP1, celle-ci a subi de légères modifications par rapport au contexte mono-composant étudié précédemment. La figure 5.11 présente la chaîne algorithmique mise en place dans le démonstrateur final.

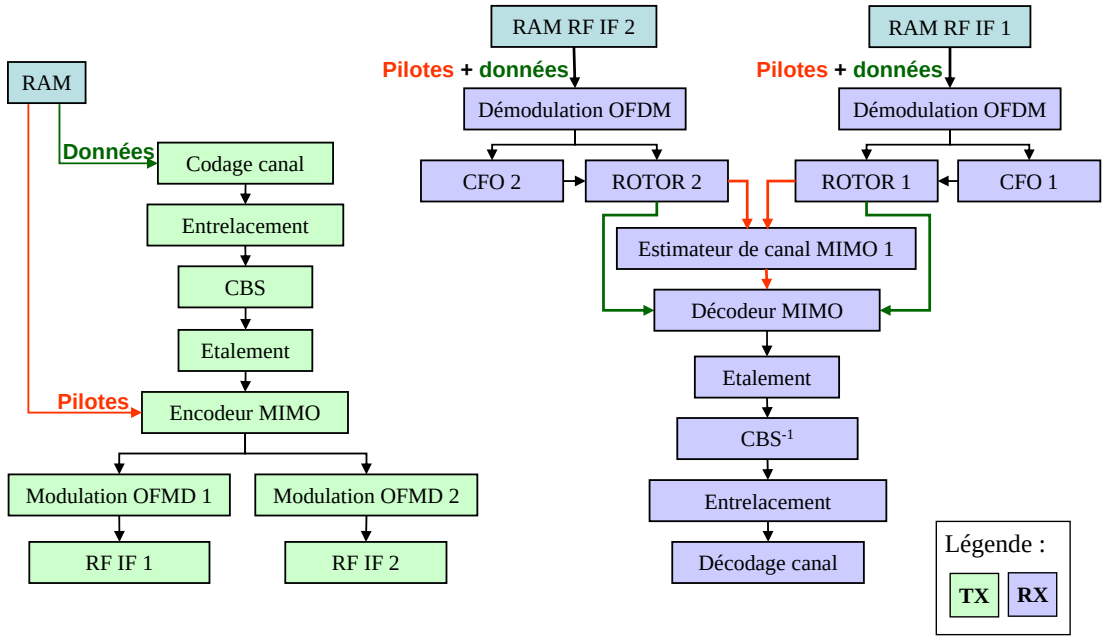

FIG. 5.11 – Diagramme fonctionnel des voies montante et descendante dans le contexte multi-composants

Les contraintes sur l'architecture du réseau et le placement des blocs fonctionnels sur celle-ci au sein de l'ASIC sont présentées sur la figure 5.12. Compte tenu de la mise œuvre de la technique MIMO dans la chaîne algorithmique et des contraintes imposées par l'architecture de l'ASIC, il est nécessaire d'avoir deux ASIC afin d'utiliser les blocs de chaque puce. Par conséquent, dans certains cas, des blocs des deux composants seront utilisés (cas de la modulation OFDM par exemple) et dans d'autres cas un seul bloc des deux ASIC ne sera utilisé (codage canal par exemple). Ces quatre composants seront interconnectés entre eux par l'intermédiaire d'entrées/sorties qui sont au nombre de deux pour les ASIC et de quatre pour les FPGA.

Nous avons utilisé le mode semi-automatique de notre flot de conception afin de pouvoir générer notre modèle SystemC avec les contraintes imposées sur les composants de type ASIC (descriptions de l'architecture, de l'application et des contraintes de l'une sur l'autre renseignées sous Excel).

Nous avons mené des études de charge des flux des communications transitant par les I/O communes entre les FPGA et les ASIC, afin de connaître la charge des liens servant d'I/O en vue de trouver un placement adéquat des blocs fonctionnels au sein des FPGA (seul composant dont la topologie et ses contraintes de placement sont modifiables). Nous verrons lors de la présentation des résultats de simulation que ces I/O sont en limite de bande passante concernant la voie descendante.

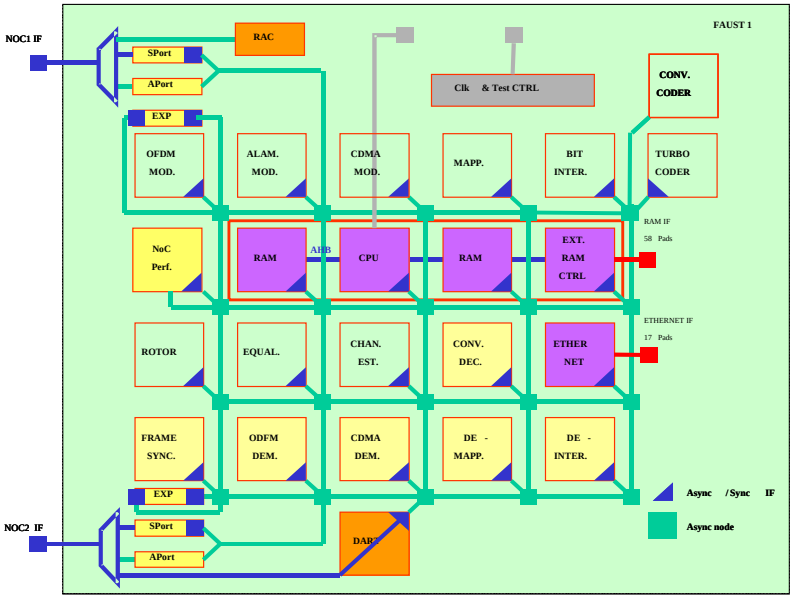

FIG. 5.12 – Architecture de l'ASIC FAUST

Les blocs fonctionnels non placés dans les ASIC possèdent un coût en surface important qui nécessite l'emploi de deux FPGA Virtex4 SX35-10 auxquels il faut ajouter le coût des routeurs du NoC. Nous le verrons dans la partie émulation présentée en fin de chapitre que le coût d'un routeur du NoC est proche des 1000 slices et que ce surcoût est à prendre en compte dans l'intégration des blocs fonctionnels sur FPGA. Le nombre de routeurs intégrés dans les FPGA est de 8 pour chacun. La figure 5.14 présente le contexte multi-composants mis en place.

Nous l'avons vu dans notre flot de conception, il est donné la possibilité au concepteur de décrire son application dans un contexte multi-composants. Par conséquent, nous avons créé une matrice globale de 64 routeurs ayant une dimension 8×8 recevant les quatre composants caractérisant l'application avec des consignes de routage au niveau des I/O de chacun (cf. figure 5.13).

Compte-tenu des technologies utilisées dans ces deux familles de composants, des limites d'utilisation en fréquence de ceux-ci apparaissent notamment au niveau du NoC. Par conséquent, les fréquences maximales d'utilisation de ces composants sont de 175MHz pour les ASIC et de 100MHz pour les FPGA. Nous allons donc présenter dans les deux sections suivantes les phases d'exploration que nous avons réalisées au niveau des voies montante et descendante.

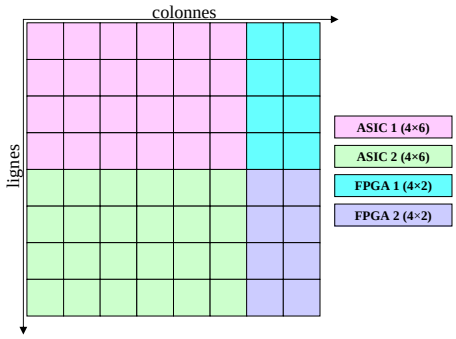

FIG. 5.13 – Modélisation de la structure multi-composant pour le flot de conception

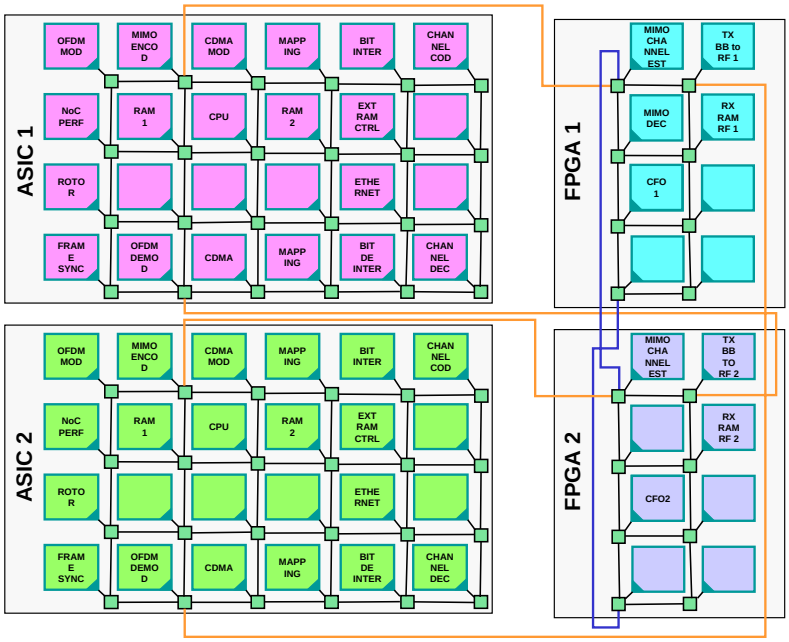

FIG. 5.14 – Architecture de la plate-forme du démonstrateur final du terminal mobile 4G

5.1.2.2 Exploration de la voie montante

Nous avons vu dans le contexte mono-composant, que l'augmentation de la profondeur des FIFO avait un impact direct sur les performances globales de l'application notamment en réduisant les latences des communications (le NI du bloc de traitement peut recevoir les données du prochain traitement pendant le traitement courant). Or ici dans le contexte multi-composants, seules les profondeurs des FIFO des blocs fonctionnels placés dans les

FPGA sont modifiables, celles de l'ASIC étant figées et fixées à 16 FLIT. Il faut cependant noter que concernant les modulations et démodulations OFDM présentent dans les ASIC, ces blocs fonctionnels possèdent une mémoire interne permettant de stocker 3 symboles OFDM. Ceci permet de recevoir le message entrant sur le premier élément, pendant qu'un symbole est en cours de traitement sur le deuxième élément et en même temps que le symbole traité précédemment est en cours d'émission vers le bloc fonctionnel ciblé.

Nous avons fait varier la fréquence de fonctionnement des composants de 83MHz à 250MHz pour des profondeurs de FIFO de 16, 1024 et 2048 FLIT. Les profondeurs de FIFO modifiées sont celles des blocs fonctionnels de la voie montante placés dans les FPGA et celles des modulations OFDM de l'ASIC. Les résultats obtenus sont présentés sur la figure 5.15.

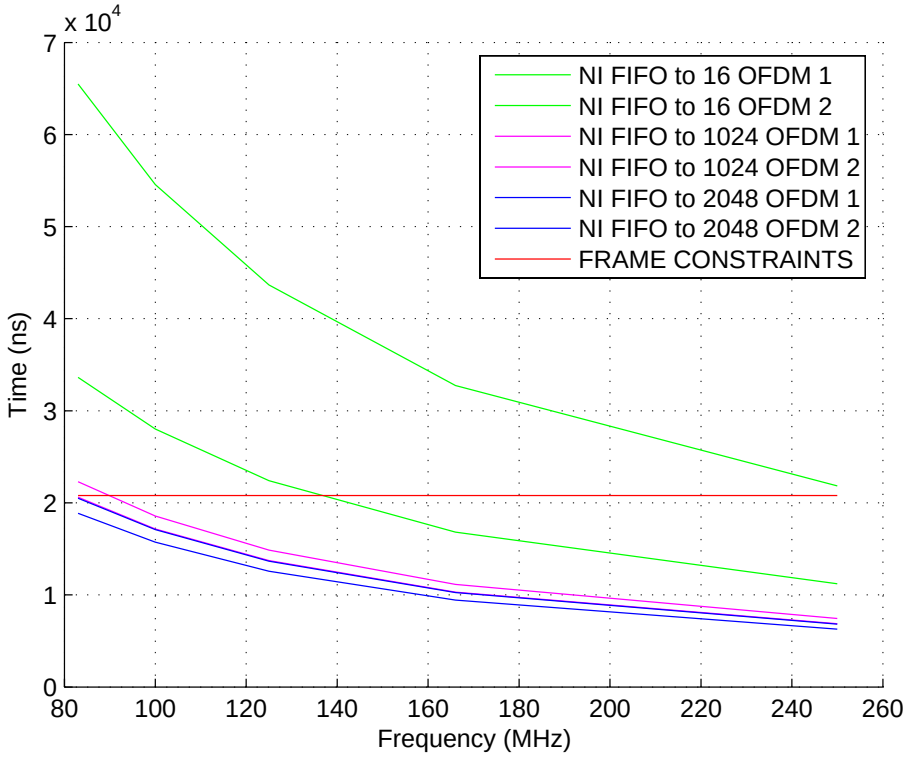

FIG. 5.15 – Latence globale des modulations OFDM pour des fréquences variant de 83 à 250MHz pour des profondeurs de FIFO de 16, 1024 et 2048 FLIT

On constate sur ces courbes de résultats de simulation que dans le cas de FIFO de petite taille, à savoir de 16 FLIT, les contraintes temporelles ne pourront pas être garanties pour toute la plage de fréquences observée (celle-ci incluant les fréquences de l'implantation réelle). On observe également une dérive de performances entre les deux branches des canaux MIMO. Cette dérive est due à l'émission des symboles OFDM de la modulation OFDM1 sur le même lien utilisé par l'encodeur MIMO émettant des données vers l'autre modulation OFDM 2 se trouvant dans le deuxième ASIC.

Par contre, on observe que les contraintes temps réel peuvent être garanties dans les cas où les profondeurs de FIFO ont été majorées. Plus particulièrement, dès lors que leur

taille est supérieure ou égale à 1024 FLIT, on constate que les contraintes temps réel sont respectées pour des fréquences d'utilisation supérieures ou égales à environ 100MHZ. Ainsi, la solution la plus adaptée pour l'implantation finale sur architecture hétérogène, pour la voie montante, est d'augmenter les profondeurs des FIFO des NI. Ici, il n'est pas conseillé d'augmenter les tailles des paquets car par cette méthode on rend artificiellement le réseau en "circuit switching" engendrant un accroissement de la probabilité de deadlock. Ce risque est déjà accru par la congestion des communications au niveau des liens des routeurs correspondant au I/O.

5.1.2.3 Exploration de la voie descendante

Ayant observé des congestions de communications au niveau de la voie montante et considérant le nombre de communications entre les composants pour la voie descendante, nous avons souhaité observer l'impact des profondeurs des FIFO pour une fréquence donnée. Ainsi, compte tenu des fréquences de fonctionnement des composants utilisés pour l'implantation finale (100MHz pour les FPGA et 175MHz pour les ASIC) nous avons choisi de fixer la fréquence de fonctionnement des 4 composants de notre modèle SystemC à 125MHz et de faire varier les profondeurs de FIFO de 16 à 2048 FLIT. Les résultats obtenus sont présentés sur la figure 5.16.

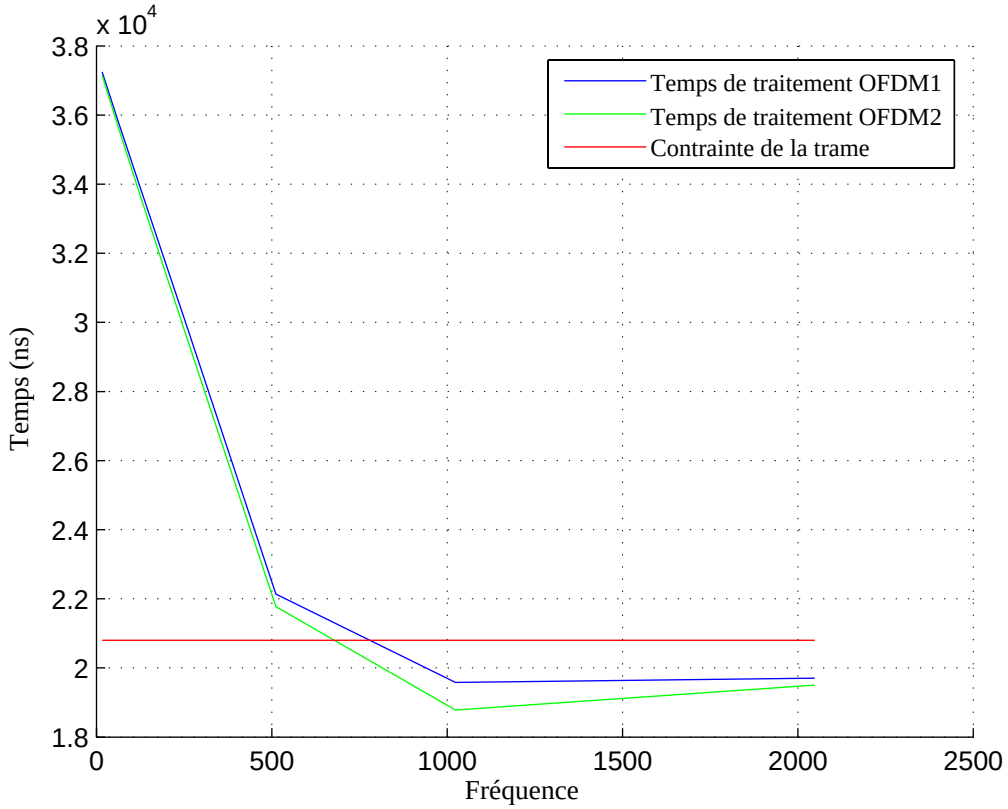

FIG. 5.16 – Performances de la démodulation OFDM vs. taille des FIFOs de la NI des FPGA pour un NoC à 125MHz

Comme dans la voie montante seules les profondeurs des FIFO des NI des FPGA et celles des démodulations OFDM de l'ASIC sont modifiées lors de ces simulations. On observe par ces résultats que les contraintes temps réel de l'application pour une fréquence globale de 125MHz ne pourront être atteintes que pour des profondeurs de FIFO supérieures à 1024 FLIT. Par contre, il est à noter que pour des tailles supérieures les gains en performances sont faibles. Compte-tenu du coût matériel engendré par ses tailles de FIFO, nous limiterons celles-ci à 1024 FLIT.

De la même manière nous avons souhaité observer l'impact de la fréquence pour deux cas de profondeur de FIFO au niveau des NI : 16 et 1024 FLIT. La figure 5.17 montre les résultats obtenus dans ces conditions.

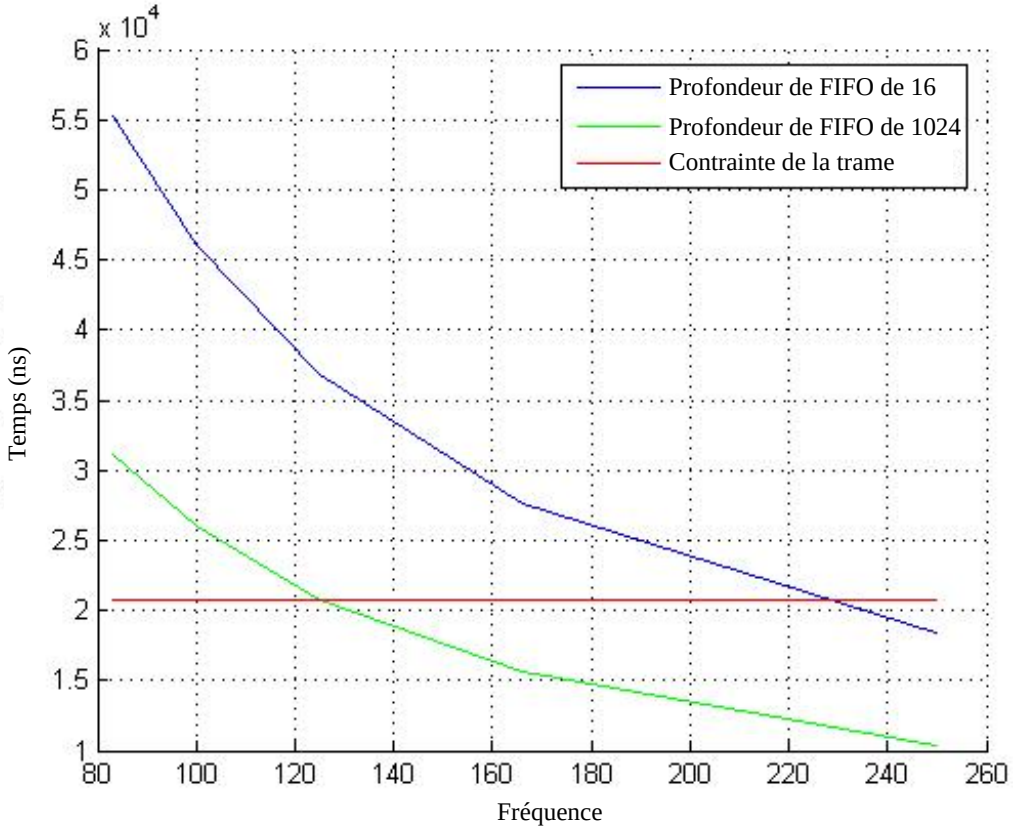

FIG. 5.17 – Performances observées sur les démodulations OFDM pour des fréquences variant de 83 à 250MHz pour des FIFO de 16 et 1024 FLIT

Ces résultats montrent que l'application respectera le cahier des charges si et seulement si la fréquence appliquée sur les 4 composants est supérieure ou égale à environ 130MHz pour des profondeurs de FIFO de 1024 FLIT. Dans le cas de taille de FIFO de 16, les contraintes temps réel de l'application ne pourront être respectées qu'à partir de 240MHz minimum.

Lors de l'implantation réelle, les fréquences de fonctionnement réelles ne seront pas identiques et équivalentes à 130MHz. Les puces reçues et testées par le CEA peuvent atteindre une fréquence maximale d'environ 166MHz. Concernant les FPGA, la fréquence

maximale applicable est de 100MHz. Par conséquent, nous avons souhaité améliorer les performances obtenues ci-dessus en analysant plus précisément les flux de communication introduisant ces latences. Nous avons constaté que le point critique se trouvait au niveau du décodeur MIMO qui pour chaque symbole complexe associe une paire de coefficients de canal fournit par l'estimateur de canal. Cette constatation a été possible au moyen de l'observation des fichiers de performances de chaque routeur du réseau donnant une image de la congestion des liens.

Par conséquent, afin d'accroître le débit entrant de ce bloc, nous avons instancié deux nouveaux bloc du même type qui vont recevoir des informations mais sans en émettre. On permet en quelque sorte à un bloc fonctionnel d'être connecté sur 3 routeurs en même temps. La topologie ainsi modifiée est présentée sur la figure 5.18.

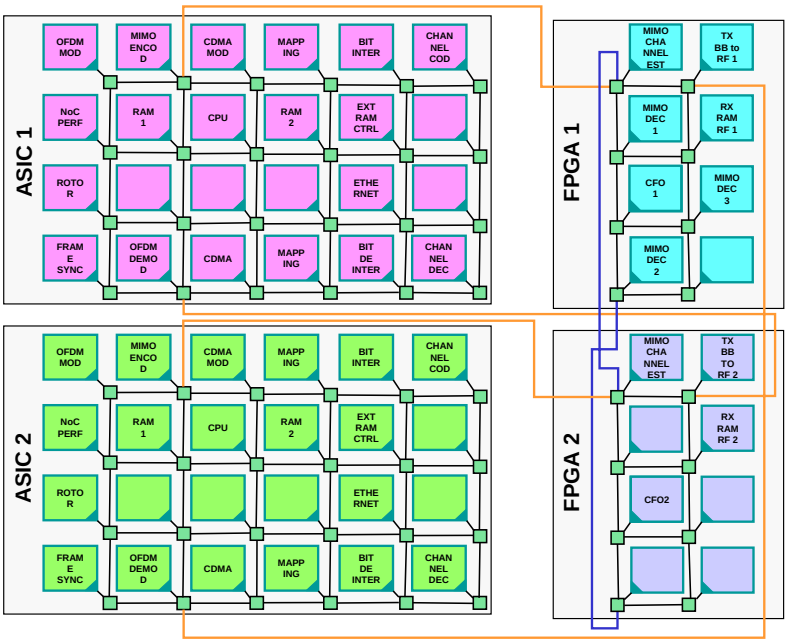

FIG. 5.18 – Architecture de la plate-forme du démonstrateur final du terminal mobile 4G modifiée (3 ports pour le MIMO décodeur)

Les résultats ainsi obtenus avec cette nouvelle topologie sont présentés sur la figure 5.19. On observe grâce à cette modification de topologie que l'on obtient un gain en performances d'environ 10MHz par rapport à la topologie précédente. Dans ces conditions, on ne peut pas affirmer que les contraintes du cahier des charges seront respectées lors de l'implantation réelle, mais nos explorations tendent à montrer que l'on s'en rapproche.

Nous présentons également sur la figure 5.20, les bandes passantes observées sur les liens du routeur 6 pour des fréquences respectives de 100 et 166MHz (routeur sur lequel est connecté l'estimateur de canal MIMO et ayant deux liens convertis en I/O). Sachant que dans le cas d'un fonctionnement du réseau à 100MHz on a une bande passante théorique de 3200Mbits/s, on observe que l'on est régulièrement en limite de saturation.

Or si l'on se place dans le contexte d'une fréquence de 166MHz, la bande passante théorique des liens de chaque routeur est dans ce cas de 5312Mbits/s. On peut observer

dans ce cas précis que certains liens exploitent de nouveau la totalité de la bande passante. Ce phénomène montre bien que la limite minimale de fréquence de 120MHz observée précédemment ne pourra vraisemblablement pas être diminuée au vu de la saturation des liens des I/O. L'ASIC réalisé dans le cadre du projet 4MORE offre une fréquence maximale de fonctionnement de 166MHz qui par conséquent nous permet une marge de manoeuvre assez souple compte tenu de la limite en fréquence de 120MHz obtenue en simulation.

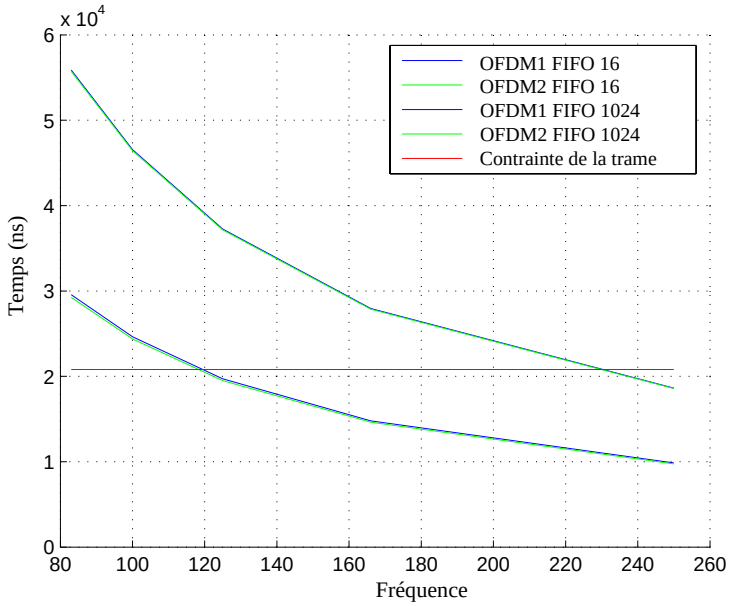

FIG. 5.19 – Performances des démodulations OFDM (FIFO de 16 et 1024 FLIT) avec un décodeur MIMO possédant trois ports d'entrée

La plate-forme matérielle a été validée et testée par le CEA LETI en charge de sa réalisation et de son exploitation. Les premiers tests se sont portés sur la validation de la voie montante dans un contexte SISO puis dans un contexte MIMO. Les premiers résultats ont montré que les performances obtenues remplissaient le cahier des charges en terme de contraintes de temps. L'expérimentation de la voie descendante est en cours d'exploration et nous n'avons à ce jour aucune information concernant des résultats d'expérimentations.

Les choix au niveau de l'architecture du réseau et de sa configuration proposés ci-dessus montrent que l'on peut atteindre les performances de l'application avec une structure de communication à base de NoC ayant une QoS de type BE. Nous avons vu que les choix proposés dans le cadre d'un contexte multi-composants pour la voie descendante ne permettent pas de certifier que les contraintes temps réel seront respectées lors de l'implantation. Par contre, il faut noter que toutes nos simulations sont basées sur des estimations pire cas. Or notre modélisation des blocs fonctionnels de la chaîne algorithmique a été réalisée en estimant leurs contraintes de performances plutôt à la hausse de façon à légèrement surévaluer les besoins matériels. Par conséquent, lors de la validation par implantation et dans des conditions d'utilisation n'étant pas nécessairement tout le temps dans de conditions pire cas, l'application respectera les conditions temps réel.

Le cahier des charges de cette application pour un terminal mobile 4G est d'offrir un débit d'information utile de 10Mbits/s pour une vitesse de 300Km/h et de 100Mbit/s pour une vitesse de 3Km/h. Or nous nous sommes placés dans les conditions pire cas d'un débit théorique maximal qui bien souvent n'est jamais atteint dans la pratique du fait notamment des conditions du canal de propagation.

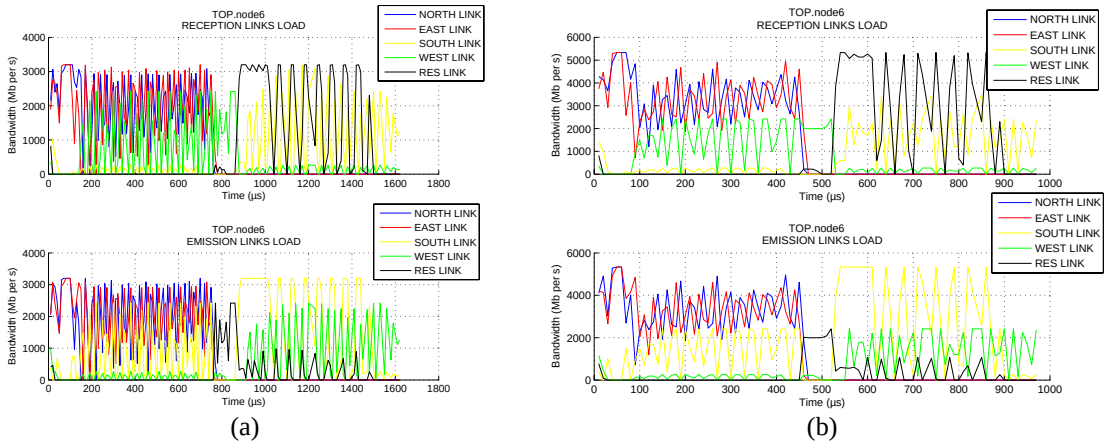

FIG. 5.20 – Performances observées sur les liens du routeur 6 de cette topologie pour une fréquence de 100MHz (a) et 166MHz (b)

Nous allons voir dans la section suivante, les développements que nous avons effectués autour d'une plate-forme de prototypage à stockage rapide.

5.2 Plate-forme de prototypage à stockage SCSI rapide

5.2.1 Présentation

Comme précisé en début de ce chapitre, l'intérêt de cette plate-forme de prototypage était de pouvoir développer, à des fins d'émulation, une application sur un composant de type FPGA en communiquant les données issues des traitements par un média de communication de type bus PCI. Cette plate-forme a pour objectif de proposer un prototypage rapide d'un ou de plusieurs bloc de traitement décrit en VHDL sur une cible matérielle à base de FPGA.

Cette émulation offre la possibilité de stocker les données produites par le SoC au moyen d'un wrapper PCI placé sur FPGA en lui offrant une grande bande passante pour le stockage de celles-ci sur des disques durs. Les données produites par le bloc de traitement sont transférées vers un tampon mémoire situé en mémoire vive du PC. Les données de ce dernier sont stockées sur des disques durs à écriture rapide de technologie SCSI Ultra 320 par l'intermédiaire d'un contrôleur de gestion mémoire programmé en langage C. Cette interface logicielle gère le vidage du tampon mémoire en transférant les données dans des fichiers écrits sur des disques durs. Un schéma simplifié de cette plate-forme de prototypage est présenté sur la figure 5.21.

Cette plate-forme est donc composée de deux entités distinctes, l'une gérant la partie matérielle, l'autre gérant la partie logicielle. Nous allons donc voir brièvement dans les deux parties ci-dessous une description de ces deux entités.

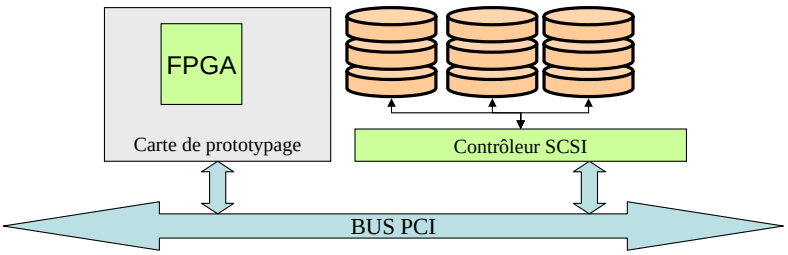

FIG. 5.21 – Schéma de la plate-forme de prototypage SCSI rapide

5.2.1.1 L'Architecture de la carte de prototypage

La carte de prototypage a été réalisée au laboratoire de façon à obtenir une carte simplifiée et dépourvue d'une puce gérant le protocole PCI mais pouvant recevoir des cartes filles par le biais de connecteur PMC. Elle avait également pour objectif de permettre des mesures de consommation électrique qui sont généralement impossibles sur les cartes proposées dans le commerce.

Cette carte possède un FPGA Xilinx Virtex II 4000-4. Celui-ci permet d'intégrer le bloc VHDL de traitement mais également l'IP permettant de gérer le protocole PCI. Un schéma bloc caractérisant la carte est présenté sur la figure 5.22.

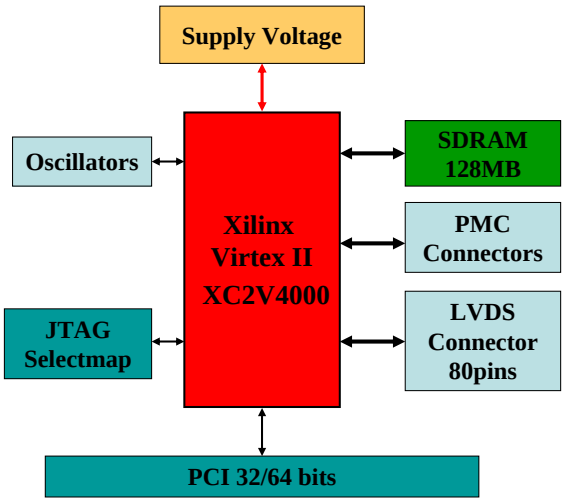

FIG. 5.22 – Schéma bloc de la plate-forme de prototypage

Nous avons précisé que les transferts des données produites par le bloc de traitement étaient écrites dans une zone mémoire allouée en mémoire vive du PC. Ces transferts

s'effectuent par l'intermédiaire du bus PCI. Par conséquent, l'IP PCI va être utilisée afin de transférer et gérer les données vers cette zone mémoire.

En effet, lorsqu'une allocation mémoire est effectuée en programmation, c'est l'OS qui va gérer les adresses physiques en RAM par l'intermédiaire du MMU (Memory Management Unit). D'un point de vue logiciel cette gestion des adresses est transparente pour l'utilisateur. Mais d'un point de vue matériel l'allocation mémoire est constituée de zones mémoires physiques non contiguës. En réalité, le tampon mémoire est constitué de plusieurs pages qui, elles, sont des zones mémoires contiguës. Ainsi, lorsque l'IP PCI doit transférer des données dans ce tampon mémoire, il est nécessaire de connaître les adresses physiques constituant le tampon. C'est pour cette raison que nous avons mis en place un contrôleur DMA (Direct Memory Access) de gestion mémoire associé à l'IP PCI afin de transférer les données vers l'allocation mémoire en fournissant les adresses des pages. Ce bloc de gestion mémoire possède une mémoire interne renseignant les différentes adresses physiques des pages de la zone allouée. Ce composant DMA va donc fonctionner en mode scatter/gather (ou fragmentation/rassemblement).

Cette zone mémoire est chargée au moyen d'un driver développé pour la carte au moyen du logiciel Windriver qui obtient les adresses des pages en communiquant avec le MMU. Enfin, afin de maintenir un débit constant pour le transfert des données, nous avons mis en place deux tampons mémoire permettant de basculer sur la deuxième zone lorsque la première est pleine (il s'agit donc d'un double buffer fonctionnant en ping-pong).

Nous allons donc voir dans la partie suivante le fonctionnement de la partie applicative ayant en charge de transférer les données inscrites dans la zone mémoire allouée vers les supports de stockage SCSI.

5.2.1.2 L'Application gérant les transferts vers les supports de stockage SCSI

Le bus PCI utilisé en largeur 32 bits ou 64 bits pouvant fonctionner à des fréquences de 33 ou 66MHz offre des bandes passantes théoriques importantes (cf. figure 5.23)

Largeur du bus	Fréquence	Taux de transfert
32	33MHz	132Mo/s
32	66MHz	264Mo/s
64	33MHz	264Mo/s
64	66MHz	528Mo/s

FIG. 5.23 – Bandes passantes théoriques maximales de la norme PCI

L'IP PCI que nous avons en notre possession fonctionne en mode 32 pour des fréquences de 33MHz ou 66MHz. Par conséquent, le débit maximal des transferts de données vers la zone mémoire seront de 264Mo/s pour une fréquence de 66MHz. Nous avons donc fait le choix de la technologie SCSI U320 car celle-ci cadrait dans les contraintes de bande passante de notre application tout en respectant les contraintes temps réel.

La couche applicative va donc transférer les données écrites dans les deux zones tampon sous forme de fichiers vers des disques durs. Ce stockage offrant la possibilité au concepteur par un traitement sous MatLab de comparer les résultats de l'implantation matérielle avec ceux obtenus par une chaîne décrite sous Simulink par exemple.

Les disques SCSI U320 possèdent des débits unitaires avoisinant les 80Mo/s. Il n'est donc pas possible d'atteindre les débits de 320Mo/s avec un seul disque. Les performances de 320Mo/s peuvent être atteintes lors de l'utilisation de plusieurs disques sur un même contrôleur SCSI. Par conséquent, nous avons mis en place 4 disques durs SCSI sur un contrôleur ADAPTEC 39320R en PCI-X fonctionnant à 100MHz. Nous avons mis en œuvre la technique RAID (Redundant Array of Independent Disks), technologie permettant de stocker des données sur de multiples disques durs, en général de manière redondante, afin d'améliorer certaines caractéristiques essentielles de l'ensemble en fonction du niveau de RAID choisi, qu'il s'agisse de la tolérance aux pannes, de l'intégrité des données, ou des performances de l'ensemble. Pour les besoins de notre application, nous avons voulu augmenter les performances de l'ensemble et notre choix s'est porté sur le RAID 0.

Le RAID 0 permet de regrouper plusieurs disques en un seul lecteur logique. Ainsi, lorsque l'on vient écrire des données sur ce lecteur, celles-ci sont découpées en N parties (N étant le nombre de disques physiques) chacune étant attribuée à un numéro de disque permettant une écriture simultanément sur chaque disque. On obtient un temps d'écriture ou de lecture théorique divisé par N (cf. figure 5.24).

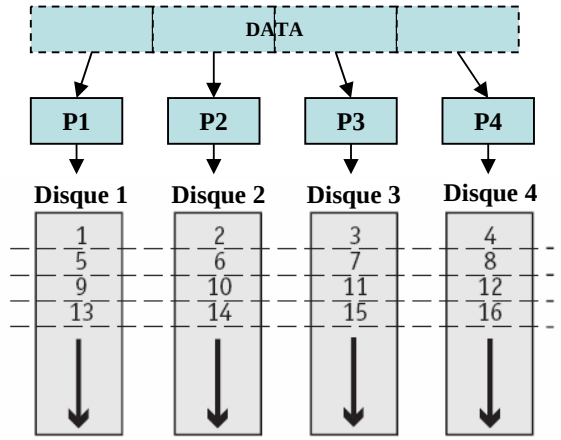

FIG. 5.24 – Concept du RAID 0 pour 4 disques

Cette technique peut être mise en œuvre matériellement par l'intermédiaire du contrôleur SCSI ou logiciellement par l'intermédiaire de l'OS. Dans cette partie applicative de la gestion des transferts de données depuis la zone mémoire vers les 4 disques SCSI, nous avons souhaité analyser les performances obtenues avec ces deux techniques. Durant ces explorations, nous avons voulu mettre en place une troisième technique de RAID 0 gérée par notre partie applicative. Nous avons réalisé un programme multi-processus, où chaque processus va gérer le transfert d'une partie de la zone tampon vers un disque. Un processus de contrôle est ajouté aux autres afin de superviser et synchroniser les transferts de chaque disque et de gérer le basculement d'une zone mémoire tampon vers une autre. Par cette troisième technique nous réalisons logiciellement le RAID 0 physique pris en charge au sein du contrôleur SCSI.

Ces trois techniques ont été testées successivement dans notre application de gestion des transferts de la mémoire vive vers les disques durs afin de les comparer en terme de

performances. Ceci afin vérifier si dans des conditions pire cas (264Mo/s sur le bus PCI) notre plate-forme peut garantir les contraintes temps réel. Les résultats ainsi obtenus dans ces trois cas sont présentés dans la section suivante.

5.2.2 Résultats d'exploration des trois techniques de RAID 0 mise en œuvre

Nous présentons dans cette section les résultats obtenus pour les trois techniques de RAID 0 testées. Nous avons souhaité faire varier la taille de l'allocation mémoire en RAM afin d'observer l'impact sur les performances des taux de transfert vers les disques en fonction de la technique de RAID 0 utilisée. Les résultats obtenus sont présentés sur la figure 5.25.

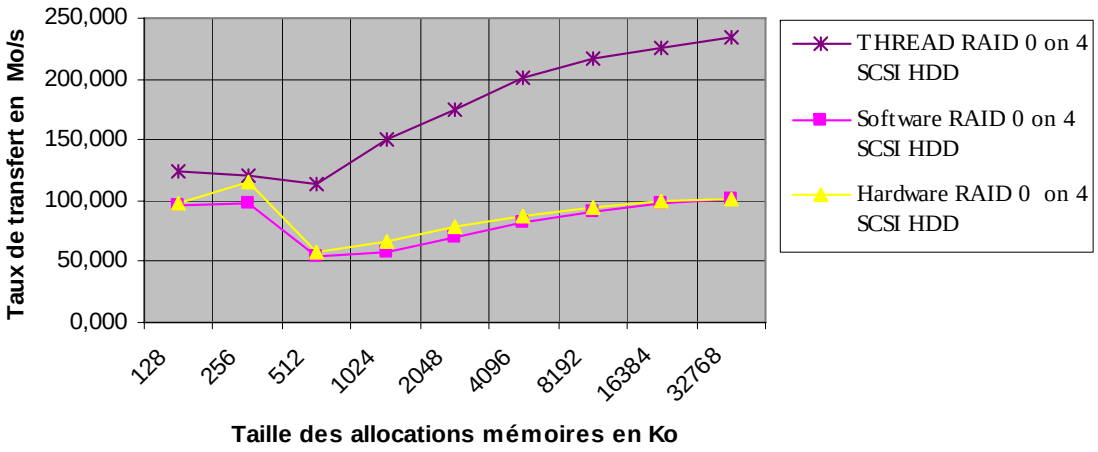

FIG. 5.25 – Comparaison de performances des trois techniques de RAID 0 en fonction de la taille de l'allocation mémoire

La courbe nommée RAID 0 Hard correspond au RAID matériel réalisé par le contrôleur SCSI tandis que celle nommée RAID 0 Soft correspond au RAID réalisé par l'OS. On peut constater parmi ces résultats que le RAID 0 Hard ou Soft ont des performances identiques quelque soit la taille de l'allocation mémoire mais ne permettent pas d'atteindre les conditions pire cas. Par contre, dans le cas d'un RAID réalisé par la méthode des processus que nous proposons, on constate que l'on arrive à des performances proches de 250Mo/s pour des allocations mémoires de 32Mo.

Afin d'améliorer ces performances, nous avons souhaité accroître le nombre de disques gérés par la technique de RAID. Nous avons mis en place deux autres disques durs de la même technologie sur le même canal de la carte contrôleur SCSI (la norme SCSI limitant le nombre de périphériques sur un même canal à 7).

Nous avons donc gardé les mêmes conditions d'expérimentation que celles mises en place dans les résultats présentés avec 4 disques. Par contre, compte tenu des résultats obtenus précédemment, nous avons observé des performances pour une technique de RAID 0 par processus. Les résultats obtenus sont présentés sur la figure 5.26.

On peut constater par ces résultats que l'ajout de deux disques durs permet d'obtenir un gain de taux de transfert supérieur à 30 Mo/s dans le cas d'expérimentation à 6 disques

comparé à celui avec 4 disques. Avec un RAID 0 par processus constitué de 6 disques, on obtient un taux de transfert qui permet de garantir le respect des contraintes temps réel de la plate-forme de prototypage dans les conditions pire cas.

Les résultats que nous venons de présenter ont été effectués sans la carte PCI. Les données transférées depuis la zone mémoire sont fictives mais permettent de montrer les performances que l'on peut atteindre avec ces différentes techniques de RAID 0. Nous avons effectué les mêmes tests avec la carte PCI connectée sur le bus PCI afin de vérifier si les performances étaient identiques. Nous avons obtenus les mêmes performances pour les cas d'allocations mémoires de taille inférieure à 256Ko. Pour des tailles d'allocation mémoire supérieures à 256Ko, nous n'avons pu effectuer de tests dû fait de la limitation du nombre de BlockRam disponibles dans le FPGA.

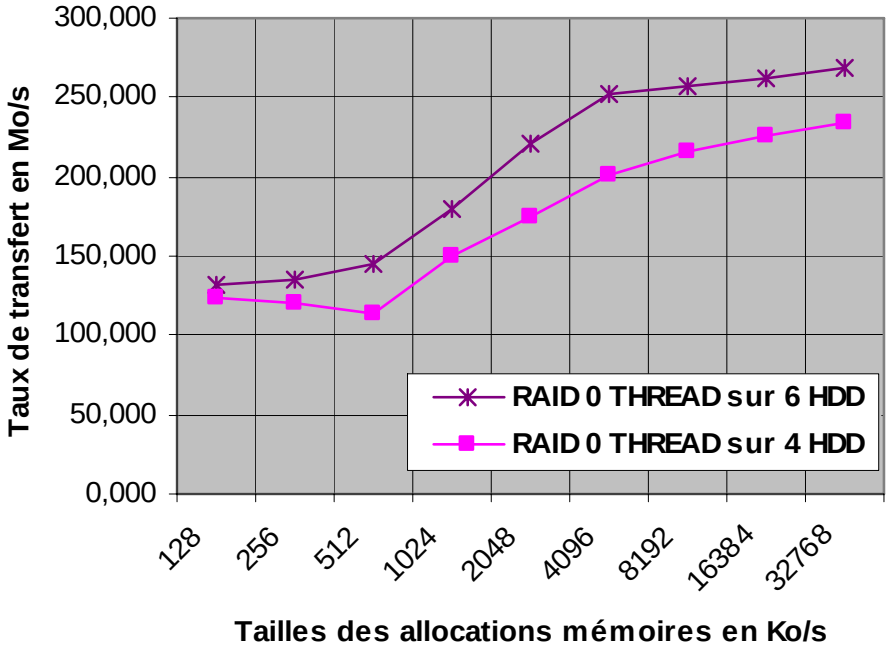

FIG. 5.26 – Comparaison de performances de la techniques RAID 0 par processus en fonction de la taille de l'allocation mémoire pour 4 et 6 disques durs

Il faut rappeler que le contrôleur de gestion mémoire (cf. section 5.2.1.1) possède une mémoire qui permet de stocker les adresses de la table des pages de l'allocation mémoire en RAM. Cette mémoire doit être assez grande pour pouvoir recevoir toutes les adresses des pages constituant les tampons mémoire.

Or lorsque les allocations mémoires deviennent importantes, cette zone mémoire croit de la même façon. De plus, cette RAM est réalisée au moyen de BlockRam disponibles dans l'architecture des FPGA qui par l'intermédiaire du CoreGen de ISE permettent de créer une mémoire de la taille souhaitée. Nous sommes arrivés en limite du composant avec une utilisation de plus de 92% de ces BlockRam pour gérer nos transferts pour un buffer de 256Ko. Cette limitation entraînant inévitablement des restrictions sur le bloc de traitement en test sur le FPGA qui peut lui aussi avoir besoin de ces éléments mémoires. Cette

solution n'est pas appropriée compte tenu du compromis à faire entre les taux de transfert de la mémoire vers les disques et les besoins en ressources mémoires au niveau du FPGA pour le contrôleur de gestion mémoire. Cette solution mémoire par le biais des BlockRam n'est pas idéale dans le cadre d'une plate-forme d'évaluation d'un SoC. Une alternative a cette limitation de BlockRam serait d'utiliser une mémoire SDRAM externe au composant permettant de stocker l'ensemble des adresses des pages composant l'allocation mémoire. Cette mémoire externe au FPGA permettrait également d'augmenter la taille des deux buffers en mémoire vive et disposer de meilleurs performances en terme de débit pour le stockage sur disques comme nous l'avons vu sur la figure 5.25. Il faut également noter que les résultats d'exploration présentés ici ont été réalisés dans des conditions idéales. En effet, les transferts de données ont été réalisé avec un seul élément communiquant sur chaque bus, allouant la totalité de la bande passante du bus pour ces transferts. Nous avons constaté des chutes de performances significatives (30 à 50 %) lorsqu'un autre élément sur le même bus effectuait des requêtes auprès de l'arbitre, engendrant un partage des ressources. Cet inconvénient montre bien la limitation des topologies bus pour des SoC intégrants plusieurs dizaines de blocs de traitement (logiciels ou matériels).

Nous allons finir ce chapitre par la présentation des résultats d'implantation d'un NoC 2×2 sur FPGA issu de notre flot de conception et d'exploration pour NoC.

5.3 Exemple d'implantation matérielle d'un NoC 2×2

5.3.1 Présentation

L'exemple d'émulation sur plate-forme reconfigurable à base de FPGA que nous présentons ici est un NoC à 4 routeurs de dimension 2×2. Ce NoC, de petite dimension, n'est pas nécessairement réaliste compte tenu des SoC actuels, mais l'idée développée ici est de démontrer la faisabilité d'une implantation du NoC FAUST sur FPGA avec la modélisation générique du comportement de notre UT présentée au chapitre 4 (cf. section 4.5.2).

Nous avons présenté l'architecture générique de l'UT constituée de différents compteurs en entrée et en sortie afin de créer des générateurs de trafics sur les FIFO des NI. L'intérêt de cette modélisation, nous l'avons dit, réside dans la possibilité de caractériser ces UT de manière logicielle. Ainsi, un modèle générique identique d'un point de vue matériel appliqué à tous les blocs de traitement pourra avoir un comportment différent au moyen de commande de configuration logicielle.

De cette manière, on obtient un gain de temps de conception important qui permet de modifier les comportements des UT sans avoir à re-synthétiser l'ensemble de l'architecture (concept présent dans le flot de conception EDK de Xilinx). Cette émulation permet de valider sur plate-forme matérielle le modèle NoC créé, simulé et exploré au niveau SystemC. On offre au concepteur la possibilité de poursuivre la validation du SoC en augmentant le jeu de tests en VHDL (cette exploration sur plate-forme matérielle étant plus rapide que les simulations). Le concepteur pourra valider les performances de débit par rapport aux contraintes temps réel avant de passer à l'étape de réalisation finale intégrant les vrais blocs de traitement. Les perspectives de ces travaux de thèse pourraient être de constituer une bibliothèque d'IP ayant leur schéma de modélisation simplifié et associé qui, une fois la validation au niveau SystemC effectuée, l'émulation sur plate-forme validée, pourrait être utilisée afin de générer directement le design final avec les véritables blocs VHDL.

Cette méthode est celle mis en place dans le flot proposé par la société Artemis dans les outils commerciaux qu'elle propose.

Nous avons également vu que pour ce réseau, il était nécessaire d'avoir un processeur de contrôle permettant de configurer et gérer le NoC. Pour l'émulation sur plate-forme nous avons choisi d'utiliser le cœur de processeur MicroBlaze de Xilinx comme processeur de contrôle. Il a donc été nécessaire de réaliser un adaptateur de protocole ("wrapper") entre celui-ci et le NoC FAUST afin de pouvoir l'interconnecter et l'utiliser.

Nous allons donc présenter dans un premier temps le MicroBlaze et le wrapper qui lui a été associé. Ensuite nous présentons la structure globale qui a été implantée. Et enfin, nous présentons les résultats obtenus lors de la validation au moyen de l'outil ChipScope proposé par Xilinx pour analyser les signaux du design en fonctionnement par le biais du cable JTAG (Joint Test Action Group).

5.3.2 Le processeur de contrôle : le MicroBlaze

Le MicroBlaze est un cœur de processeur RISC (Reduced Instruction-Set Computer) 32 bits développé par la société Xilinx et disponible avec l'outil de conception EDK. L'outil EDK est un logiciel conçu pour développer des applications utilisant un processeur logiciel (MicroBlaze) ou un processeur matériel (PowerPC) sur FPGA, suivant le type de FPGA utilisé. L'intérêt de ce processeur est son faible coût en surface sur FPGA (environ 1000 slices) et la possibilité de l'interfacer rapidement avec des blocs IP.

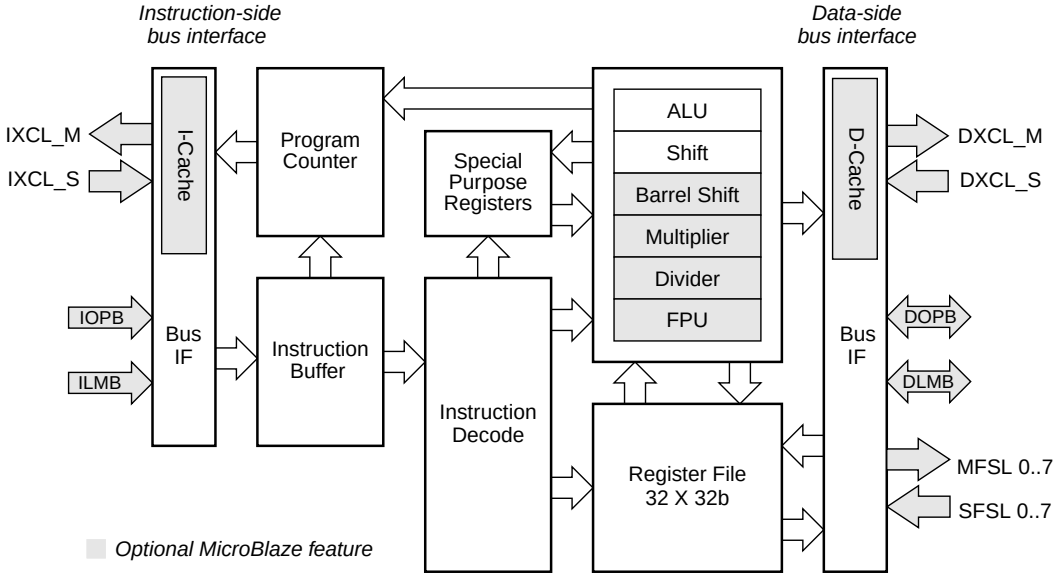

FIG. 5.27 – Schéma bloc de l'architecture du MicroBlaze

Afin d'exploiter ce processeur, nous avons dû réaliser un wrapper. Pour connecter ce dernier avec le MicroBlaze, deux types de connexions s'offraient à nous : le bus OPB ou les ports FSL. Le bus OPB est un bus propriétaire de IBM également utilisé dans les processeurs PowerPC présents dans certaines versions de FPGA. Ce bus nécessite la

réalisation d'une IP standard OPB avec plusieurs signaux qui ne sont pas nécessaires pour notre implantation.

Nous avons donc opté pour l'utilisation de ports FSL pour leur simplicité de mise en œuvre mais également pour leur mode de fonctionnement basé sur des échanges par FIFO similaire à celui du NoC. L'architecture du MicroBlaze est présentée sur la figure 5.27.

Les ports FSL sont au nombre de 8 paires par processeur. La largeur du bus est sur 32 bits et des routines en C permettent de lire ou écrire des données dans les FIFO du port. Ainsi, créer un wrapper sur ce port est assez simple puisque qu'il suffit de lire ou écrire des mots dans une FIFO en vérifiant l'état de celle-ci (drapeau plein ou vide). La figure 5.28 montre l'architecture d'un port FSL.

Le processeur de contrôle ayant pour rôle de gérer et configurer le réseau, l'ensemble des commandes va être écrit dans la FIFO et interprété par notre wrapper afin d'envoyer directement les signaux de requêtes sur les ports du routeur. La NI du MicroBlaze est donc réduit à un wrapper. Les différentes commandes à envoyer vont donc permettre de :

- configurer les ICC
- configurer les OCC
- configurer la FSM de l'UT
- configurer les registres des compteurs de génération de trafic dans les UT
- valider les configurations d'OCC
- valider les configurations d'ICC

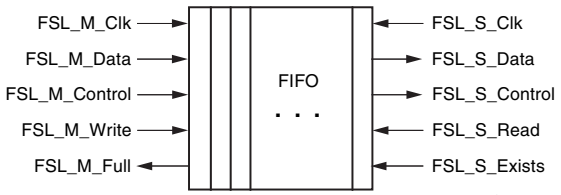

FIG. 5.28 – Schéma bloc de l'architecture d'un port FSL

Par conséquent, le programme C va utiliser des routines permettant de communiquer sur les ports FSL (pour une écriture : *microblaze_ bwrite_ cntlfsl(valeur,0)*). Il va venir écrire les données héxadécimales stockées en RAM sur le port FSL dans l'ordre de la liste énoncée ci-dessus. Un exemple de codes héxadécimaux pour une UT est présenté sur la figure 5.29.

5.3.3 Description de l'exemple implanté

L'exemple que l'on souhaite implanté ici est un NoC de taille 2×2. Compte tenu du fait que le réseau nécessite un processeur de contrôle, il ne reste que trois routeurs disponibles pour connecter des ressources de traitement. Nous avons donc mis en place trois UT dont leurs dépendances de données et leur placement sur le NoC sont présentés sur la figure 5.30.

Le code VHDL équivalent à cette application a été généré manuellement en intégrant le modèle générique de notre UT permettant de modéliser une RAM ou un bloc de traitement comme le montre cet exemple. Par conséquent, l'ensemble des paramètres de configuration nécessaire à la configuration et à la validation du réseau est renseigné dans le code qui

va être exécuté par le MicroBlaze. Afin de générer la structure du réseau en utilisant le MicroBlaze, nous avons utilisé le logiciel EDK version 8.1.02. La validation fonctionnelle a été réalisée sous Modelsim 6.0d afin de vérifier le bon fonctionnement du réseau avant de lancer la synthèse et le placement routage sous ISE version 8.1.03.

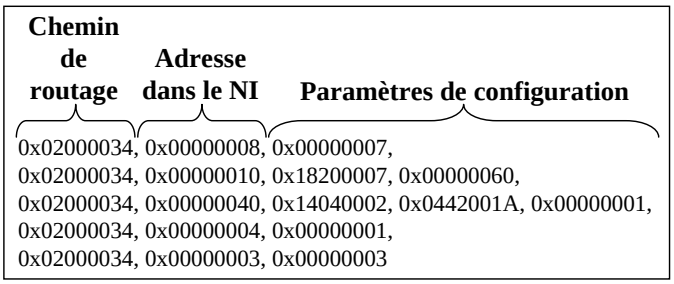

FIG. 5.29 – Exemple de codes héxadécimaux envoyés par le CPU à une ressource

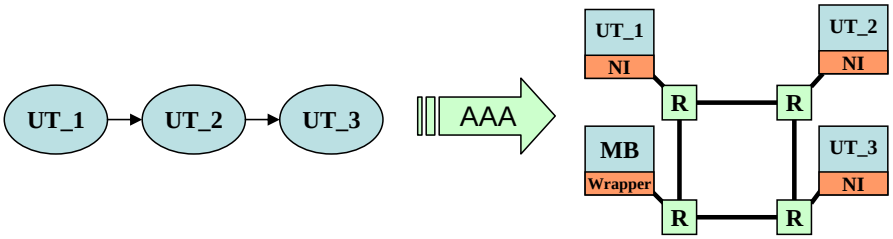

FIG. 5.30 – L'architecture réseau implanté sur FPGA

Cette implantation a été effectuée sur un Virtex4 SX35-10 d'une carte de développement de la société Nallatech. Afin d'observer le bon fonctionnement du réseau lors de l'implantation sur carte, nous avons utilisé Chipscope Pro version 8.18.2. Nous allons voir dans la section suivante les résultats obtenus.

5.3.4 Résultats

Compte tenu du flot de conception présenté au chapitre 4, nous avons modélisé cet exemple en SystemC en mode semi-automatique, mais étant donné la simplicité de l'application, et n'ayant pas de contraintes temporelles particulières, les critères de dimensionnement de l'architecture sont restés les plus faibles possibles. Ainsi, les FIFO des NI ont toutes été fixées à 16 FLIT et chaque NI ne possède qu'une seule FIFO en entrée et une seule FIFO en sortie, le contexte de l'application n'en nécessitant pas plus. Notre flot n'intégrant pas encore la génération automatique du code VHDL, nous avons donc réalisé manuellement cette génération du code. Il faut préciser que le code VHDL du NoC FAUST nous a été livré en même temps que le code SystemC sous NDA, celui-ci étant utilisé lors de la réalisation des ASIC.

La synthèse et le placement routage du design ont donc été effectués à l'aide de l'outil de synthèse ISE de Xilinx. Les résultats de synthèse nous ont conduit à l'obtention d'un

design ayant une fréquence maximale d'utilisation de 60MHz avec un taux de remplissage du FPGA de 50% (environ 7000 slices sur les 15360). Les coûts respectifs des différents élément constituant cette architecture sont renseignés dans le tableau 5.1 ci-dessous.

Fonction	**Coût (slices)**	**Nombre**	**Coût total (slices)**
Routeur	1000	4	4000
MicroBlaze	1000	1	1000
NI+UT générique	500	3	1500

TAB. 5.1 – Tableaux récapitulatifs des résultats de synthèse d'un NoC de dimension 2×2

Par ailleurs, afin de vérifier le bon fonctionnement du SoC comparativement aux validations effectuées en amont par le biais de notre flot de conception en SystemC, nous avons souhaité utiliser l'outil ChipScope afin d'analyser les différentes communications au sein de ce design. Cet outil offrant l'avantage de pouvoir analyser et étudier différents signaux directement à l'intérieur du design et de restituer les différentes transactions sous forme de chronogramme sur PC au moyen de la connection JTAG faisant le lien entre la plate-forme et le PC. Ainsi, les différentes transactions sur les différentes signaux analysés sont présentés dans la figure 5.31 ci-après.

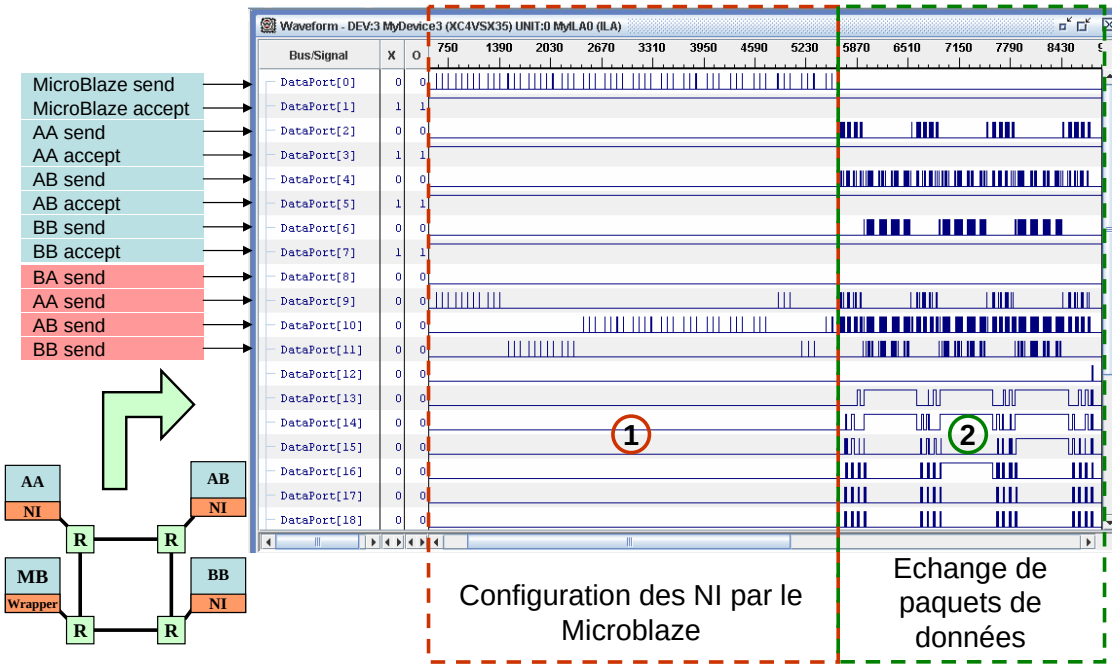

FIG. 5.31 – Analyse des transactions de paquets de données et de crédits avec l'outil Chipscope

Ce chronogramme nous a permis de vérifier les deux phases caractérisant le fonctionnement de ce réseau :

1. phase d'initialisation : chargement des configurations et validation de celles-ci par le processeur de contrôle
2. phase de fonctionnement : échanges de données entre les différentes UT au moyen des liens de communications entre routeurs du réseau

Par cette implantation, nous avons donc validé l'architecture générique de l'UT proposée, ceci nous permettant de rendre possible la mise en œuvre de la génération automatique du code VHDL équivalent au SystemC en utilisant une description en langage XML. Cependant, ces premiers résultats d'implantation nous ont également permis de mettre en évidence la nécessité d'optimiser le code VHDL du routeur FAUST afin de pouvoir envisager l'implantation de NoC de plus grande dimension (limitation actuelle à 8 routeurs).

5.4 Conclusion

Nous avons présenté dans ce chapitre les différents résultats de simulations et d'expérimentations obtenus durant ces travaux de thèse. Ces contributions nous ont permis d'illustrer les possibilités offertes par le flot de conception proposé afin d'explorer plus rapidement l'espace des solutions architecturales. Ces explorations ont été réalisées principalement dans le cadre de nos contributions au projet Européen 4MORE pour lesquels trois rapports techniques ont été réalisés [88, 89, 90]. Ces contributions nous ont amené à proposer des choix d'architecture et de contraintes de placement dans deux contextes différents : mono et multi-composants. Ces choix ont permis de guider les concepteurs du WP6 lors de l'implantation sur la plate-forme servant de démonstrateur final.

Le contexte multi-composants abordé dans ces travaux de thèse est particulièrement nouveau et nous a permis notamment d'explorer l'architecture du démonstrateur du terminal mobile 4G dans sa version finale. Les choix proposés dans le cadre de ce contexte ont été effectués au moyen de notre flot de conception offrant la possibilité de réaliser des explorations dans une architecture à composants hétérogènes, chaque composant possédant ses propres contraintes (routeurs réservés, fréquences de fonctionnement, tailles de FIFO de NI imposées, ...).

De plus, nous avons également présenté dans ce chapitre des travaux effectués en début de thèse visant à proposer une plate-forme de développement matérielle à base de FPGA proposant un support de stockage rapide utilisant un bus PCI. Nous avons vu que celle-ci offrait de bonnes perspectives en terme de performances mais la limitation matérielle observée au niveau du FPGA nous a conduit à proposer une alternative passant par l'emploi d'une mémoire externe au FPGA de type SDRAM.

Pour finir, nous avons présenté un exemple d'émulation de NoC [91][92] intégrant notre modèle générique d'unité de traitement. Cette implantation offre la possibilité au concepteur de valider le modèle SystemC simulé par une émulation sur plate-forme matérielle. Les résultats obtenus sont très récents et nous ont permis de valider notre modélisation de l'unité de traitement. Compte tenu des performances obtenues, nous prévoyons d'optimiser le code source (modifications également effectuées par le CEA dans le cadre de la réalisation du démonstrateur où une implantation sur FPGA a été effectuée), notamment celui du routeur qui a été écrit pour une technologie ASIC, non optimisé pour les FPGA, afin d'améliorer les performances du NoC et réduire son coût en surface. A terme, nous prévoyons de générer automatiquement le code VHDL en sortie du flot de conception par le biais d'une description en langage XML. Cette génération est facilitée par la possibilité

de paramétrer logiciellement le comportement de chaque UT connectée aux routeurs du réseau. Cette phase s'accompagnera également de la génération automatique du programme C exécuté par le processeur de contrôle, en l'occurrence le MicroBlaze en exploitant les paramètres de configuration fournis dans le modèle SystemC.

Conclusions et perspectives

Conclusions

Les travaux menés durant cette thèse nous ont permis d'étudier et d'évaluer les performances d'un réseau sur puce dans le cadre d'une application radio-logicielle de 4^{e} génération dans le contexte du projet Européen 4MORE.

Le premier chapitre de ce manuscrit présente un état de l'art des réseaux sur puce. Un rappel des médias de communication déjà employés dans les systèmes sur puce actuels est fait afin de présenter les limites de ces médias de communication et de comprendre les motivations qui conduisent à utiliser les réseaux sur puce. Cette émergence des NoC est rendue possible notamment grâce à l'évolution de la technologie silicium offrant une plus grande capacité d'intégration. Ceux-ci permettent de simplifier l'intégration des IP et offrent une plus grande souplesse d'intégration ainsi que des bandes passantes plus grandes, nécessaires pour les applications actuelles et futures de plus en plus exigeantes sur ces critères. Cependant, comme toute nouvelle architecture, les NoC requièrent des efforts de recherches importants car leur mise en œuvre est plus complexe du fait de la nécessité de prendre en compte plusieurs paramètres dans leur phase de conception. Ainsi les concepteurs doivent intégrer dans leur flot de conception de nouveaux critères d'implantation comme : le protocole de communication, la topologie du réseau, les contraintes de placement des blocs IP, la qualité de service, le dimensionnement matériel des ressources. Ceci entraîne le besoin de synchroniser les échanges de données, de contrôler le traitement de ces données et d'estimer les performances de l'architecture avec son ou ses applications associées.

Ainsi, dans le second chapitre, nous avons présenté l'application radio-logicielle de 4^{e} génération et ses caractéristiques définies dans le cadre du projet 4MORE dans lequel nous étions impliqués. Nous proposons dans ce chapitre un modèle de représentation simplifié des blocs de traitement pouvant être connectés à un NoC. Ce modèle est celui adopté dans le modèle SystemC utilisé du réseau FAUST permettant d'effectuer des simulations du réseau en intégrant les spécifications de l'application. Par conséquent, chacun des éléments de la chaîne algorithmique de 4MORE est décrit selon ce même modèle.

Puis dans le chapitre 3, nous avons détaillé les caractéristiques du réseau FAUST et son flot de conception associé. Ce réseau sur puce est celui que nous avons utilisé lors de ces travaux de thèse dans le cadre de notre contribution au sein du WP4 du projet Européen 4MORE. Ce réseau est disponible en version SystemC TLM afin de réaliser des simulations haut niveau, mais il est également disponible en langage VHDL pour émulation sur plate-forme matérielle reconfigurable par exemple.

Puis, dans le chapitre 4 nous présentons le flot de conception que nous avons mis en œuvre afin de simuler une application sur le réseau sur puce FAUST au moyen d'une description en langage SystemC. Ce flot permet de générer automatiquement tous les paramètres de contrainte qui caractérisent un réseau sur puce afin de le simuler et de mesurer les performances du contexte spécifié par le concepteur. Nous avons été amené, dans le cadre du projet 4MORE, à inclure la notion de réseau multi-composants dans notre flot de conception, représentant le contexte de simulation du démonstrateur final du projet. Par ailleurs, nous avons intégré à ce flot une heuristique permettant de réaliser une Adéquation Algorithme Architecture (AAA) pour un NoC en qualité de service BE. Cette heuristique prend en compte deux critères majeurs qui sont le positionnement des blocs IP sur la matrice du réseau en choisissant les chemins de communications les plus courts tout en veillant à ce que la charge de tous les liens du réseau n'excède pas la bande passante théorique maximale. Ce flot fournit des résultats de simulation renseignant les performances des blocs de traitement mais également celles des liens des routeurs, permettant ainsi de vérifier si les contraintes temps réel de l'application sont respectées.

Les résultats de ces travaux de thèse sont présentés dans le chapitre 5. Nous avons présentés des travaux menés en début de thèse visant à analyser les performances d'une plate-forme de prototypage rapide utilisant un média de communication de type bus PCI. Ces travaux ont montrés les performances d'une topologie bus et ces limites en terme de scalabilité.

Puis, nous avons présenté les deux contributions majeures réalisées dans le cadre du projet 4MORE. L'application radio-logicielle utilisée emploie la technique MC-CDMA et SS-MC-MA associée à la technique MIMO (2×2). Ces deux contributions ont été réalisées dans un contexte mono-composant puis multi-composants (plate-forme matérielle du démonstrateur final). Par ces résultats, nous avons proposé des choix architecturaux permettant de remplir le cahier des charges de l'application en augmentant notamment les profondeurs de FIFO ou les paquets de données et de crédits ou bien en modifiant des chemins de routage. Ces choix architecturaux mais également de paramétrage des registres des interfaces réseau ont permis d'apporter des solutions aux concepteurs en charge de l'intégration des blocs IP dans le démonstrateur final du projet 4MORE. Pour finir, nous avons réalisé une émulation d'un réseau de dimension 2×2 sur une plate-forme reconfigurable à base de FPGA. Cette émulation permet de valider le modèle SystemC simulé de manière matérielle mais également d'accélérer les phases d'exploration en les réalisant matériellement par le biais de changement de configuration au sein du code exécuté par le processeur de contrôle.

Nous avons également souhaité tester l'intérêt de l'utilisation du logiciel Syndex [75] pour concevoir un réseau sur puce concernant les aspects de placement des blocs IP et de routage des communications entre ces blocs de traitement sur le réseau. Les résultats obtenus n'ont pas été satisfaisants pour un exemple simple pour lequel le coût des communications n'était pas minimisé. Ces résultats nous ont donc conduit à la mise en œuvre d'un outil de conception dédié aux architectures NoC dont nous avons présenté le flot associé.

Ce travail de thèse a permis de rendre plus flexible le flot de conception pour ce réseau et de l'améliorer en offrant d'autres services comme la méthode AAA ou encore l'émulation sur plate-forme matérielle. Ceci permettant d'accélérer les phases d'explorations et d'aider

le concepteur à trouver la meilleure adéquation possible, plus rapidement, par le biais des données issues des résultats de simulations obtenus via notre flot.

Pour finir, notre volonté était de répondre au mieux aux contraintes de notre contribution dans le cadre du projet 4MORE tout en anticipant les besoins futurs. Par conséquent, ces travaux de thèse ouvrent des perspectives sur les poursuites à leur donner et nous présentons celles-ci dans la section ci-après.

Perspectives

Ces travaux de thèse menés sur les réseaux sur puce étant nouveaux pour le laboratoire, les perspectives de ces travaux sont donc nombreuses. Lors de la présentation de nos résultats de simulation nous avons montré que l'accroissement des profondeurs des FIFO au niveau des interfaces réseau permettait d'obtenir des gains en performance intéressants, notamment concernant la réduction des latences dans les communications. Lors de ces explorations, nous avons augmenté les profondeurs des FIFO de plusieurs interfaces réseau de manière identique. Or il se peut que certaines d'entre-elles soient sur-dimensionnées et sous-utilisées. Ainsi, afin d'avoir la meilleure adéquation possible, tout en veillant à minimiser les coûts en surface et en consommation électrique, nous prévoyons de mettre en place un traceur pour chaque FIFO, celui-ci ayant pour rôle de donner des statistiques de remplissage des FIFO durant les simulations afin que le concepteur puisse ajuster les tailles des FIFO indépendamment les unes des autres. Et par conséquent, réduire les coûts en surface et indirectement en consommation électrique.

D'autre part, nous prévoyons de mettre en place une génération de code VHDL automatique du modèle SystemC simulé par le biais d'une description en langage XML. Cette génération automatique est possible grâce à la solution que nous proposons consistant à rendre générique les unités de traitement et de permettre leur configuration logiciellement. Cette étape est nécessaire afin de compléter notre flot de conception et d'offrir au concepteur plus de possibilités d'exploration et d'accélérer celles-ci par une émulation matérielle. Cette étape devra également s'attarder sur le code VHDL du routeur qui n'est pas optimisé pour une structure FPGA mais pour une technologie ASIC entraînant un coût en surface important (1000 slices par routeur) qui doit être réduit pour offrir plus de souplesse lors des émulations.

Pour finir, les différents travaux de recherche menés sur les réseaux sur puce dans les domaines de la recherche industrielle et universitaire sont souvent menés sur un seul et même réseau. Or, l'intérêt de ces travaux serait de comparer les performances entre chaque réseau sur puce disponible pour une même application. Par contre, nous l'avons vu dans le chapitre 1, ces réseaux peuvent avoir des structures, des topologies et des qualités de services différentes rendant leur comparaison difficile. Par conséquent, nous souhaitons approfondir ce point en proposant un modèle de description simplifié de routeur permettant de décrire les différents réseaux existants et de les intégrer à notre outil de conception. Ceci permet donc de choisir une architecture adaptée de réseau en fonction de l'application utilisée. Une architecture réseau en qualité de service BE sera peut-être plus adaptée à un GT dans le cas de tâches traitées par des processeurs interconnectés sur un réseau. Il a été développé dans le cadre d'une collaboration entre le LESTER et l'IETR un NoC μspider. Nous souhaitons comparer ces deux réseaux qui ont une structure et une topologie similaire mais une qualité de service différente.

On peut ainsi lister l'ensemble des travaux des perspectives à moyen terme et court terme par ordre de priorité pour les prochains mois :

1. intégrer des traceurs pour les FIFO de chaque NI afin de pouvoir adapter leur profondeur à chaque bloc de traitement et réduire le coût matériel du design.
2. étudier le code VHDL du routeur afin de réduire son coût en surface et augmenter sa fréquence maximale de fonctionnement
3. intégrer la génération automatique du code VHDL du SoC simulé en SystemC par une description en langage XML
4. optimiser la phase AAA afin qu'elle puisse gérer les architectures matérielles d'implantation multi-composants
5. modéliser le routeur et sa NI associée en consommation électrique
6. proposer un format de modélisation de NoC afin de pouvoir évaluer deux NoC aux caractéristiques différentes (n'ayant pas la même QoS par exemple) pour une même application
7. mettre en place une interface graphique permettant d'assouplir notamment le mode semi-automatique dont les paramètres sont actuellement renseignés dans un classeur Excel

Acronymes & Abréviations

La signification d'une abréviation ou d'un sigle n'est souvent indiquée que lors de sa première apparition dans le texte. Il existe dans la plupart des cas une abréviation en français et une abréviation en anglais. Dans les deux cas, les deux abréviations sont données une première fois et nous employons ensuite l'abréviation la plus usuelle, celle-ci étant le plus souvent celle en anglais.

4G	*4^e Génération de téléphonie mobile*
AAA	*Adéquation Algorithme Architecture*
ad-hoc	*qui va vers ce vers quoi il doit aller, c'est-à-dire formé dans un but précis*
AMBA-AHB	*Advanced High-performance Bus*
AMBA-APB	*Advanced Peripheral Bus*
AMRC	*Accès Multiple par Répartition en Code*
APG	*APplication Graph*
ARG	*ARchitecture Graph*
ASIC	*Application Specific Integrated Circuit*
BE	*Best Effort*
CAN	*Convertisseur Analogique Numérique*
CBS	*Codage Binaire à Symbole*
CDMA	*Code Division Multiple Access*
CFM	*Configuration Manager*
CFO	*Carrier Frequency Offset*
CNA	*Convertisseur Numérique Analogique*
CPU	*Central Processing Unit*
DFS	*Dynamic Frequency Scaling*
DMA	*Direct Memory Access*
DVS	*Dynamic Voltage Scaling*
EDK	*Embedded Development Kit*
Fat-Tree	*Arbre Elargi*
FAUST	*Flexible Architecture of Unified System for Telecommunication*
FFT	*Fast Fourrier Transform : Transformée de Fourrier rapide*
FIFO	*First In First Out*
FLIT	*FLow control unIT*
FPGA	*Field Programmable Gate Array*
FSL	*Fast Simplex Link*
FSM	*Finite State Machine*

GALS	*Globalement Asynchrone, Localement Synchrone*
GPRS	*General Packet Radio Service*
GT	*Guaranteed Traffic*
ICC	*Input Communication Controller*
IEEE	*Institute of Electrical and Electronics Engineers*
IG	*Intervalle de Garde*
IP	*Intellectuel Property*
IP	*Input Port : port d'entrée*
ITM	*Interrupt Manager*
JTAG	*Joint Test Action Group*
LLR	*Log Likelihood Ratios*
MAC	*Medium Access Control*
MC-CDMA	*Multiple Carrier Code Division Multiple Access*
MIMO	*Multiple-Input Multiple-Output*
MMU	*Memory Management Unit*
NI	*Network Interface*
NoC	*Network on Chip*
NTTP	*NoC Transaction and Transport Protocol*
OCC	*Output Communication Controller*
OCP	*Open Core Protocol*
OCP-IP	*Open Core Protocol International Partnership*
OFDM	*Orthogonal Frequency Division Multiplexing : Multiplexage par répartition en fréquences orthogonales*
OMI	*Open Microprocessor Initiative*
OP	*Output Port :port de sortie*
OPB	*On-chip Peripheral Bus*
OS	*Operating System*
PC	*Personal Computer*
PCB	*Printed Circuit Board*
PCI	*Peripheral Component Interconnect*
PI-BUS	*Peripheral Interconnect Bus*
PSK	*Phase Shift Keying*
QAM	*Quadrature Amplitude Modulation*
QoS	*Quality of Service*
QPSK	*Quadrature Phase Shift Keying*
RAID	*Redundant Array of Independent Disks*
RAM	*Random Access Memory*
RISC	*Reduced Instruction-Set Computer*
RTC	*Réseau Téléphonique Commuté*
RTL	*Register Transfer Level*
RWD	*Read\Write decoder*
SCSI	*Small Computer System Interface*
SF	*Store-and-Forward*
SFO	*Sampling Frequency Offset*
Siso	*Single-Input Single-Output*
SoC	*System on Chip*

SS-MC-MA	*Spread Spectrum Multiple Carrier Multiple Access*
TDMA	*Time Division Multiple Access*
TLM	*Transaction Level Model*
ULB	*Ultra Large Bande*
UMARS	*Unified MApping, Routing and Slot allocation*
UMTS	*Universal Mobile Telecommunication System*
UT	*Unité de Traitement*
VCI	*Virtual Component Interface*
VCT	*Virtual Cut-Trough*
VHDL	*Very High Description Language*
W-CDMA	*Wideband-Code Division Multiple Access*
WiFi	*Wireless Fidelity ou IEEE802.11b Direct Sequence*
WLAN	*Wireless Local Area Network*

Table des figures

Liste des tableaux

Bibliographie

[1] Julien Delorme et Dominique Houzet, « Fast prototyping PCI platform for real time SoC emulation with SCSI capabilities ». In *ISPASS*, March 2005.

[2] Julien Delorme et Dominique Houzet, « Latency-constrained embedded benchmark for evaluation of cores mapping onto NoC architectures ». In *RecoSoC*, juin 2005.

[3] J. Delorme et D. Houzet, « A complete 4G radiocommunication application mapping onto a 2D mesh NoC architecture ». In *NEWCAS2006*, pages 93–96, june 2006.

[4] J. Delorme et D. Houzet, « Une technique de mapping pour les réseaux sur puce de structure 2D ». In *JNRDM2006 Journée Nationnale du Reséau Doctorale en Microélectronique*, mai 2006.

[5] Julien Delorme et Dominique Houzet, « An automatic design flow for mapping application onto a 2D mesh NoC architecture ». In *NoCsymphosium*, mai 2006.

[6] Ludovic BARRANDON, *Synthèse architecturale analogique / numérique appliquée aux systèmes sur puce dans un contexte radio logicielle.* Electronique, Universite de RENNES 1 / IETR, 08 Décembre 2005.

[7] J. Rabaey, « Busses and networking ». In *Computer Science 252*, spring 2000.

[8] ARM, « http ://www.arm.com/products/solutions/AMBAAPB.html ». In *AMBA-APB*, 2006.

[9] Sonics, « http ://www.sonicsinc.com/ ». In *SonicsMX*, 2006.

[10] OMI, « OMI PI-Bus Specification ». In *OMI 324, PI-Bus Rev. 0.3d*, 1994.

[11] Philip de Nier, « Property Checking of PI-Bus Modules ». In *ProRISC*, (P.O. Box 513, 5600 MB), 1999.

[12] AVALON, « http ://www.altera.com/products/software/products/sopc/avalon ». In *ALTERA*, 2006.

[13] IBM, « http ://www-03.ibm.com/chips/products/coreconnect/ ». 2006.

[14] William J. Dally et Brian Towles, « Route packets, not wires : on-chip inteconnectoin networks ». In *DAC '01 : Proceedings of the 38th conference on Design automation*, (New York, NY, USA), pages 684–689, 2001.

[15] IBM CoreConnect, « http ://www.chips.ibm.com/products/coreconnect/ ». In *IBM*, 2006.

[16] G. de Micheli et L. Benini, « Networks on Chip : A New Paradigm for Systems on Chip Design ». In *DATE '02 : Proceedings of the conference on Design, automation and test in Europe*, (Washington, DC, USA), page 418, IEEE Computer Society, 2002.

[17] A. Grbic, S. Brown, S. Caranci, R. Grindley, M. Gusat, G. Lemieux, K. Loveless, N. Manjikian, S. Srbljic, M. Stumm, Z. Vranesic et Z. Zilic, « Design and implementation of the NUMAchine multiprocessor ». In *DAC '98 : Proceedings of the 35th annual conference on Design automation*, (New York, NY, USA), pages 66–69, ACM Press, 1998.

[18] S.K. Tewksbury, M. Uppuluri et L.A. Hornak, « Interconnections/Micro-Networks for Integrated Microelectronics ». In *GLOBECOM'92*, (Morgantown, WV 26506), 1992.

[19] Ahmed Hemani, Axel Jantsch, Shashi Kumar, Adam Postula, Johnny Oberg, Mikael Millberg et Dan Lindqvist, « Network on chip : An architecture for billion transistor era. ». In *In Proceeding of the IEEE NorChip Conference*, November 2000.

[20] Théodore Marescaux, Andrei Bartic, Dideriek Verkest, Serge Vernalde et Rudy Lauwereins, *Interconnection Networks Enable Fine-Grain Dynamic Multi-tasking on FPGAs*, vol. Volume 2438/2002 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 2-4 septembre 2002.

[21] IST 4MORE project, « http ://4more.av.it.pt/ ». In *European project*, 2004-2006.

[22] H. Charlery, E. Encrenaz, A. Greiner, A. Andriahantenaina et al., « SPIN, un micro réseau d'interconnexion à commutation de paquets respectant la norme VCI. Concepts généraux et validation ». In *SympAAA 2003*, (La Colle sur Loup, France), pages 337–344, October 2003.

[23] Pierre GUERRIER, *Un réseau d'interconnexion pour systèmes intégrés*. Thèse de Doctorat, UNIVERSITÉ PARIS VI - PIERRE ET MARIE CURIE U.F.R. D'INFORMATIQUE, 10 mai 2000.

[24] P. Guerrier et A. Greiner, « A Generic Architecture for on-chip Packet Switched Interconnections ». In *Design Automation and Test in Europe (DATE)*, (Paris, France), pages 250–256, 2000.

[25] P. Guerrier et A. Greiner, « A Scalable Architecure for System-On-Chip Interconnections ». In *Proceedings of the Sophia-Antipolis MicroElectronics Conference (SAME)*, (Sophia Antipolis FRANCE), pages 90–93, October 1999.

[26] V. Bangalore et N. Nayak Sujir, « Performance improvement of on-chip communication architecture using Multicast ». In *On-chip communication Networks Research paper*, Spring 2002.

[27] Krishnan Srinivasan, Karam S. Chatha et Goran Konjevod, « Linear Programming based Techniques for Synthesis of Network-on-Chip Architectures ». In *ICCD '04 : Proceedings of the IEEE International Conference on Computer Design (ICCD'04)*, (Washington, DC, USA), pages 422–429, IEEE Computer Society, 2004.

[28] Krishnan Srinivasan et Karam S. Chatha, « A technique for low energy mapping and routing in network-on-chip architectures ». In *ISLPED '05 : Proceedings of the 2005 international symposium on Low power electronics and design*, (New York, NY, USA), pages 387–392, ACM Press, 2005.

[29] Christopher J. Glass et Lionel M. Ni, « The turn model for adaptive routing ». vol. 41, (New York, NY, USA), pages 874–902, ACM Press, 1994.

[30] L. M. Ni et P. K. McKinley, « A Survey of Wormhole Routing Techniques in Direct Networks ». *Computer*, vol. 26, pages 62–76, 1993.

[31] Andreas Hansson, Kees Goossens et Andrei Radulescu, « A unified approach to constrained mapping and routing on network-on-chip architectures ». In *CODES+ISSS '05 : Proceedings of the 3rd IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis*, (New York, NY, USA), pages 75–80, ACM Press, 2005.

[32] E. Rijpkema, K. G. W. Goossens, A. Radulescu, J. Dielissen, J. van Meerbergen, P. Wielage et E. Waterlander, « Trade Offs in the Design of a Router with Both Guaranteed and Best-Effort Services for Networks on Chip ». In *DATE '03 : Proceedings of the conference on Design, Automation and Test in Europe*, (Washington, DC, USA), page 10350, IEEE Computer Society, 2003.

[33] Jingcao Hu et Radu Marculescu, « Exploiting the Routing Flexibility for Energy/Performance Aware Mapping of Regular NoC Architectures ». In *DATE '03 : Proceedings of the conference on Design, Automation and Test in Europe*, (Washington, DC, USA), page 10688, IEEE Computer Society, 2003.

[34] Jingcao Hu et Radu Marculescu, « Energy-Aware Mapping for tile-based NoC Architectures under performance constraints ». In *Design Automation Conference, 2003. Proceedings of the ASP-DAC 2003. Asia and South Pacific*, (Washington, DC, USA), pages 233 – 239, IEEE Computer Society, 2003.

[35] Jingcao Hu et Radu Marculescu, « Energy-Aware Communication and Task Scheduling for Network-on-Chip Architectures under Real-Time Constraints ». In *DATE '04 : Proceedings of the conference on Design, automation and test in Europe*, (Washington, DC, USA), page 10234, IEEE Computer Society, 2004.

[36] Kees Goossens, John Dielissen, Om Prakash Gangwal, Santiago Gonzalez Pestana, Andrei Radulescu et Edwin Rijpkema, « A Design Flow for Application-Specific Networks on Chip with Guaranteed Performance to Accelerate SOC Design and Verification ». In *DATE '05 : Proceedings of the conference on Design, Automation and Test in Europe*, (Washington, DC, USA), pages 1182–1187, IEEE Computer Society, 2005.

[37] Srinivasan Murali et Giovanni De Micheli, « Bandwidth-Constrained Mapping of Cores onto NoC Architectures ». In *DATE '04 : Proceedings of the conference on Design, automation and test in Europe*, (Washington, DC, USA), page 20896, IEEE Computer Society, 2004.

[38] Srinivasan Murali et Giovanni De Micheli, « SUNMAP : A Tool for Automatic Topology Selection and Generation for NoCs ». (New York, NY, USA), pages 914–919, ACM Press, 2004.

[39] N Koziris et al., « An efficient mappping algorithm for the physical mapping of clustered task graphs onto multiprocessor architectures ». In *EuroPDP : Proceedings of the 8th conference EuroPDP*, pages 406–413, 2000.

[40] Antoine Jalabert, Srinivasan Murali, Luca Benini et Giovanni De Micheli, « XpipesCompiler : A Tool for Instantiating Application Specific Networks on Chip ». In *DATE '04 : Proceedings of the conference on Design, automation and test in Europe*, (Washington, DC, USA), page 20884, IEEE Computer Society, 2004.

[41] Matteo Dall'Osso, Gianluca Biccari, Luca Giovannini, Davide Bertozzi et Luca Benini, « xpipes : a Latency Insensitive Parameterized Network-on-chip Architecture For Multi-Processor SoCs ». In *ICCD '03 : Proceedings of the 21st International*

Conference on Computer Design, (Washington, DC, USA), page 536, IEEE Computer Society, 2003.

[42] Srinivasan Murali, Martijn Coenen, Andrei Radulescu, Kees Goossens et Giovanni De Micheli, « Mapping and Configuration Methods for Multi-Use-Case Networks on Chips ». In *ASPDAC '06 : Proceedings of the conference on ASPDAC*, (Washington, DC, USA), IEEE Computer Society, 2006.

[43] CEA LETI, « http ://www-leti.cea.fr/fr/index-fr.htm ». 2006.

[44] OCB Design Working Group, « VSI Alliance Virtual Component Interface Standard ». In *Virtual Sockets Interface Alliance*, 2000.

[45] « System VC Interface Strawman Version 0.3.2, Strawman. ».

[46] Hervé Charlery et Alain Greiner, « Systèmes intégrés : un micro-réseau d'interconnexions à commutation de paquets respectant la norme VCI ». In *Troisième Colloque CAO de circuits et systèmes intégrés*, mai 2002.

[47] Adrijean Adriahantenaina, Herve Charlery, Alain Greiner, Laurent Mortiez et Cesar Albenes Zeferino, « SPIN : A Scalable, Packet Switched, On-Chip Micro-Network ». In *DATE '03 : Proceedings of the conference on Design, Automation and Test in Europe*, (Washington, DC, USA), page 20070, IEEE Computer Society, 2003.

[48] R. Lemaire, F. Clermidy, Y. Durand, D. Lattard et A. A. Jerraya, « Performance Evaluation of a NoC-Based Design for MC-CDMA Telecommunications Using NS-2 ». In *RSP '05 : Proceedings of the 16th IEEE International Workshop on Rapid System Prototyping (RSP'05)*, (Washington, DC, USA), pages 24–30, IEEE Computer Society, 2005.

[49] Lemaire R, Lattard D et Jerraya A, « Evaluation des performances de transferts de données sur un NoC régulé par un mécanisme de controle de flux ». JNRDM05, mai 2005.

[50] E. Beigne, F. Clermidy, P. Vivet, A. Clouard et M. Renaudin, « An Asynchronous NOC Architecture Providing Low Latency Service and Its Multi-Level Design Framework ». In *ASYNC '05 : Proceedings of the 11th IEEE International Symposium on Asynchronous Circuits and Systems*, (Washington, DC, USA), pages 54–63, IEEE Computer Society, 2005.

[51] E. Beigne et P. Vivet, « Design of On-chip and Off-chip Interfaces for a GALS NoC Architecture ». In *ASYNC '06 : Proceedings of the 12th IEEE International Symposium on Asynchronous Circuits and Systems*, (Washington, DC, USA), page 172, IEEE Computer Society, 2006.

[52] Dobkin (Reuven) Rostislav, Victoria Vishnyakov, Eyal Friedman et Ran Ginosar, « An Asynchronous Router for Multiple Service Levels Networks on Chip ». pages 44–53, 2005.

[53] Evgeny Bolotin, Israel Cidon, Ran Ginosar et Avinoam Kolodny, « QNoC : QoS architecture and design process for network on chip ». *J. Syst. Archit.*, vol. 50, n°2-3, pages 105–128, 2004.

[54] E Bolotin, A Morgenshtein, I Cidon, R Ginosar et A Kolodny, « Automatic hardware-efficient SoC integration by QoS network on chip ». In *Electronics, Circuits and Systems, 2004. ICECS 2004*, no. 0-7803-8715-5, (Dept. of Electr. Eng., Technion-Israel Inst. of Technol., Haifa, Israel), 13-15 Dec 2004.

[55] Evgeny Bolotin, Israel Cidon, Ran Ginosar et Avinoam Kolodny, « Cost considerations in network on chip ». *Integr. VLSI J.*, vol. 38, n°1, pages 19–42, 2004.

[56] Faraydon Karim, Anh Nguyen et Sujit Dey, « An Interconnect Architecture for Networking Systems on Chips ». *IEEE Micro*, vol. 22, n°5, pages 36–45, 2002.

[57] Luciano Bononi et Nicola Concer, « Simulation and analysis of network on chip architectures : ring, spidergon and 2D mesh ». In *DATE '06 : Proceedings of the conference on Design, automation and test in Europe*, (3001 Leuven, Belgium, Belgium), pages 154–159, European Design and Automation Association, 2006.

[58] « ARTERIS ANNOUNCES STMICROELECTRONICS USE OF NoC FOR NEXT GENERATION WIRELESS INFRASTRUCTURE PLATFORM ». 15 mars 2006.

[59] « A comparison of NoC and busses ». 2005.

[60] Srinivasan Murali, Luca Benini et Giovanni De Micheli, « Mapping and physical planning of networks-on-chip architectures with quality-of-service guarantees ». In *ASP-DAC '05 : Proceedings of the 2005 conference on Asia South Pacific design automation*, (New York, NY, USA), pages 27–32, ACM Press, 2005.

[61] Kees Goossens, John Dielissen, Jef van Meerbergen, Peter Poplavko, Andrei R&#259 ;dulescu, Edwin Rijpkema, Erwin Waterlander et Paul Wielage, « Guaranteeing the quality of services in networks on chip ». pages 61–82, 2003.

[62] John Dielissen, Andrei Radulescu, Kees Goossens et Edwin Rijpkema, « Concepts and Implementation of the Philips Network-on-Chip ». IPSOC, 2003.

[63] Andrei Radulescu, John Dielissen, Kees Goossens, Edwin Rijpkema et Paul Wielage, « An Efficient On-Chip Network Interface Offering Guaranteed Services, Shared-Memory Abstraction, and Flexible Network Configuration ». In *DATE '04 : Proceedings of the conference on Design, automation and test in Europe*, (Washington, DC, USA), page 20878, IEEE Computer Society, 2004.

[64] ARM, « AMBA AXI Protocol Specification ». Juin 2003.

[65] Philips Semiconductors, « Device Transaction Level (DTL) Protocol Specification. Version 2.2 ». Juillet 2002.

[66] Andrei Radulescu, John Dielissen, Kees Goossens, Edwin Rijpkema et Paul Wielage, « An Efficient On-Chip Network Interface Offering Guaranteed Services, Shared-Memory Abstraction, and Flexible Network Configuration ». In *DATE '04 : Proceedings of the conference on Design, automation and test in Europe*, (Washington, DC, USA), page 20878, IEEE Computer Society, 2004.

[67] Srinivasan Murali, Martijn Coenen, Andrei Radulescu, Kees Goossens et Giovanni De Micheli, « A methodology for mapping multiple use-cases onto networks on chips ». In *DATE '06 : Proceedings of the conference on Design, automation and test in Europe*, (3001 Leuven, Belgium, Belgium), pages 118–123, European Design and Automation Association, 2006.

[68] S. Evain, J. P. Diguet et D. Houzet, « μSpider : A CAD Tool for Efficient NoC Design ». In *IEEE NORCHIP 2004*, 8-9 November 2004.

[69] S. Evain, J. P. Diguet et D. Houzet, « A Generic CAD Tool for Efficient NoC Design ». In *IEEE ISPACS 2004, International Symposium on Intelligent Signal*, (Corea), November 2004.

[70] Samuel Evain, *Conception et modélisation d'un système de contrôle d'applications de télécommunication avec une architecture de réseau sur puce (NoC).* Thèse de Doctorat, Institut National des Sciences Appliquées, 11 octobre 2006.

[71] IETR UMR CNRS 6164, « Rennes ». In *www.ietr.org.*

[72] LESTER, « http ://web.univ-ubs.fr/lester/www-lester/Index.php ». In *Lorient.*

[73] Projet Européen MATRICE, « http ://www.ist-matrice.org/ ».

[74] « SystemC ». In *http ://www.systemc.org/.*

[75] C. Sorel et Y. Lavarenne, « From Algorithm and Architecture Specifications to Automatic Generation of Distributed Real-Time Executives : a Seamless Flow of Graphs Transformations ». In *Formal Methods and Models for Codesign Conference,France*, June 2003.

[76] Stéphane Nobilet, *Etudes et optimisation MC-CDMA pour les futures générations de systèmes de communications hertziennes.* Thèse de Doctorat, 03 octobre 2003.

[77] Siavash M. Alamouti, « A Simple Transmit Diversity Technique for Wireless Communications ». *IEEE Journal on Selected Areas in Communications*, vol. 16, pages 1451–1458, Octobre 1998.

[78] XILINX, « Site de la société Xilinx ». In *http ://www.xilinx.com*, 2006.

[79] Nallatech, « BEN ADDA ». In *http ://www.nallatech.com/mediaLibrary/images-/english/4429.pdf*, 2006.

[80] XILINX, « VIRTEX4 ». In *http ://www.xilinx.com*, 2006.

[81] Romain LEMAIRE, *Conception et modélisation d'un système de contrôle d'applications de télécommunication avec une architecture de réseau sur puce (NoC).* Thèse de Doctorat, Institut National Polytechnique de Grenoble, 11 octobre 2006.

[82] Anoop Iyer et Diana Marculescu, « Power and performance evaluation of globally asynchronous locally synchronous processors ». In *ISCA '02 : Proceedings of the 29th annual international symposium on Computer architecture*, (Washington, DC, USA), pages 158–168, IEEE Computer Society, 2002.

[83] Cesar Marcon, Ney Calazans, Fernando Moraes, Altamiro Susin, Igor Reis et Fabiano Hessel, « Exploring NoC Mapping Strategies : An Energy and Timing Aware Technique ». In *DATE '05 : Proceedings of the conference on Design, Automation and Test in Europe*, (Washington, DC, USA), pages 502–507, IEEE Computer Society, 2005.

[84] Flot de conception d'arteris, « http ://www.arteris.com/products.html ».

[85] Davide Bertozzi et Antoine Jalabert, « NoC Synthesis Flow for Customized Domain Specific Multiprocessor Systems-on-Chip ». *IEEE Trans. Parallel Distrib. Syst.*, vol. 16, n°2, pages 113–129, 2005. Student Member-Srinivasan Murali and Student Member-Rutuparna Tamhankar and Student Member-Stergios Stergiou and Member-Luca Benini and Fellow-Giovanni De Micheli.

[86] Rickard Holsmark et Shashi Kumar, « An option for accessing regions in noc ». In *DATE*, School of engineering, 2006.

[87] Isask'har Walter, Asrael Cidon, Ran Ginosar et Avinoam Kolodny, « Curing Hotspots in Wormhole NoCs ». In *DATE*, ElectricalEngineering Department, 2006.

[88] Julien Delorme, « Experience plan part 1 ». Rapport, *IETR*, 23 Mars 2005.

[89] Julien Delorme, « Experience plan part 2 ». Rapport, *IETR*, 19 septembre 2005.

[90] Julien Delorme, « Experience plan part 3 ». Rapport, *IETR*, 23 février 2006.

[91] F. Deslauriers, M. Langevin, G. Bois, Y.Savaria et P.Paulin, « RoC : A scalable Network on Chip based on the token ring concept ». In *NEWCAS2006*, pages 157–160, june 2006.

[92] R. Ben Mouhoub et O. Hammami, « NoC Monitoring feedback for parallel programmers ». In *NEWCAS2006*, pages 141–144, june 2006.

Le résumé

Les densités d'intégration actuelles des circuits intégrés permettent de disposer de SoC (systèmes sur puce) de plus en plus complexes, intégrant de plus en plus de standards. Par conséquent, le problème des interconnexions entre tous les blocs IP (Intellectual Property) constituant le SoC devient un point critique que les structures de communications actuelles ne parviennent plus à solutionner.

Ces problèmes sont notamment liés aux besoins de plus en plus forts en mobilité et en débit dans les architectures de communication actuelles et futures. Ainsi, les solutions à base de NoC (Network on Chip) offrent de bonnes perspectives en terme de bande passante et de flexibilité pour pallier notamment aux limites actuelles des topologies bus. Les travaux de thèse présentés ici portent sur la méthodologie de modélisation et d'exploration d'architectures de réseaux sur puce appliquée aux télécommunications.

Le contexte radio-télécommunications étudié est celui proposé dans le cadre du projet Européen 4MORE pour lequel nous avons contribué. Une des contraintes de ce projet était d'intégrer dans un SoC la technique MC-CDMA (Multiple Carrier Code Division Multiple Access) combinant la technique MIMO en utilisant un média de communication innovant.

Ainsi, nous avons contribué à cette intégration en proposant une méthodologie de conception permettant d'aider le concepteur dans le choix des différents paramètres caractérisant le NoC pour satisfaire les contraintes temps réel de l'application spécifiées dans le cahier des charges.

Ces travaux de thèse ont porté sur la modélisation et l'interconnexion des composants IP constituant la chaîne algorithmique du projet 4MORE afin de les intégrer dans un modèle SystemC du NoC. Par ailleurs, les choix de dimensionnement du réseau et des contraintes de placement des blocs IP sur celui-ci ont un impact important sur les performances globales de l'application. Nous avons mis en place un outil AAA (Adéquation Algorithme Architecture) permettant de réaliser l'adéquation des contraintes de l'application sur l'architecture en minimisant les chemins de communication tout en veillant à ne pas violer les bandes passantes théoriques des liens de communication entre routeurs.

Le flot de conception mis en oeuvre permet au concepteur de générer le modèle SystemC du NoC et permettra à cours terme de générer le code VHDL associé du modèle SystemC simulé afin d'accélérer les phases de simulation et de donner la possibilité de valider logiciellement et matériellement (cible FPGA) l'architecture avec son application.

Mots clés : Système sur puce, SoC, conception de circuit, synthèse d'architecture, communication, réseau sur puce, NoC, routeur, interface réseau, outil d'aide à la conception, GALS.