μSpider Environnement de Conception de Réseau sur Puce

Samuel Evain

N° d'ordre : D06-16

Thèse

présentée devant
l'INSTITUT NATIONALE DES SCIENCES APPLIQUÉES DE RENNES

pour obtenir le titre de

Docteur

spécialité : *Électronique*

μSpider
Environnement de Conception de Réseau sur Puce

par
Samuel Evain

Soutenue le 24 Novembre 2006 devant la commission d'Examen

Composition du jury

Rapporteurs	
M. Ahmed Amine JERRAYA	Directeur de recherche CNRS, TIMA, Grenoble
M. Tanguy RISSET	Professeur, LIP, INSA, Lyon
Examinateurs	
M. Fréderic ROBERT	Professeur, BEAMS, Université Libre de Bruxelles
M. Philippe MARTIN	Docteur, ARTERIS, Guyancourt
Codirecteurs	
M. Dominique HOUZET	Professeur, LIS, ENSERG, Grenoble
M. Jean-Philippe DIGUET	Chargé de recherche CNRS, LESTER, Université de Bretagne Sud, Lorient

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

Résumé

Ce travail de thèse porte sur la conception de l'interconnexion entre les nombreux composants IP (Intellectual Property) d'un système électronique sur puce (SoC pour *System on Chip*).

Notre étude repose sur une solution émergente qui est celle des réseaux sur puce (NoC pour *Network-on-chip*), celle-ci est inspirée des réseaux de communication entre ordinateurs.

Un NoC offre de nombreuses possibilités et un large espace de conception. La maîtrise des choix des paramètres d'un NoC vis à vis des contraintes d'une application n'est pas triviale et nécessite de la méthode.

Cette thèse propose un flot de conception afin de déterminer ces paramètres automatiquement. Le problème de l'horloge dans les circuits de grande taille, ainsi que l'aspect sécurité sont également traités.

Ce travail a conduit au développement de l'outil μSpider, qui est un environnement de conception composé d'outils de décisions et d'un générateur de code (VHDL synthétisable).

Ce travail a été validé avec des applications dans les domaines du traitement du signal, de l'image et des télécommunications.

Mots clés : Système sur puce, conception de circuit, synthèse d'architecture, communication, réseau sur puce, NoC, routeur, interface, outil d'aide à la conception, allocation de chemins, GALS, sécurité.

Abstract

This PhD thesis deals with interconnection design between IP cores (Intellectual Property) in a System on Chip. This work is based on an emerging infrastructure solution called Network on Chip.

The design space is wide and methodologies are required to help designer to obtain ad hoc NoC matching application requirements.

This work presents a design flow to design automatically ad hoc NoC parameters.

Clock synchronicity subject in wide chip and security concerns in NoC are also addressed.

We have developed a design tool called μSpider. This tool is able to make choices and to configure NoC parameters according to applications constraints and needs. Moreover, it generates the VHDL RTL code of the specified NoC.

This work has been validated on three real-life applications, a smart camera application, a telecom MC-CDMA application and a multiprocessor turbo-decoder application.

Keywords : System-On-Chip, architecture design, high-level synthesis, communication, Network-on-Chip, NoC, router, interface, computer aided design, path decision, security.

Remerciements

Je tiens tout d'abord à remercier Monsieur Jean-Philippe Diguet, Chargé de Recherche CNRS au laboratoire LESTER, pour avoir accepté d'être le codirecteur de cette thèse. Je tiens à lui exprimer ma reconnaissance pour sa disponibilité, ses conseils et sa gentillesse.

Je remercie Monsieur Dominique Houzet, Maître de Conférence à l'IETR durant ma thèse et maintenant Professeur au LIS de Grenoble, pour m'avoir proposé cette thèse, dirigé et soutenu durant ces quatre années.

Je remercie Monsieur Ahmed Amine Jerraya et Monsieur Tanguy Risset pour m'avoir fait l'honneur d'être les rapporteurs de cette thèse. Je remercie également Monsieur Fréderic Robert et Monsieur Philippe Martin pour avoir accepté d'examiner ces travaux de recherche.

Je remercie les Directeurs successifs du laboratoire LESTER pour m'avoir accueilli, Monsieur Éric Martin, aujourd'hui Président de l'Université de Bretagne Sud, et Monsieur Emmanuel Boutillon, Professeur à l'Université de Bretagne Sud et Directeur du LESTER.

Merci à tous mes collègues des Laboratoires LESTER et IETR avec qui j'ai eu le plaisir de travailler.

Je remercie tout particulièrement Samuel Rouxel et Bertrand Le Gal pour leur amitié et leur soutien précieux.

Je tiens également à remercier les membres du projet R-PUCE avec lesquelles il a été très enrichissant de travailler.

Je remercie le CEA-LETI pour les données concernant l'application MC-CDMA du projet 4MORE.

Enfin, je remercie ma famille et mes amis pour leur soutien et leurs encouragements.

Avant propos

Ces travaux ont débuté dans le cadre d'une Equipe Projet Multi-Laboratoire (EPML) appelée « Méthode de conception des SoC pour la radio logicielle » (Sep 02 - Sep 05). Cette équipe projet regroupait des membres du laboratoire IETR [(1)] de l'INSA [(2)] de Rennes, et du laboratoire LESTER [(3)] de l'UBS. [(4)]. Ce projet portait sur la définition des moyens et méthodes à mettre en œuvre pour obtenir un flot de conception de SoC adapté aux applications de radio logicielle. Il traitait différents aspects dont l'étude des réseaux intégrés. Ces travaux traitent de cet aspect précis.

Pour apporter plus de cohésion à cette équipe Multi-Laboratoire, cette thèse, a été menée au sein de l'équipe « Codesign » du LESTER tout en étant dirigée conjointement par l'équipe « Communications Propagation Radar » de l'IETR. Au début de cette thèse, le domaine des NoCs constituait une nouvelle thématique pour ces deux laboratoires. Le laboratoire IETR disposait d'une expérience en prototypage et en modélisation dans le domaine des applications de télécommunications. Le laboratoire LESTER disposait d'une expérience en reconfiguration, partitionnement et synthèse architecturale. Dans ce cadre, les contraintes de communication de l'application 4G MIMO MC-CDMA du projet 4MORE [1] nous ont été fournies pour nos expérimentations.

Durant la dernière année de thèse, ces travaux se sont intégrés dans le projet R-PUCE (Réseau embarqué sur puce) composé des départements d'Électronique et RSM [(5)] de l'ENST [(6)] Bretagne, du département RST[(7)] de l'INT [(8)] d'Évry, et du LESTER de l'UBS. Ce projet porte lui aussi sur le problème de l'interconnexion dans les futures puces électroniques. Il vise une application du domaine des communications numériques, le turbo-décodage. L'objectif est de tirer parti de la pluridisciplinarité des équipes partenaires de ce projet pour adapter les modèles, les techniques et les outils du domaine des réseaux d'ordinateurs, au contexte d'intégration sur silicium. Nous avons abouti à des solutions qui sont présentées dans le chapitre expérimentations.

[(1)] Institut d'Electronique et de Télécommunications de Rennes
[(2)] Institut National des Sciences Appliquées
[(3)] Laboratoire d'Électronique des Systèmes TEmps Réel
[(4)] Université de Bretagne-Sud
[(5)] Réseaux, Sécurité et Multimédia
[(6)] École Nationale Supérieure des Télécommunications
[(7)] Réseaux et Services des Télécommunications
[(8)] Institut National des Télécommunications

Table des matières

Chapitre 1

Introduction

1.1 Contexte

Nous assistons à la convergence de plusieurs produits vers un seul appareil multifonctions, avec des appareils tels que les *set-top boxes* (TV, décodeurs, DVD, enregistreur numérique, accès réseaux) à domicile [2], les PC multimédias, les consoles de jeux vidéo, les téléphones portables, etc.

Ceux sont des systèmes dont la pluralité des fonctionnalités nécessite l'utilisation d'architectures électroniques offrant de hautes performances et une grande flexibilité pour leurs permettre de s'adapter aux besoins de l'utilisateur, à l'environnement et aux évolutions des normes futures. Cette multiplicité des fonctions nécessite d'importants moyens de communication au sein du système.

Cette augmentation des fonctionnalités des appareils est rendue possible grâce aux progrès continus des méthodes de conception et des capacités d'intégration dans les puces électroniques. Le concepteur dispose de composants matériels appelés IP(s) (*Intellectual Property*) qui, comme des cellules de bibliothèque peuvent être utilisés comme éléments de base pour concevoir de nouveaux circuits. Ceci permet la réutilisation de ce qui a déjà été conçu et ainsi d'accélérer le développement. De plus, la densité d'intégration des transistors augmentant de façon continue, il est aujourd'hui possible de construire des circuits contenant plusieurs centaines de millions de transistors. Il est ainsi concevable de développer un système complet sur une seule puce de silicium, nous parlons alors de système mono-puce (SoC pour System on Chip). Un SoC intègre en général des composants IPs variés, tels que des processeurs généralistes et spécialisés, de la mémoire, des interfaces pour des périphériques, des moyens de communication, des circuits dédiés/spécialisés pour accélérer le traitement de certaines fonctions (codec video, moteur graphique, cryptographie). L'évolution montre que de nouveaux types de composants peuvent également être intégrés, tels que des dispositifs mécaniques (MeMS[(1)]), opto-électroniques, chimiques ou biologiques ou des circuits radio (antennes) [3].

Cependant, la conception de systèmes sur puces entraîne un certain nombre de défis à relever pour concevoir les circuits de demain. En effet, les méthodes de conception et les outils dont disposent les concepteurs évoluent moins vite que la capacité d'intégration. Les concepteurs ne sont alors plus en mesure de tirer réellement profit

[(1)] Micro-electro-mechanical system

des capacités d'intégration offertes par la technologie. Un SoC intégrera bientôt plusieurs dizaines, voir centaines d'IPs interconnectées [4]. La difficulté majeure lors de la conception de systèmes sur des puces de grande taille est la communication entre les différents composants du système. En 2008, il est estimé que la traversée d'un circuit prendra 16 cycles d'horloge [5]. Le délai dans les fils domine le délai des portes à partir de la technologie à 0,25µm pour l'aluminium, et 0,18µm pour le cuivre [6]. Les communications sont ainsi devenues dominantes du point de vue des délais, de la consommation en énergie et de la surface de silicium ; Nous avons donc besoin de rendre ces communications explicites pour qu'elles puissent être prises en charge et optimisées durant le flot de conception du SoC.

De plus, se pose le problème de la *scalabilité*. La *scalabilité* est la capacité d'un système à évoluer (augmenter ses performances) en permettant d'exploiter l'amélioration apportée par son extension au niveau matériel. Or, les solutions d'interconnexions classiques ne sont pas adaptées à l'extension des SoC.

Une solution performante et économique consiste à tirer profit d'une infrastructure de communication configurable préétablie, à savoir un réseau sur puce (NoC pour *Network on Chip*) [7, 8]. Les composants communiquent alors entre eux en échangeant des paquets au travers d'un réseau d'interconnexions, fournissant une structure de communication réutilisable. Le recours à un NoC s'impose pour les circuits de grande dimension, aussi bien du point de vue technologique (longs délais entre les parties éloignées d'une même puce), que du point de vue fonctionnel (facilité de connexion des composants IPs et donc de leur réutilisation). La maîtrise des caractéristiques du NoC est un atout majeur car la consommation et la performance d'un SoC seront de plus en plus dominées par les ressources de communication.

Les contraintes imposées aux systèmes électroniques afin de satisfaire les exigences des applications et du marché sont diverses. Celles-ci sont un faible coût en surface de silicium, une faible consommation électrique, le respect des exigences de performances de bande passante, de latence et de fiabilité. La reconfigurabilité du système impose aussi comme qualités requises la flexibilité pour manipuler des trafics variés et la facilité d'interfaçage pour permettre de potentielles reconfigurations des interfaces. À ces critères, nous pouvons encore ajouter des aspects plus architecturaux tels que la testabilité, la tolérance aux incertitudes temporelles de l'implémentation et la tolérance aux défauts de fabrication. Tous ces critères doivent guider le choix de l'architecture de communication à mettre en œuvre. De plus, il est nécessaire de disposer de méthodes et d'outils qui permettent de réduire le temps de mise sur le marché (*time to market*) des produits développés.

Le NoC peut lui même être vu comme une IP rendant des services aux IPs qui lui sont connectées. Il doit permettre d'offrir un niveau d'abstraction permettant de séparer le traitement et le transport des données. Comme tout composant IP, il doit être optimisé en performances, surface et consommation d'énergie. Seulement, le NoC offre beaucoup de paramètres (topologie, fréquence, chemins, taille des tampons de mémorisation, stratégie de mémorisation, routage, arbitrage, contrôle de flux, etc.). De plus, ces choix sont très dépendants de l'application et des services attendus. L'espace de conception étant extrêmement large, il est illusoire de vouloir explorer toutes les solutions en fonction des différents compromis possibles. Il est donc nécessaire de

disposer de méthodologies et d'outils pour assister le concepteur afin de le guider dans ces choix et l'aider à concevoir le NoC adapté à ses besoins.

Enfin, dans la logique de l'évolution des architectures, le NoC, de part son rôle central dans le système, doit permettre les reconfigurations du système, pour offrir la flexibilité et l'évolution. Il doit de plus garantir la sécurité des données et des IPs raccordées. Il faudra ainsi déterminer le compromis entre le coût de mise en oeuvre et la flexibilité désirée. Enfin, il apparaît que le NoC a un lien de plus en plus fort avec les services du système d'exploitation pour contrôler et adapter la qualité de service de l'application à celui-ci et vis et versa.

1.2 Objectifs et contributions

Le but de cette thèse était d'étudier le domaine émergeant des NoCs et de proposer un outil de CAO (conception assistée par ordinateur) intégrant le concept du NoC dans le flot de conception d'un SoC.

En 2002, il existait peu de réalisations de NoC et il n'existait pas d'outil consacré au NoC. Le NoC était un concept qui commençait tout juste à montrer son intérêt.

Ce sujet étant nouveau pour les équipes des laboratoires LESTER et IETR, nous n'avions pas de NoC à notre disposition. Nous avons décidé de concevoir un routeur générique en langage VHDL au niveau RTL. Cette réalisation a eu pour intérêt de permettre de mener des simulations précises ainsi que des évaluations concrètes des caractéristiques d'un réseau.

Le concept semblait assez simple au premier abord, mais il est rapidement apparu que nous avions besoin d'un moyen pour contrôler le trafic afin de garantir les contraintes temps-réel. Nous avons alors décidé d'intégrer les canaux virtuels dans les routeurs afin de créer des classes de trafic différents.

Avec des paramètres de plus en plus nombreux, il devenait difficile d'écrire le code VHDL générique et il est alors devenu nécessaire de créer un outil de génération du code VHDL du NoC. Nous avons alors opté pour le langage de description XML et une conception de l'outil orientée objet pour nous permettre de modifier plus facilement les paramètres du NoC (topologie et paramètres divers).

Ensuite nous avons opté pour l'utilisation de tables TDMAs (*Time Division Multiple Access*) pour permettre de garantir les trafics dans le NoC. Nous avons alors développé une interface réseau capable de contrôler ce type d'accès. Celle-ci est venue enrichir l'outil de synthèse.

Il est alors devenu nécessaire de renseigner ces tables de façon appropriée. Nous avons élaboré un flot de conception complet et des méthodes de décision, afin d'organiser et d'automatiser ces choix à partir de contraintes applicatives renseignées par le concepteur. Nous avons réalisé un outil de décision capable de déterminer la taille de la table TDMA appropriée et de sélectionner, pour chaque communication, les slots qu'elle doit utiliser dans la table de TDMA, ainsi que le chemin qu'elle doit emprunter dans le réseau, afin de garantir qu'elle ne rencontrera aucune collision avec les autres communications durant son transport dans le réseau. De plus, l'outil détermine la taille des FIFOs nécessaires dans les interfaces du réseau.

Dans le cadre du projet R-PUCE, nous avions besoins de connecter des processeurs au NoC. Nous avons alors développé un adaptateur de protocole permettant de connecter les interfaces du NoC avec le bus OPB utilisé par le processeur MicroBlaze.

Parallèlement à cela, nous nous sommes penchés sur la question de la sécurité dans les NoCs et avons trouvé une solution qui s'appuie sur les instructions de routage pour offrir un service d'authentification de l'émetteur d'un paquet dans le réseau.

Enfin, nous nous sommes confrontés au problème de la distribution de l'horloge dans les circuits de grande taille dans le contexte de l'utilisation d'un TDMA qui nécessite une notion d'horloge commune. Nous avons également proposé des solutions à ce problème.

1.3 Organisation du mémoire

Ce mémoire de thèse comporte 7 autres chapitres.

Le chapitre 2 présente un état de l'art des interconnexions et des NoCs existants ainsi que les pistes de recherches que nous nous sommes fixées au fur et à mesure de l'avancement de cette thèse.

Le chapitre 3 présente l'architecture de notre NoC, le flot de conception que nous proposons et l'outil que nous avons développé pour aider le concepteur à réaliser un NoC.

Le chapitre 4 détaille la mise en œuvre des étapes de notre flot. Il décrit comment dériver les données extraites de l'application, en données utilisables au niveau du NoC. Il présente également notre technique d'exploration et d'affectation des chemins dédiés aux communications à garantir dans le NoC.

Le chapitre 5 présente des techniques de codage des instructions de routage à des fins de sécurité, ou de flexibilité.

Le chapitre 6 présente des solutions pour faire face aux problèmes d'horloges dans les futures circuits de grande taille. Nous proposons de diviser le NoC en plusieurs sub-NoCs synchrones qui garantissent le service. Des interfaces permettent de relier ces sub-NoCs.

Le chapitre 7 présente les résultats de synthèse du NoC sur FPGA, ainsi que les résultats obtenus pour des applications des domaines du traitement du signal, de l'image et des télécommunications.

Enfin, le chapitre 8 conclue sur les travaux réalisés durant cette thèse et propose des perspectives.

Chapitre 2

État de l'art

2.1 Introduction

Le domaine des NoC est un sujet de recherche récent. L'état de l'art, s'est étoffé de façon importante durant cette thèse. Le sujet est de plus passé rapidement de la recherche universitaire à la recherche industrielle (Philips research) pour arriver à une solution industrielle avec la société Arteris qui commercialise sa solution. Ceci montre l'intérêt des industriels pour obtenir des solutions de communications à base de NoCs.

Pour présenter le NoC, dans un premier temps, nous expliquons en quoi les moyens de communication classiques ne répondent plus aux contraintes futures des systèmes sur puce et ce que le réseau sur puce apporte comme solution. Dans un second temps, nous expliquons un certain nombre de notions fondamentales relatives aux NoCs. Puis, nous dressons un historique de l'évolution de la recherche dans le domaine des NoCs. Nous présentons les caractéristiques des NoCs et outils existants les plus représentatifs et présentons quelques solutions technologiques à long terme. Enfin, nous expliquons les objectifs que nous nous sommes fixés et le contexte dans lequel nous nous sommes placés.

2.1.1 Présentation de différentes interconnexions

2.1.1.1 Introduction

Nous présentons ici les moyens d'interconnexion classiques et évaluons leur capacité à satisfaire les besoins des applications futures et les contraintes des circuits complexes de grande taille.

2.1.1.2 La communication point à point

La première communication que l'on peut considérer est la communication point à point entre les IPs. Cette solution n'offre pas de flexibilité, or, les IPs auront besoin de communiquer avec des IPs différentes en fonction des évolutions des besoins de l'application. Cette solution ne peut donc pas être retenue.

2.1.1.3 La connexion complète

Dans le cas de la connexion complète (*full connected*), toutes les IPs sont reliées les unes aux autres. C'est donc dans le principe un bon moyen de communication. Mais, le taux d'utilisation moyen des fils est généralement très faible, les fils sont sous exploités. C'est aussi le moyen de communication le plus coûteux en nombre de fils et en nombre d'interfaces par IP connecté. De plus, cette solution n'est pas extensible à un grand nombre d'IPs communicantes. Nous avons donc besoin d'un moyen de communication pouvant être partagé.

2.1.1.4 Le *crossbar*

Toutes les IPs sont reliées les unes aux autres par un *crossbar*. Celui-ci permet des communications parallèles ainsi que d'obtenir une grande bande passante. Cependant, la complexité de câblage est très grande dans une telle architecture. Elle augmente avec le carré du nombre d'éléments communicants : $O(n^2)$.

2.1.1.5 Le bus unique partagé

Le bus unique partagé possède plusieurs avantages. Tout d'abord, il est simple à maîtriser par les concepteurs car les opérations se font de façon séquentielle. De plus, la topologie du bus supporte directement le modèle de communication par adressage mémoire des IP microprocesseurs. Enfin, il a généralement une sémantique d'arbitrage simple du type arbitrage par priorité.

Cependant, il a aussi ses limites, puisqu'il s'agit d'une ressource partagée. L'arbitrage central du bus est un goulot qui implique un délai non négligeable lorsqu'il doit élire un maître parmi plusieurs. Le nombre de composants raccordés est donc limité. Le bus ne permet pas le parallélisme des communications (un seul transfert à la fois). Il n'est donc pas *scalable* (la bande passante n'augmente pas lorsque le nombre de composants augmente).

De plus, le bus est sujet à un problème électrique. En effet, les lignes du bus sont longues et le temps de charge augmente avec le nombre d'éléments raccordés du fait des capacités parasites de ces derniers. Sa fréquence de fonctionnement s'en trouve donc réduite et sa consommation électrique augmentée.

Enfin, le temps de propagations des signaux sur les longs fils cause la dérive des horloges et ainsi des problèmes de synchronisation.

2.1.1.6 Le bus hiérarchique

Le bus hiérarchique consiste en l'interconnexion de plusieurs bus, généralement de performances différentes, reliés par un pont. Comme exemples de bus hiérarchiques, nous pouvons citer : AMBA de ARM [9] et CoreConnect de IBM [10]. Le Bus hiérarchique possède plusieurs avantages sur le bus partagé. Les différents segments du bus hiérarchique ont des liaisons courtes et peu de composants IPs connectés, ce qui implique une faible capacité sur chacun de ces segments et donc une plus faible consommation, ainsi que la possibilité d'utiliser une fréquence plus élevée. Des transactions peuvent se faire en parallèle sur les différents segments de bus. Lors d'un transfert entre deux bus d'un bus hiérarchique, un pont permet la mise en communication entre ces bus. L'un

des bus asservit alors l'autre. Cependant, l'accès d'un bus à un autre au travers d'un pont, implique un coût en latence ; C'est pourquoi il est important de bien répartir les différentes IPs communicantes pour limiter l'utilisation du pont et ainsi favoriser le fonctionnement parallèle des bus. Ce découpage en segment est un premier pas qui tend vers l'approche réseau.

2.1.1.7 Le réseau distribué sur puce

Nous trouvons depuis déjà longtemps les réseaux distribués dans les architectures multiprocesseurs [11]. Le NoC s'inspire du modèle des réseaux d'ordinateurs et a été adapté au contexte du SoC. Cependant, toutes les fonctionnalités que l'on connaît dans les réseaux d'ordinateurs ne sont pas nécessaires dans les NoCs et auraient de plus un coût prohibitif en surface et en temps.

L'utilisation d'un réseau sur puce a pour premier intérêt de structurer les communications [8]. Les liens sont point à point, ainsi les propriétés électriques peuvent être optimisées et bien contrôlées ce qui peut permettre de réduire la consommation de puissance. Le partage des ressources de communication entre plusieurs flots de communication permet une meilleure utilisation des fils. De même que les bus, il facilite la modularité en définissant une interface standard. Mais vis à vis des bus, les réseaux ont généralement une plus grande bande passante et surtout supportent les communications concurrentes multiples. De plus, il est flexible et extensible [12, 13].

2.2 Les notions fondamentales dans les NoCs

2.2.1 Les éléments de base d'un NoC

La figure 2.1 montre un exemple de NoC en grille 2D de taille 2x2. Elle montre également les éléments qui composent un NoC et son environnement.

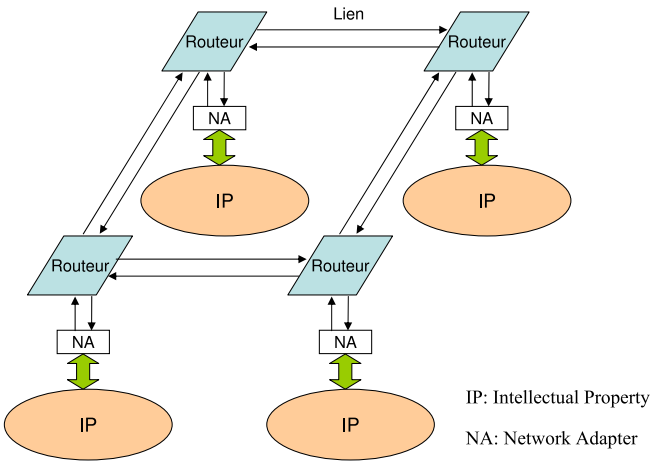


FIG. 2.1 – Les éléments d'un NoC

Les composants de base du NoC sont :

– **Les adaptateurs réseau** (NA) : Ils réalisent l'interface entre le protocole du NoC et celui des blocs IPs qui composent le système. Leur rôle est de séparer

le traitement (effectué dans les IPs) des communications (acheminées par le réseau). Un adaptateur réseau peut être séparé en deux parties, l'**interface réseau** (NI) et **l'adaptateur de protocole** ou *wrapper*.
- **Les nœuds routeurs** : Ils dirigent les données dans le réseau en accord avec le protocole choisi et intègrent la stratégie de routage.
- **Les liens** : Ils connectent les nœuds deux à deux, ce sont eux qui offrent la bande passante. Un lien peut consister en plusieurs canaux virtuels et peut être monodirectionnel ou bidirectionnel.

2.2.2 La topologie

La topologie désigne le graphe des liens entre les différents éléments du réseau. Une topologie régulière peut se caractériser par son nombre de dimensions (1D, 2D, 3D, etc.) Les topologies en 3D et supérieures peuvent poser un problème lors du routage sur silicium. Une topologie se caractérise également par sa forme (linéaire, grille, arbre, tore, multigrille, hypercube, etc.) et sa régularité ou non.

Les NoCs de la littérature utilisent très généralement des topologies régulières. La topologie standard est la topologie en grille 2D (*2D mesh*). Elle est adaptée à la technologie 2D des circuits intégrés et est *scalable*. Elle permet d'utiliser des stratégies de routage simple et donc peu coûteuses. Une topologie en arbre élargi (*Fat-tree*) a pour intérêt d'être *scalable* et d'offrir une latence pouvant être plus faible que la topologie en grille 2D. Une topologie en anneau, telle que celle de l'octogone utilisée par [14], permet de réduire la latence.

Une structure régulière présente l'intérêt d'être une topologie de structure mathématique simple, ce qui permet d'utiliser des règles de routage simples. Cependant, en pratique, le placement et la taille des composants IP du SoC permettent rarement d'intégrer une topologie régulière. Une topologie irrégulière permet plus de liberté et ainsi de tailler précisément le réseau requis. Elle peut être issue d'une topologie régulière qui a été retaillée au plus juste de façon à enlever les éléments non utilisés [15]. Une topologie irrégulière nécessite en revanche une plus grande attention pour le routage car les règles à appliquer ne sont plus triviales. Ainsi, dans la mesure du possible nous privilégions l'utilisation d'une topologie régulière, mais nous devons disposer d'une solution qui permette l'utilisation plus générale d'une topologie quelconque.

2.2.3 Les couches réseau

Le modèle OSI (*Open Systems Interconnection*) décrit le processus de fonctionnement d'un réseau sous forme de 7 parties appelées couches OSI.

Or, cette représentation n'est pas adaptée à la description d'un NoC car celui-ci n'intègre pas toutes ces divisions en couches. Il n'existe pas de modèle standardisé pour représenter les couches réseau d'un NoC. Nous pouvons utiliser le modèle représenté par la figure 2.2 qui est adapté de [16]. Cette figure nous montre le flot d'une donnée depuis une IP source vers une IP destination au travers des éléments d'un NoC. Elle indique de plus la correspondance avec les domaines de recherches et les couches réseau du modèle OSI.

Nous distinguons les domaines suivants :
- Le système : il représente l'application et l'architecture (IP et réseau entier) ;

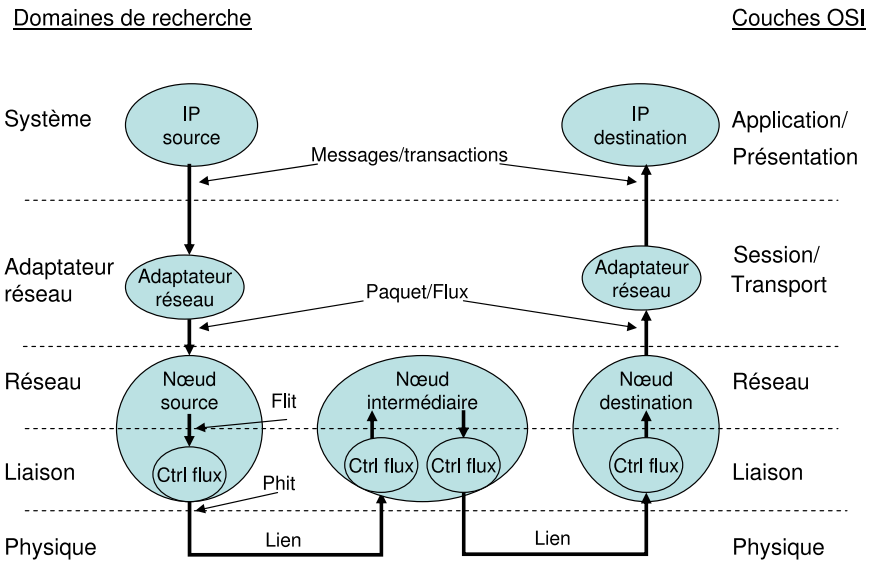


FIG. 2.2 – Les éléments du NoC et les couches réseau

- L'adaptateur-réseau : il adapte les protocoles de l'IP et du NoC (rôle du *wrapper*). Il découpe les messages en paquets, leur ajoute l'information indiquant leur destination et contrôle le flux des communications de bout-en-bout (rôle de la NI). Le contrôle de flux de bout-en-bout a pour but d'éviter que des données ne soient envoyées vers une destination alors qu'elle n'a plus de place disponible pour les recevoir.
- Le réseau : il définit la technique de routage employée pour l'acheminement des paquets à travers le réseau et la topologie ;
- La liaison : elle définit le protocole d'échange des bits sur les liens entre les routeurs (poignée de mains, crédit d'émission) ainsi que la technique de détection et correction d'erreur de transmission (bit de parité) ;
- Le niveau physique : c'est la couche de plus bas niveau d'un NoC, elle définit le moyen physique par lequel seront transportés les paquets, la nature de l'interconnexion (fils de cuivre), ainsi que la façon dont les données sont transportées (largeur des mots transportés en nombre de bits).

Un flit (*FLow control unIT*) correspond à la plus petite unité de contrôle de flux sur un lien. Un phit (*PHysical unIT*) correspond à la quantité de bits qui peut être transportée en une seule fois sur le lien (largeur du mot transporté). Le contrôle peut être réalisé avec une granularité de un ou plusieurs mots. Le contrôle se fait donc à la fréquence des flits, alors que les données circulent à la fréquence des phits. Si le contrôle est lent, un flit de plusieurs phits peut permettre d'échanger des données à une plus grande fréquence. En revanche, cela réduit le taux de commutation et nécessite un mécanisme pour signaler les flits qui ne sont que partiellement remplis [17]. Dans la majorité des NoCs, un flit contient un seul phit, c'est également le choix que nous avons fait pour notre NoC.

2.2.4 Le paquet

Les IPs doivent pouvoir communiquer en s'échangent des messages au travers du réseau. La taille des messages peut varier en fonction des applications. Pour une utilisation efficace et équitable des ressources du réseau, un message est souvent divisé en paquets avant d'être transmis. Un paquet est la plus petite unité de communication qui contienne des informations de routage et de séquençage. Sur la figure 2.3 nous pouvons voir comment est composé un paquet. Un paquet est composé de flits eux même formés de phits (sur l'exemple de la figure un flit contient deux phits). Le paquet débute par un entête (ou *header*), celui-ci transporte notamment des informations relatives au routage de ce paquet afin qu'il puisse être acheminé vers sa destination. Le paquet possède également une queue. Celle-ci contient une information qui indique la fin du paquet. Le corps du paquet peut transporter une charge utile, qui est le message à transmettre.

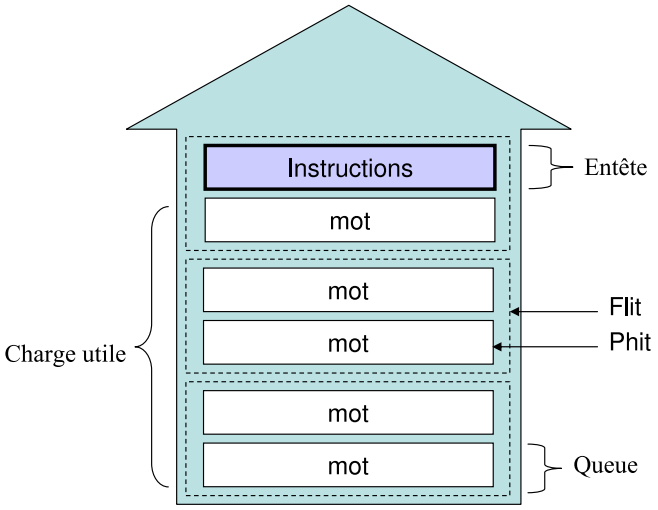


FIG. 2.3 – La structure d'un paquet

2.2.5 Le contrôle de flux par crédits d'émission

Nous distinguons un contrôle de flux sur deux niveaux des couches réseau, l'un de bout-en-bout du réseau et l'autre aux extrémités d'un lien. La technique la plus largement utilisée est le contrôle de flux par crédit d'émission [12]. Cette technique offre une indépendance vis à vis de la latence. Le lien peut être ainsi pipeliné sans que cela n'influe sur le mécanisme. Les crédits indiquent à l'émetteur la quantité de places libres dans la mémoire du destinataire. À l'initialisation l'émetteur dispose de la totalité des crédits. L'émetteur peut uniquement envoyer des données au récepteur dans la limite des crédits dont il dispose. De plus, l'émetteur décrémente son nombre de crédits chaque fois qu'il envoie une données vers ce destinataire. Lorsque le destinataire consomme les données, il l'indique à l'émetteur en lui renvoyant le nombre de crédits correspondant à l'espace nouvellement libéré dans sa mémoire. L'utilisation d'une technique de contrôle de flux n'est pas systématique, le flux peut être garanti, il n'est alors pas nécessaire de réaliser son contrôle. L'absence de contrôle de flux peut

aussi être un choix du concepteur en fonction des caractéristiques de son application, comme nous le verrons dans la section 7.5.

2.2.6 Le protocole

Le protocole, est le mécanisme par lequel les messages vont transiter dans le réseau. Le chemin (ou *path*) est le parcours qu'utilise le message pour aller d'une source à une destination dans le réseau. Il existe de nombreuses techniques de routage en fonction de la topologie visée et des compromis entre flexibilité et complexité [18]. Nous distinguons plusieurs aspects.

2.2.6.1 La technique de commutation

Nous distinguons deux types principaux de techniques de commutation (ou *switching*), la commutation de circuit et la commutation de paquets.

- La commutation de circuit consiste en des interconnexions qui commutent sur le circuit. Un chemin fixé est mis en place pour chaque paire source-destination pour la durée de la communication. Le message est ainsi transmis entièrement par ce chemin. Tous les composants du réseau le long du chemin sont engagés simultanément pour la durée de la communication. Lorsque la communication est terminée, le chemin est dissout ;
- La commutation de paquets, quant à elle, consiste à diviser la communication en plusieurs morceaux (paquets) et à envoyer chacun d'entre eux sur un lien de façon différente. Les délais de communications sont non-déterministes, du fait des contentions des ressources de communications. Ce type de commutation nécessite un contrôle de flux et de congestion.

La commutation de circuit est sûre, alors que la commutation de paquets est flexible. Nous avons fait le choix de la commutation de paquets et nous verrons dans la section 2.2.8 qu'elle bénéficie de moyens qui lui permettent d'être également déterministe.

2.2.6.2 La stratégie de routage déterministe ou adaptative

Dans le cas d'une stratégie de routage déterministe, le chemin est déterminé uniquement par la source et la destination. Avec le routage par la source, la source spécifie le chemin à emprunter pour atteindre la destination.

Dans le cas du routage adaptatif, le chemin est décidé au fur et à mesure du parcours de proche en proche [19]. Cela implique un mécanisme d'arbitrage, celui-ci peut être basé sur la congestion des liens. Le routage adaptatif a une implantation plus coûteuse dans les routeurs, mais permet une régulation dynamique de la charge dans le réseau. Cependant, cet aspect dynamique va à l'encontre de la prédictibilité du comportement du NoC. Comme nous visons des applications temps-réel, nous préférons un routage déterministe. Il est cependant envisageable d'utiliser un routage adaptatif, pour les trafics sans contrainte particulière.

2.2.6.3 Routage minimal ou non-minimal

Un algorithme de routage est minimal s'il offre toujours le chemin le plus court entre la source et la destination.

2.2.6.4 Le modèle avec retard ou perte

Dans le cas d'un conflit entre deux paquets pour accéder à une ressource du réseau (contentions), il existe deux possibilités, l'un des deux paquets est soit retardé, soit supprimé. Dans le premier cas, il est nécessaire de disposer de tampons de mémorisation dans les routeurs pour éviter que le corps du paquet en attente ne crée de nouveaux conflits. Dans le second cas, si l'on veut éviter que des paquets soient perdus, le paquet supprimé devra être retransmis. Il faut alors que des mécanismes soit mis en jeu pour signaler les paquets devant être retransmis. Ce mécanisme et la retransmission implique un coût. De plus, il faut qu'une sauvegarde de la donnée existe à la source. Nous verrons dans la section 7.5 qu'un choix délibéré de ne pas retransmettre les données peut se justifier si les données perdues peuvent être négligées.

2.2.6.5 La politique de mémorisation

La commutation de paquets présente 3 politiques de mémorisation :

- *store-and-forward* : Chaque routeur attend d'avoir reçu la fin du paquet et le stocke entièrement avant de le faire suivre.
- *cut-through* : Chaque routeur fait suivre les paquets au plus vite, mais il peut stocker tout le paquet s'il y a une contention.
- *wormhole* (ou trou-de-ver) : Cette politique consiste à acheminer les paquets sous forme de flits, ce qui permet de réaliser dans les routeurs une économie de mémoire et donc de surface en silicium. De plus, cette solution offre une faible latence pour la traversée d'un routeur puisque ce dernier ne mémorise pas le paquet en entier. Le paquet est ainsi acheminé en pipeline sur plusieurs routeurs. L'inconvénient du *wormhole* est que cet acheminement des paquets en pipeline peut conduire, en cas de contention d'un des paquets dans le réseau, à une contention en cascade de tout le réseau, voir a un interblocage (*deadlock*) [18].

La technique *wormhole* est la plus populaire, pour son faible coût en mémoire dans les routeurs et sa faible latence d'acheminement, ceci en dépit du risque de contention qu'elle fait courir et du contrôle de flux qu'il est nécessaire de mettre en œuvre. Dans la suite du document, nous ne nous intéresserons qu'à cette dernière.

2.2.6.6 Le risque d'interblocage

Un interblocage ou *deadlock* est la situation dans laquelle un ou plusieurs paquets peuvent rester bloqués indéfiniment dans le réseau. Elle peut apparaître si des paquets sont autorisés à détenir des ressources alors qu'ils en requièrent d'autres [18]. Dans le cas du routage *wormhole*, les canaux sont les ressources critiques.

Une façon de résoudre le problème de *deadlock*, est de faire de l'anticipation des paquets impliqués dans un potentiel *deadlock*. Les paquets concernés peuvent alors être re-routés ou supprimés. Dans le premier cas, cela nécessite des techniques de routage adaptatives. Dans le second cas, cela nécessite que les paquets puissent être retrouvés à la source et retransmis. Du fait des exigences de faible latence et de fiabilité, l'anticipation de paquet n'est pas utilisée dans la plupart des architectures de réseau direct.

L'absence de *deadlock* est généralement garantie par l'algorithme de routage. Un graphe de dépendance de canaux peut être utilisé pour développer un algorithme de routage sans interblocage (ou *deadlock-free*). En mettant les ressources du réseau dans un ordre et en exigeant que les paquets demandent et utilisent ces ressources dans un ordre strictement monotonique, l'attente circulaire (qui est une condition nécessaire à l'apparition d'un *deadlock*) est évitée.

2.2.6.7 Le codage des instructions et la règle de routage

Il faut distinguer la technique d'adressage et les règles de routage. La technique d'adressage définit comment sont codées les instructions de routage. Les règles de routage définissent la façon dont le paquet peut se déplacer dans le réseau.

Le Routage ordonné par dimensions X-Y : Il est très utilisé pour le routage déterministe sur les topologies en grille 2D car il est simple à réaliser. Il impose que les routeurs acheminent le paquet sur la dimension X puis sur la dimension Y. Ainsi, dans l'entête du paquet, le champ des instructions de chemin du paquet peut consister simplement en deux valeurs, X et Y. Ces valeurs sont exprimées soit en relatif aux coordonnées de l'émetteur (le nombre de sauts de routeur en routeur à réaliser dans chaque dimension pour arriver au destinataire), soit en absolu (les coordonnées de la destination). Dans le cas du routage relatif, à chaque saut, les routeurs décrémentent la valeur correspondant à cette dimension. Ainsi, arrivée à destination, ces valeurs sont nulles dans le cas du routage relatif et valent les coordonnées de la destination dans le cas du routage absolu. Cette technique de routage exclue le risque de *deadlock*.

Le codage *street-sign* : Le codage *street-sign* utilise une liste d'instructions du type Nord, Sud, Est, Ouest et descendre, indiquant la direction à prendre à chaque routeur, ou à des routeurs identifiés. Une fois exécutée, l'instruction courante est supprimée et les instructions restantes sont décalées pour que la suivante prenne sa place en tête de liste. Il est également possible d'utiliser un identifiant pour signaler à quel routeur doit être exécutée chaque instruction. Une direction par défaut est alors déterminée pour les routeurs pour lesquelles aucune instruction n'est donnée. La direction par défaut est généralement le port situé en face de celui par lequel le paquet est entré. Cette technique permet de réduire le nombre d'instructions nécessaires.

Le codage *street-sign* permet de mette en œuvre n'importe quelle technique de routage. Il ne garantit d'aucune manière l'absence des *deadlocks*. Celle-ci doit donc être résolue par la technique de routage ou par un ordonnancement. Nous verrons que cet aspect est résolu dans la section 4.4.

2.2.7 Les canaux virtuels

Les canaux virtuels (VCs) sont des canaux qui se répartissent entre eux un canal physique en utilisant des tampons de mémorisation différents. Le coût supplémentaire qu'ils impliquent en mémorisation et contrôle résulte en une augmentation de la surface et de la latence. Cependant ils présentent de nombreux avantages :

– Éviter les interblocages. Les canaux virtuels sont indépendants, ainsi en les utilisant avec des techniques de routage appropriées, ils permettent de briser des

cycles du graphe des dépendances de ressources [19, 18]. Les VCs facilitent notamment l'utilisation des algorithmes de routage adaptatifs.
- Optimiser l'utilisation des liens. Le partage du lien physique par plusieurs canaux virtuels peut permettre de faire une meilleure utilisation du lien. Ceci offre aussi une réduction du nombre de liens nécessaires.
- Augmenter les performances. Dally montre que les performances de l'interconnexion peuvent être améliorées en divisant la taille d'un tampon sur plusieurs canaux virtuels [20].
- Fournir des services séparés. Les canaux virtuels peuvent être utilisés pour séparer des trafics et leurs offrir des niveaux de priorité différents.

Nous verrons que le surcoût induit par l'utilisation de canaux virtuels peut être compensé par les avantages qu'elle apporte.

2.2.8 La qualité de service

Pour qu'une application puisse assurer des contraintes de bon fonctionnement et de temps-réel, il faut que les communications bénéficient d'une qualité de service.

La qualité de service (QoS) comprend plusieurs notions. Les services attendus sont que la communication doit être correcte, complète et respecter les contraintes de performances. Les contraintes de performances de la communication sont la bande passante, la latence ou la variation de l'instant d'arrivée (*jitter*).

Nous identifions deux classes de services, le trafic au mieux (BE pour *best-effort*) et le trafic garanti (GS pour *guaranteed services*). Le BE désigne un trafic qui n'offre pas de garantie sur les performances, mais généralement, il garantit tout de même que les données seront correctement et complètement transmises. C'est pourquoi, on utilise aussi le sigle GT *(guaranteed throughput)* au lieu de GS pour montrer que le trafic se différentie du BE par sa capacité à garantir la bande passante du trafic.

Les NoCs qui offrent un trafic GT utilisent généralement des variantes du TDMA (*Time Division Multiple Access*) pour construire une connexion [21, 22]. Le TDMA consiste à discrétiser le temps en périodes appelées slots de temps. Durant un slot de temps, la bande passante d'un lien est réservée pour une communication. Les ressources du réseau sont ainsi partagées temporellement entre les communications. La table de TDMA pour allouer les accès peut être soit dans les routeurs, soit dans les NIs. En effet, le slot de temps durant lequel le paquet est entré dans le réseau, ainsi que son chemin, déterminent les slots de temps des liens que ce paquet va utiliser. Les slots de temps non utilisés par le trafic GT peuvent être mis à disposition d'un trafic BE évoluant sur un autre canal virtuel. Ceci permet de faire une meilleure utilisation des ressources et de bénéficier à la fois du déterminisme d'une commutation de circuit et de la flexibilité d'une commutation de paquets.

Pour garantir le service, les auteurs de [23] proposent une alternative au TDMA qui consiste à allouer individuellement les fils d'un lien à des connections différentes. Cette approche est appelée SDM (*Spatial Division Multiplexing*). Les données sont sérialisées sur un nombre de fils proportionnel à la bande passante allouée à la communication. Cette approche est intéressante et pourrait être combinée avec l'approche TDMA.

2.2.9 La synchronisation du NoC

La distribution d'un arbre d'horloge sur une puce de grande taille pose des problèmes de synchronisation. De plus, l'horloge représente une part importante du routage et de la consommation. Il existe différents types d'approches concernant la synchronisation dans les NoCs.

- Le NoC entièrement synchrone a besoin d'une horloge commune sur chacun des éléments du NoC [22] [21].
- Le NoC peut aussi fonctionner avec des éléments synchrones qui s'échangent les données sur les liens de façon asynchrone ;
- Le NoC peut encore utiliser l'approche GALS (Globally Asynchronous Locally Synchronous) avec des îlots synchrones qui s'échangent des données de façon asynchrone.
- Et enfin, il existe le NoC entièrement sans horloge [24].

La consommation de l'arbre d'horloge d'un NoC synchrone peut représenter 54% de la consommation de puissance du NoC [25]. Un NoC sans horloge peut à priori réduire sa consommation, cependant, la mise en œuvre de l'asynchronisme nécessite des mécanismes qui peuvent être coûteux en surface. À l'heure où la consommation statique croit par rapport à la consommation dynamique, l'intérêt d'un système sans horloge n'est pas évident et reste à démontrer.

Dans le chapitre 6 nous nous intéresserons à cet aspect et présenterons notre approche pour surmonter les problèmes de synchronisation en proposant une approche intermédiaire.

2.3 Historique des NoCs

Nous présentons l'histoire encore jeune des NoCs. Celle-ci permet de situer dans le temps l'arrivée des différents réseaux et travaux liés à ce domaine [26].

Le NoC a pour origine les réseaux de multi-processeurs. Des réseaux comme le TRANSPUTER (*TRANSistor + comPUTER*) permettaient déjà en 1983 de faire communiquer des processeurs pour réaliser des machines parallèles [27]. Les processeurs étaient également les routeurs, un processeur complet possédant quatre canaux d'Entrées/Sorties permettant de le relier aux autres.

(2000) Pour parler de réseau sur puce, l'un des pionniers a été le réseau SPIN du LIP6 qui était nettement en avance sur son temps [12].

On a alors démontré l'intérêt des NoCs dans les architectures SoCs de demain [8, 28, 7]. Les travaux portaient essentiellement sur des simulations (systemC, outil NS2 (*Network Simulator*) [29, 30]), basées sur des générateurs de trafics qui au mieux simulaient des traces de trafic obtenues par simulation. L'attention était essentiellement portée sur les topologies et les routeurs [31].

(2002) Ensuite les statistiques de bande passante et latence en fonction de la charge de communication ont laissé place à des travaux visant à trouver le moyen de garantir le trafic [32, 22, 33] et d'interconnecter des ports d'IPs au réseau. Ceci a donné lieu à des travaux sur les interfaces réseau et les *wrappers* [34]. Le problème des horloges dans les circuits de grande taille a ensuite été considéré [24, 35].

(2004) Des méthodes et outils d'aide à la décision (topologie, taille des FIFOs, ordonnancement des TDMAs) ont fait leur apparition pour permettre d'alimenter

l'outil dédié jusqu'alors qu'à la génération du code VHDL du NoC [13, 36, 37, 38, 39, 40]. Le flot de conception du NoC a ainsi vu son niveau d'abstraction s'élever.

Nous voyons que les recherches sur les NoCs ont gravi les couches réseaux pour offrir un niveau d'abstraction de plus en plus haut.

L'avenir s'oriente vers une interaction croissante entre le système d'exploitation et le NoC, avec des adaptations mutuelles du NoC et de la QoS de l'application. Ceci suppose des aspects surveillance et reconfiguration du système.

2.4 Exemple de NoCs et d'outils pour les NoCs

Il existe un grand nombre de NoCs et il est impossible de tous les détailler. Ils ont évolué dans le même temps où nous travaillions sur notre NoC. Dans cette section, nous décrivons les NoCs qui nous semblent les plus représentatifs.

2.4.1 SPIN

Développé par le laboratoire LIP6 en 2000 [41, 12], le réseau SPIN (pour *Scalable Programmable Integrated Network*) et son routeur RSPIN ont constitué la première intégration complète d'un réseau sur puce à commutation de paquets. De plus, ce réseau a une topologie originale puisqu'il utilise une topologie en arbre élargi quaternaire. Cette structure hiérarchique permet de tirer parti de la localité et est *scalable*. Le routage des paquets dans le réseau peut être dynamique ou statique dans le sens montant de l'arbre, et il est obligatoirement statique dans le sens descendant.

Le routage dynamique permet de disposer de chemins alternatifs et ainsi de contourner des points de congestion, cependant, pour reconstituer le message, il est nécessaire de réordonnancer les paquets au niveau de la réception, du fait que les paquets n'utilisent pas tous le même chemin.

Les interfaces réseau réalisent le contrôle de flux de bout-en-bout par le passage de crédits d'émission et le réordonnancement des paquets à la réception dans le cas du routage dynamique. Ce dernier est réalisé à l'aide d'un tampon circulaire et d'un pointeur d'écriture qui permettent de réorganiser les paquets dans le bon ordre. Le pointeur de lecture s'arrête si le paquet requis n'est pas arrivé. Les paquets qui ne peuvent pas être stockés dans le tampon circulaire par faute de place (le pointeur d'écriture ayant atteint le pointeur de lecture) sont réexpédiés temporairement dans le réseau en attendant que de la place soit faite. La taille du tampon circulaire de réception est un problème critique, car elle doit être minimisée pour des raisons de surface et de consommation, mais néanmoins suffisante pour limiter les débordements.

Enfin, le SPIN dispose de *wrappers* VCI entre le réseau et les IPs qui permettent les conversions entre les deux protocoles de communication VCI et SPIN [34].

Le réseau SPIN est un réseau qui fait référence par son antériorité, cependant, ce réseau ne fournit pas de garantie en termes de latence et de bande passante pour l'acheminement des paquets. En effet, sous une grande charge de communication, le réseau souffre alors d'une grande latence [42].

Il n'est pas fait état d'outils destinés à aider le concepteur du réseau dans le cas de SPIN, cependant un modèle systemC du réseau est disponible pour permettre des simulations rapides.

Le LIP6 travaille actuellement sur des dérivés du SPIN, appelés DSPIN [43] et ASPIN. Le DSPIN utilise une topologie non plus en arbre élargi comme SPIN mais une grille 2D. Le routage est déterminé par la source et utilise la technique X-Y. Le ASPIN est un DSPIN avec des ports réseau asynchrones.

2.4.2 Æthereal

Philips Research a développé un flot de conception complet autour de son réseau Æthereal. Leurs travaux ont évolué en parallèle des nôtres et nous verrons que nous avons de nombreux points en commun.

Ce réseau utilise le routage *wormhole*, la technique de routage *street-sign* et le routage par la source. Il utilise une horloge unique qui doit offrir une notion de temps commune à l'ensemble des éléments du NoC. Le réseau Æthereal permet de garantir la bande passante d'un trafic [22, 33, 44]. Il utilise deux canaux virtuels, l'un pour la classe de trafics garantie GT *Guaranteed troughput*, et l'autre pour la classe BE (*Best effort*). Le trafic GT est mis en œuvre par l'utilisation de tables TDMA dans les interfaces réseau (NI) pour autoriser l'entrée des paquets dans le réseau selon un séquencement pré-ordonnancé afin de garantir leur traversée sans se rencontrer (absence de contention dans le réseau). Le BE utilise le reste de la bande passante.

L'équipe de recherche de Philips avait initialement mis ses tables TDMA dans les routeurs [22] au lieu de les mettre dans les NIs (il s'agissait donc d'une commutation de circuit), mais cette idée avait dû être abandonnée car cette solution nécessitait trop de surface.

Æthereal permet d'utiliser des connexions paramétrées (réservation de bande passante, latence bornée) entre les IPs qui désirent communiquer ensemble. À l'intérieur de cette connexion, la communication est faite par des transactions contenant des messages prédéfinis (Commande (lecture/écriture), Données, etc.). De plus, ces connexions permettent des émissions vers des récepteurs multiples (*multicast*). Enfin, des adaptateurs *shell* permettent la conversion du protocole de ce réseau vers les protocoles DTL et AXI.

Récemment, le groupe de recherche de Philips a développé des méthodes et outils pour automatiser les étapes de son flot de conception [36]. Il offre ainsi un flot prenant comme contrainte la bande passante et la latence des communications et génère la connexion des IPs au NoC, le routage (affectation des chemins aux communications) et l'affectation des slots de temps, ainsi que le dimensionnement des tailles des tampons. Enfin, il génère le code VHDL du NoC, ainsi que les fichiers de configuration de celui-ci. Le flot de conception de Æthereal [45] est celui montré dans la figure 2.4.

Pour faire coïncider les objectifs des phases de placement des IPs (*mapping*), de sélection du chemin et d'allocation des slots, les auteurs dans [45] proposent d'unifier ces phases de recherches par leur algorithme appelé UMARS (*Unified MApping,Routing and Slot allocation*). Ils utilisent une topologie de NoC calculée à priori. Leur originalité est de considérer l'existence de canaux de *mapping* virtuels reliant aux NIs du réseau, les IPs non encore placées. Ainsi, l'exploration de chemin sélectionne non seulement le chemin dans le réseau, mais aussi le canal de *mapping* virtuel qui relie l'IP à une NI. Après cette sélection, l'association de l'IP à cette NI devient alors définitive et les canaux de *mapping* virtuels reliant l'IP considérée aux autres NIs sont supprimés.

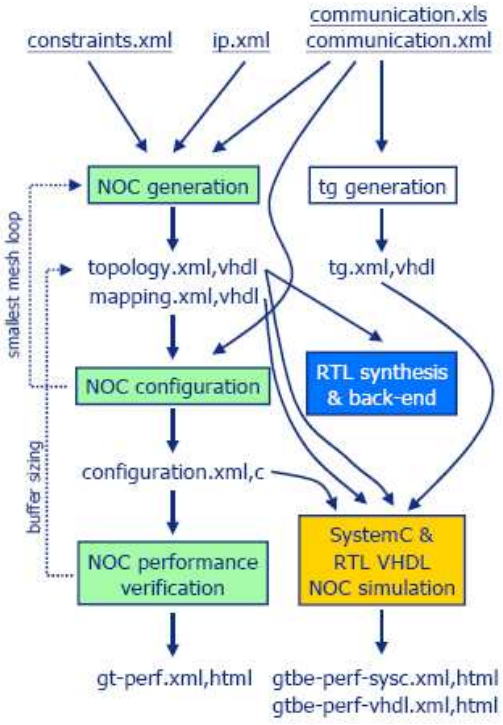


FIG. 2.4 – Le flot de conception de NoC de Æthereal

Le *mapping* réalisé ainsi suppose que les IPs sont interchangeables (compatibles en terme de tailles et de formes), or c'est rarement compatible avec la réalité.

De plus, des stratégies ont été développées afin de permettre d'utiliser plusieurs cas d'utilisation de l'application sur un même NoC [46].

Le flot de conception Æthereal a été appliqué avec succès sur un SoC pour la télévision numérique [25]. Nous notons que la flexibilité a un coût puisque l'architecture utilisant le NoC a une surface 4% plus importante et une consommation 12% supérieure à la réalisation initiale qui utilise des liens dédiés et quelques multiplexeurs.

2.4.3 QNoC

Le réseau QNoC (*Electrical Engineering Department, Technion - Israel Institute of Technology*) a pour caractéristique de séparer les trafics en quatre classes et de les répartir sur des canaux virtuels différents avec un ordre de priorité [32, 47]. Ces classes de trafics sont ordonnées en fonction de leur nature :

- Signalisation : les messages urgents et les paquets courts qui sont de type contrôle ou interruption. La plus haute priorité leur est donnée pour leur garantir une courte latence ;
- Temps-réel : les communications nécessitant une garantie en bande passante et latence, tels que les flux audio ou vidéo. Cette classe de trafic bénéficie d'une grande bande passante ;
- Lecture/Écriture : les communications de sémantique bus et les accès courts à de la mémoire ou à des registres ;

– Transfert de gros bloc de données : les transferts du type DMA. Cette classe de trafic a la plus faible priorité.

Un ordonnancement préemptif permet aux paquets appartenant à un trafic plus prioritaire d'utiliser les ressources utilisées jusque là par un trafic moins prioritaire.

De plus, QNoC autorise l'utilisation de topologies en grille 2D retaillées et contenant donc des irrégularités [15]. Le routage est déterminé par la source et utilise une technique X-Y améliorée pour permettre le routage dans une grille 2D non régulière. Enfin, il peut fonctionner avec des échanges synchrones ou asynchrones [48].

2.4.4 Arteris

Arteris est une jeune société française. Elle propose depuis 2005 la première solution commerciale pour concevoir des NoCs [49].

Son NoC est extrêmement paramétrable et tire partie de la séparation des couches réseaux pour faciliter l'optimisation de chacune d'entre elles de façon indépendante.

Il n'y a à priori pas de garantie de service en dehors de celle offerte par la simulation, il est cependant possible de favoriser une communication par rapport à une autre selon une technique de priorité au sujet de laquelle nous n'avons pas d'information.

Le NoC peut être synchrone ou bien utiliser une approche GALS avec des horloges mésochrones. Il utilise le *wormhole* dans le domaine synchrone pour minimiser la latence et le *store-and-forward* quand il change de domaine d'horloge.

Arteris utilise un protocole de transport propriétaire appelé NTTP (*NoC Transaction and Transport Protocol*) qui permet d'utiliser les transactions requête/réponse et d'intégrer les fonctionnalités traditionnelles des bus (adresse, transaction *load/store*, *burst*, bit de parité). Ceci lui permet d'être compatible avec un grand nombre d'interfaces standards (AMBA AHB, AMBA AXI, OCP 2.0). De plus, il permet de mixer sur un même NoC des composants qui utilisent des interfaces différentes [50]

Il est aussi possible d'utiliser une unité de translation pour connecter des NoCs ayant des paquets de formats différents.

L'offre d'Arteris consiste en la librairie Danube qui contient les IPs configurables de son NoC, ainsi que deux outils complémentaires, *NoCExporer* et *NoCCompiler*.

NoCexplorer est un environnement capable de capturer les besoins en communications entre les blocs IP et permet au concepteur d'analyser différentes topologies qui satisfassent les performances et le placement. Cette analyse repose uniquement sur la bande passante requise et n'est pas précise au niveau cycle.

L'outil *NoCcompiler* utilise la topologie et les autres données générées par *NoCexplorer* pour créer une base de données qui décrit le NoC. Il permet au concepteur de sélectionner les options voulues dont le format du paquet, les caractéristiques des communications des blocs IP (types de burst) et les paramètres de chacun des éléments du NoCs. Enfin, il est capable de générer un modèle systemC simulable précis au niveau cycle, ou des descriptions Verilog, VHDL, au niveau RTL. La figure 2.5 montre l'outil *NoCCompiler*.

Enfin, le Noc de Arteris sera intégré dans la plate-forme de développement de CoWare [51].

La force d'Arteris est de proposer un produit adaptable et qui se fond parfaitement avec les outils de conception existants.

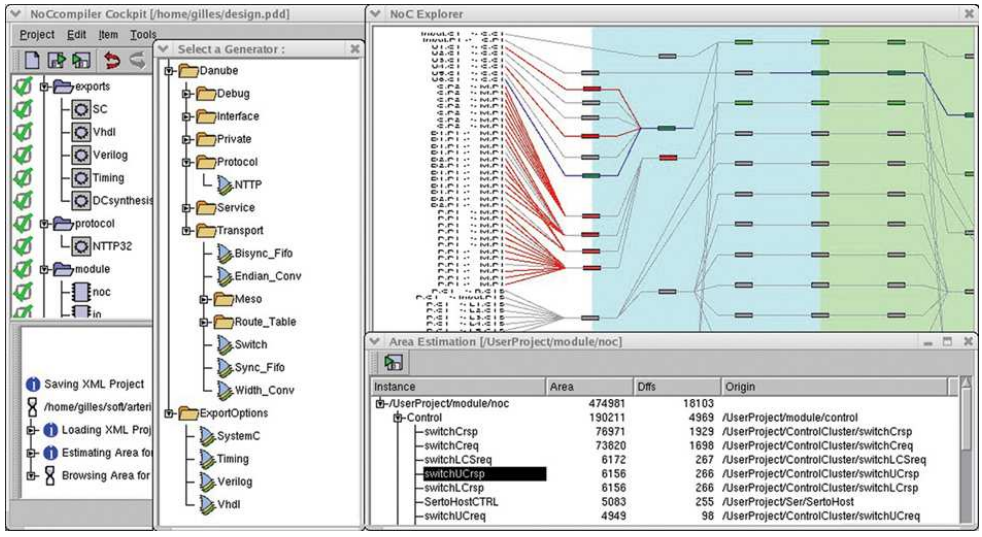


FIG. 2.5 – L'outil NoCcompiler de Arteris

L'engouement que Arteris rencontre avec bon nombre des *leaders* des outils de conception en électronique montre l'intérêt que suscitent les NoCs.

2.4.5 Spidergon

STMicroelectronics a également développé un NoC appelé STNoC utilisant une topologie baptisée Spidergon qui consiste en une topologie en anneau dans laquelle sont ajoutés des liens qui traversent l'anneau en diagonale [52, 53]. Ce type de topologie rappelle la topologie en octogone de Octagon [31]. Ce NoC ne présente pas d'innovation particulière, sinon sa topologie originale.

Notons qu'en début d'année 2006, STMicroelectronics a sélectionné Arteris pour réaliser les communications dans les SoCs de ses infrastructures sans fil [54].

2.4.6 CHAIN

Le réseau CHAIN (*Computer Science Department, The University of Manchester, UK*) possède une architecture entièrement asynchrone [24]. Il utilise des liens insensibles à la latence et un protocole double rails à quatre phases. Il utilise la commutation de paquet et le routage par la source. Il manipule un trafic BE. L'absence d'horloge lui permet de réduire sa consommation.

2.4.7 MANGO

Le réseau MANGO (*Technical University of Denmark*) est un NoC sans horloge[55]. Son architecture est donc entièrement asynchrone, cependant, elle permet de satisfaire le trafic GT en plus du classique BE. MANGO utilise un ordonnancement appelé ALG (*Asynchronous Latency Guarantee*). La garantie est obtenue sous certaines conditions. Il faut en effet que le délai maximum de transmission d'un flit soit connu et que l'intervalle inter-flits soit borné. Cette approche est intéressante, cependant, l'implantation d'un grand nombre de trafics à garantir, se traduit par l'ajout de nombreux canaux

virtuels ce qui implique une augmentation importante du coût en surface au niveau des routeurs.

2.4.8 ANoC-FAUST

Le CEA-LETI propose un réseau appelé ANoC (*Asynchronous Network-on-Chip*) qui s'intègre sur une architecture appelée FAUST (*Flexible Architecture of Unified System for Telecom*) dans le cadre du projet 4MORE [56, 57, 58]. Ce projet a pour but d'offrir une plate-forme ouverte pour les applications multimédia des terminaux mobiles de quatrième génération (4-G).

Le réseau est ici partagé sur plusieurs circuits de la plate-forme de prototypage. Il se répartit sur 2 FPGAs et deux circuits FAUST en technologie ASIC de 0.13µm. L'application visée par le projet 4More est un démonstrateur de modem MC-CDMA. Nous utiliserons cette application dans le chapitre 7.4 consacré aux expérimentations.

L'architecture de ANoC est basée sur une topologie en grille 2-D. Elle utilise la commutation de paquet, la stratégie *wormhole* et le routage par la source (*Adaptative Turn Model*). Pour les communications, ANoC utilise une logique asynchrone *Quasi-Delay-Insensitive* (QDI) 4 phases et un multi-rail. Il utilise deux canaux virtuels pour séparer les trafics. Ces trafics n'offrent pas de garantie. Le bon fonctionnement est vérifié par les simulations. ANoC dispose d'adaptateurs pour se connecter à un bus AHB. Les modèles de ce NoC sont en SystemC au niveau TLM (*Transaction level modeling*) et VHDL RTL.

2.4.9 xPIPES

Le réseau xPIPES [35] (*Stanford University, USA* et *University of Bologna, Italy*) est un réseau à commutation de paquets utilisant la technique *wormhole*, avec un routage par la source (codage *street-sign*).

La figure 2.6 montre le flot de conception dans lequel s'intègrent ses outils [37].

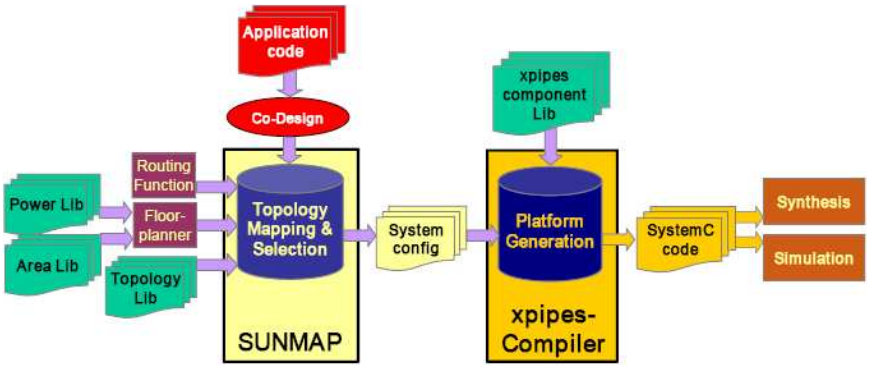


FIG. 2.6 – Les outils SUNMAP et xpipesCompiler du flot de conception de xPIPES

À partir d'une simulation initiale, l'application est décrite au moyen d'un graphe de tâches. Ce graphe indique la bande passante requise par chaque communication entre les paires de tâches. Chaque tâche est assignée à une IP. Les bandes passantes sont alors utilisées par l'outil SUNMAP [38, 39] pour générer une topologie pour le NoC. Il prend en considération la taille et le placement physique des IPs, pour placer

les liens, les routeurs et les NIs du NoC. Il teste plusieurs topologies et réalise une analyse de la surface et de la consommation du NoC. Il ajuste ses choix par une boucle itérative. Il supprime les liens et ports de routeurs non utilisés. Enfin, il calcule les tailles de FIFOs nécessaires.

À partir de la bibliothèque de composants et des paramètres choisis, l'outil xpipesCompiler [40] réalise la génération du code SystemC pour la simulation, ainsi que le code pour la synthèse.

C'est une approche intéressante qui permet de créer une topologie de réseau qui considère le placement des éléments sur le circuit.

2.5 Les solutions technologiques à long terme

Pour satisfaire les besoins de performance futurs, de nouveaux moyens de communication sont envisagés dans les SoCs. Ces moyens consistent à utiliser l'intégration verticale, les interconnexions optiques, les liaisons radio (RF), ou l'utilisation de guides d'ondes [3].

La forme du circuit silicium est essentiellement 2D. L'utilisation de la troisième dimension dans le silicium permettrait en effet de réduire la distance des communications. C'était un point fort du workshop NoC de DATE'06 [59].

Les interconnexions optiques [60] sont une option possible pour réaliser les communications globales sur les puces, alors que les communications locales continueraient à utiliser l'interconnexion métallique classique. La difficulté est d'intégrer cette solution optique sur le circuit silicium.

Les capacités d'intégration rendent possible l'utilisation de petits transmetteur-récepteur radio sur puce. Cette technologie s'appelle RoC (*Radio-on-Chip*). Elle pourrait permettre de simplifier le problème des communications entre des points éloignés du circuit. De plus, les communications radio sont prometteuses pour réaliser le test des circuits avant leur mise en boîtier. Les usines de fabrication de circuits électroniques réalisent des tests sur les circuits avant leur mise en boîtier. Or, les connections physiques au circuit sur les plates formes de test nécessitent du temps et peuvent causer la destruction du circuit. Il est ainsi proposé par [61] d'utiliser des communications par liaisons sans fil sur les SoCs pour envoyer les séquences de tests et recevoir les résultats.

Enfin, une alternative au RoC consiste à transmettre les signaux RF en utilisant un guide d'ondes métallique au lieu des antennes.

2.6 Conclusion

Le domaine de recherche des NoC est extrêmement large [16]. Nous avons évolué en même temps que les autres NoCs de l'état de l'art. Nous avons fait des choix qui se sont parfois révélés être ceux adoptés par l'équipe de recherche de Philips qui a conçu le NoC Æthereal. Notamment sur l'utilisation d'un TDMA. Nous nous différencions néanmoins d'eux par de nombreux aspects, par notre technique d'allocation des réservations des slots TDMA (section 4.4), ainsi que par notre utilisation de multi-réservations pour les communications appartenant à une même clique d'exclusion mutuelle (section 3.3.3). De plus, nous débutons notre flot de conception à un niveau

d'abstraction plus élevé par notre technique de dérivation des contraintes applicatives (section 4.3). Enfin, nous proposons une technique dans le cadre de la sécurité (chapitre 5), ainsi que des solutions permettant de s'accommoder des problèmes d'horloges (chapitre 6). Nous présentons tous ces aspects dans les chapitres suivants.

Chapitre 3

μSpider : Architecture du NoC, flot de conception et outil

3.1 Introduction

Un NoC performant et adaptable est nécessaire pour prendre en charge les différents types de trafics, notamment le trafic GT. Une des principales difficultés est la maîtrise des paramètres permettant d'obtenir le NoC approprié aux besoins de l'application. En raison de la taille de l'espace de conception (topologie, chemins, ordonnancement, etc.), il est illusoire de vouloir explorer toutes les solutions en fonction des différents compromis possibles. Il est donc nécessaire de disposer de méthodologies et d'outils pour assister le concepteur afin de le guider dans ses choix et de l'aider à concevoir le NoC adapté à ses besoins.

Ce chapitre présente, dans un premier temps, l'architecture générale de notre NoC. Nous présentons ses fonctionnalités, ainsi que ses composants de base, le routeur, l'interface réseau et l'adaptateur de protocole pour la connexion avec les bus des cœurs de processeurs ou IPs. Dans un second temps, ce chapitre présente notre flot de conception. Enfin, l'outil de conception, μSpider qui automatise ce flot, est présenté.

3.2 Architecture du NoC

3.2.1 Présentation

Notre NoC utilise la commutation par paquets pour la flexibilité qu'elle offre. De plus, il repose sur la stratégie de mémorisation *wormhole* pour son faible temps de traversée des routeurs et le faible coût en mémorisation dans ces derniers.

Par défaut, il est sans perte, en effet les paquets qui rencontrent une contention ne sont pas détruits, mais attendent dans le réseau avant de pouvoir progresser. De plus, il permet l'utilisation du contrôle de flux de bout-en-bout pour s'assurer, avant d'émettre, qu'il y a de la place disponible dans la FIFO de destination. Enfin, il est capable d'assurer le service aux communications ayant une contrainte de temps réel (trafic GT) grâce à l'utilisation de la technique de réservation de bande passante par TDMA (*Time Division Multiple Access*).

Un flit contient 1 phit (un mot), et un slot de temps du TDMA contient 2 flits ou phits.

3.2.2 Le routeur

Le nombre de ports de chaque routeur est paramétrable. Un routeur possède au minimum trois ports réseau, sinon il n'a pas lieu d'être. La mémorisation se fait par des FIFOs sur les ports d'entrée, avant le décodage.

La figure 3.1 nous montre la structure d'un routeur qui possède 3 ports. Nous distinguons les différents modules qui appliquent les différentes politiques.

Le contrôle de flux au niveau lien est réalisé en entrée et en sortie du routeur. La politique utilisée est le crédit d'émission [12].

Le module de routage décode les instructions de routage de l'entête du paquet et demande l'accès à l'arbitre du canal de sortie approprié. Les politiques de routage supportées sont le routage ordonné par dimensions X-Y, ainsi que le codage *street-sign* qui offre plus de liberté.

Le module arbitre décide quel canal d'entrée obtiendra l'accès au canal de sortie auquel cet arbitre appartient.

Le *switch* réalise la commutation qui permet la connexion physique d'un port d'entrée à un port de sortie. En pratique, nous avons réalisé le *switch* par des multiplexeurs. Nous avons jugé inutile de permettre à un paquet de ressortir par le même canal que celui par lequel il est entré. Ainsi, nous simplifions les arbitres et le *switch*.

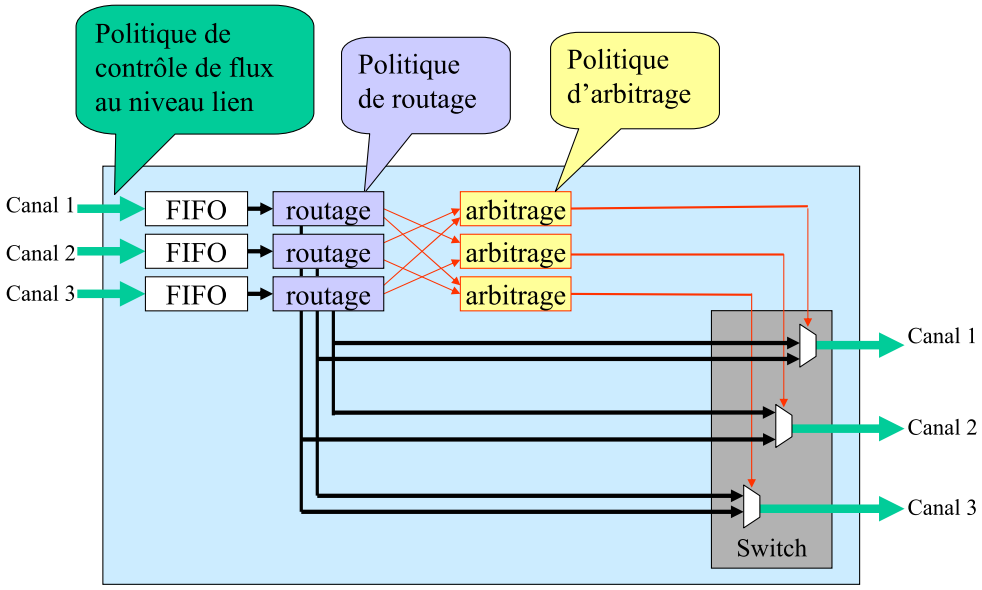


FIG. 3.1 – Architecture d'un routeur

Nous utilisons le terme canal au lieu de port, car ce routeur est capable de manipuler plusieurs canaux virtuels (VCs).

Les étapes de décodage de l'entête, d'arbitrage et de commande du *switch* sont pipelinées. De plus, nous utilisons les fronts montants et descendants de l'horloge. Nous obtenons ainsi une latence de traversé de deux cycles et une cadence d'un cycle.

Nous avons tout de même une limitation. Comme un entête nécessite deux cycles pour être traité, alors que les mots du corps du paquet nécessitent un seul cycle, deux entêtes qui se suivent à un cycle d'intervalle rompent le pipeline. C'est pour cette raison que pour le trafic GT, nous avons choisi une taille de slot de temps d'une largeur de deux cycles et que nous imposons aux NIs de n'émettre les entêtes des paquets que durant le premier cycle du slot de temps. Ainsi, la rupture de pipeline ne se produit jamais dans le cas du GT et le comportement des flits des slots reste facilement prédictible.

3.2.2.1 Les canaux virtuels

Notre NoC supporte les canaux virtuels (VCs). La flexibilité de l'architecture permet d'en spécifier le nombre. Cependant il est préférable d'utiliser un nombre limité de canaux virtuels car ils ont un coût en surface et en vitesse [62].

Les canaux virtuels ont un ordre de priorité correspondant à leur numéro. Le premier canal virtuel est le plus prioritaire pour obtenir l'accès au lien. Un canal virtuel n'est candidat à l'accès d'un port de sortie que s'il a au minimum un flit à envoyer et qu'il possède des crédits pour émettre sur le lien. Le plus prioritaire des canaux virtuels candidats obtient l'accès à ce port. Un signal est alors envoyé aux arbitres des canaux virtuels de priorité inférieure pour les geler.

Un canal bénéficie soit d'une classe de trafic GT ou BE. Seul le premier canal peut bénéficier de la classe de trafic GT car celle-ci nécessite le niveau le plus prioritaire. La bande passante est assurée par la réservation de slots de temps (TDMA).

Ainsi, nous pouvons proposer par exemple la répartition suivante pour les trafics.

- Un trafic de classe GT qui garantit le trafic temps réel prédictible (bénéficie de la garantie de service par TDMA) ;
- Un trafic de classe BE prioritaire pour les communications urgentes et courtes ;
- Un trafic BE qui utilise les slots non utilisés pour le trafic sans garantie de bande passante et de latence.

Le tableau 3.1 nous montre le numéro de canal virtuel, la classe de trafic et la règle appliquée pour chaque type de trafic. Notons que le trafic GT nécessite pour fonctionner une cohérence dans la réservation des slots le long de son chemin ;

Le fait que le trafic GT soit prioritaire sur les trafics qui évoluent sur les autres canaux virtuels ne suffit pas. En effet, il faut éviter que les communications du type trafic GT ne puissent se perturber entre elles. Ainsi, il est nécessaire de réaliser de manière cohérente les réservations des slots le long des chemins des communications GT. La recherche de la solution à ce problème fera l'objet de la section 4.4.

Trafic	Priorité (numéro du VC)	Classe	Réservation
Temps réel	1	GT	Réservation cohérente des slots consécutifs sur l'ensemble de son chemin
Message court et urgent	2	BE prioritaire	Au minimum 1 slot libre sur chaque lien de son chemin
Message sans garantie	3	BE	Pas de réservation

TAB. 3.1 – Répartition des trafics sur les canaux virtuels

Plus généralement, nous utilisons seulement 2 VCs pour transporter les trafics GT et le BE. Le trafic BE permet de tirer parti de la bande passante laissée libre par les slots non utilisés du trafic GT. Le trafic BE prioritaire peut être envisagé pour transporter des signaux de surveillance du réseau ou les paramètres de configurations du NoC. Cela évite de transporter ces informations sur d'autres NoCs comme cela est fait dans [63], où trois NoCs sont requis pour transporter les messages, les contrôles et les configurations.

3.2.2.2 La politique de contrôle de flux au niveau lien

Nous utilisons une paire de liens unidirectionnels en sens opposés. Il s'agit d'une méthode classique qui évite de réaliser un contrôle d'accès par jetons pour accéder à un lien bidirectionnel [64].

La figure 3.2 indique comment un paquet est transmis sur un lien unidirectionnel. Le signal *valid* permet de signaler que le mot du paquet présent sur le lien est valide. Le contrôle de flux au niveau lien (entre les routeurs ainsi qu'entre les NIs et les routeurs) se fait par crédit d'émission. L'envoi d'un crédit est codé sur un seul bit. Comme nous utilisons plusieurs canaux virtuels, les bits *ack_credit* indiquent les crédits retournés pour chaque canal virtuel. Il y a un bit *ack_credit* d'affecté à chaque canal virtuel. Un niveau 1 sur un bit indique que la FIFO de ce canal virtuel dans le port du récepteur s'est vidé d'un mot. L'émetteur possède un compteur pour chacun des canaux virtuels qui lui permet de savoir combien de places sont disponibles dans la FIFO de ce canal virtuel dans le récepteur. Il est initialisé avec la valeur de la profondeur de la FIFO du récepteur. L'émetteur décrémente ce compteur de 1 quand il envoie un mot et l'incrémente de 1 lorsqu'il reçoit un crédit. Il ne peut plus émettre si le compteur atteint 0. L'intérêt de cette technique est qu'elle permet de *pipeliner* par des registres l'envoi des mots sur le lien si ce dernier est long.

Le trafic GT ne nécessite pas de contrôle de flux particulier, car grâce au préordonnancement des réservations de slots de temps dans le cadre de la technique TDMA, il ne peut pas rencontrer de conflit par construction.

3.2.2.3 La politique de routage et la topologie

Pour chaque canal virtuel, nous pouvons spécifier la politique de routage. Notre NoC peut utiliser deux techniques de codage des instructions de routage, le routage ordonné par dimensions X-Y et le *street-sign*.

Le protocole de routage ordonnée par dimensions X-Y est couramment utilisé dans la littérature. Il est adapté uniquement à la topologie en grille 2D.

La technique de codage des instructions *Street-Sign* nous permet d'utiliser n'importe quel routage et d'être compatible avec les topologies irrégulières.

3.2.2.4 La politique d'arbitrage

Pour chaque canal virtuel, nous pouvons spécifier la politique d'arbitrage. Notre NoC autorise différents modèles :

- Sans arbitrage, utilisé dans le cas du trafic GT uniquement puisque les conflits sont résolus en amont par le préordonnancement ;
- Avec priorité fixe, ce qui est peu coûteux ;

– À priorité tournante (tourniquet ou *round-robin*) qui permet d'éviter les famines ;
– Selon une priorité fixée dans un champ de l'entête du paquet. En cas d'égalité entre plusieurs paquets l'arbitrage à priorité tournante est mis en œuvre.

Cette dernière politique d'arbitrage permet d'affecter à un paquet un niveau de priorité qui peut être fonction de l'importance ou de l'urgence des données qu'il transporte. Cette priorité est limitée puisqu'elle ne permet pas à un paquet plus prioritaire d'interrompre un paquet moins prioritaire qui est déjà engagé. Chaque technique d'arbitrage a une complexité différente et donc un coût différent en surface et en vitesse.

3.2.3 La structure du paquet

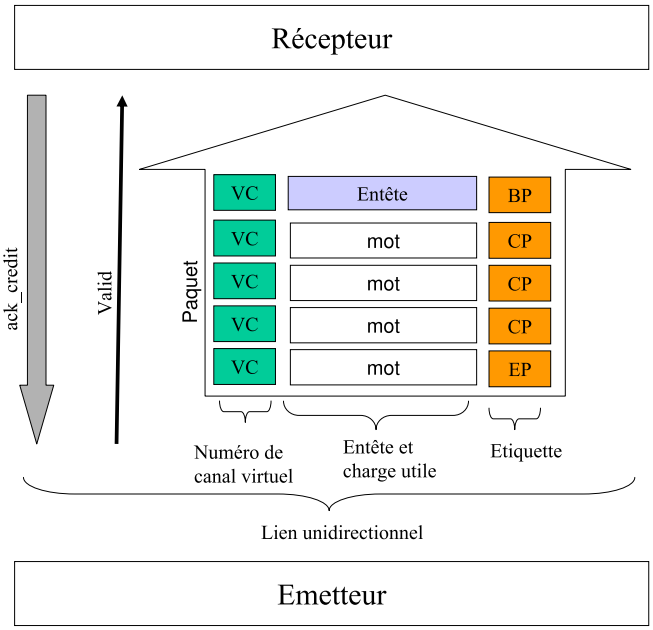


FIG. 3.2 – La transmission d'un paquet sur un lien unidirectionnel

La figure 3.2 montre comment est codé un paquet. Il est composé de flits. Chaque flit contient plusieurs champs.

Le champ « VC » indique sur quel canal virtuel ce paquet est transporté. Cette information est présente dans chaque flit du paquet car le paquet peut se faire interrompre par un autre circulant sur un canal virtuel plus prioritaire. Lorsque sa communication reprend, ses flits indiquent alors sur quel canal virtuel elle doit avoir lieu. En pratique, comme les paquets appartenant à des canaux virtuels différents sont stockés dans des FIFOs différentes dans les routeurs, le champ « VC » n'est pas mémorisé lors de l'entrée du paquet dans la FIFO et est recréé lors de la sortie du paquet. Cela nous permet de réaliser une économie sur la largeur des FIFOs.

Le champ « étiquette » est codé sur deux bits, comme précisé dans le tableau 3.2. Il permet d'indiquer le début du paquet pour que le décodeur du port d'entrée du routeur décode l'entête. Il indique aussi la fin du paquet pour que l'arbitre du port de sortie du routeur libère l'accès à la fin du passage de ce paquet.

Étiquette		Commentaire
Début	Fin	
1	0	Entête de paquet
0	0	Corps du paquet
0	1	Queue de paquet
1	1	Paquet de 1 flit (entête et queue à la fois)

TAB. 3.2 – Le champ « étiquette » du flit d'un paquet

L'entête du paquet contient des informations de routage destinées aux routeurs, ainsi que des informations pour la NI de destination que nous verrons dans la section suivante.

La figure 3.3 montre la manière dont les informations sont codées dans l'entête du paquet. Les informations dans l'entête du paquet sont :

- « Crédit » : Dans le cadre du contrôle de flux de bout-en-bout, ce champ permet à un consommateur de retourner des crédits d'émission vers son producteur.
- « ID du ExtraNiChannel de destination » : Indique à quel *ExtraNiChannel* de la NI adressée le paquet est destiné (ceci sera expliqué dans la section suivante) ;
- « Instructions de chemin » : Contient les instructions de chemin qui sont utilisées par les routeurs pour guider le paquet vers la NI de destination ;

La largeur de bits consacrée à chacun de ces champs est paramétrable. Les champs « crédits » et ID du « ExtraNiChannel » de destination nécessitent quelques bits. La largeur du champ « Instructions de chemin » dépend de la technique de routage et de la longueur des chemins dans le réseau.

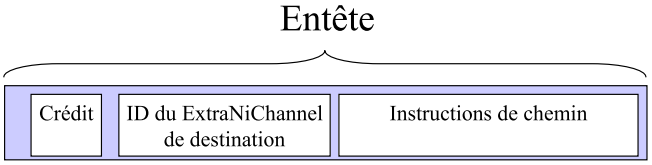


FIG. 3.3 – Les champs dans l'entête du paquet

3.2.4 L'interface réseau

L'interface réseau (NI) permet de relever le niveau dans les couches réseau pour passer au niveau adaptation réseau. Elle gère les messages. Nous utilisons des connections pour réaliser les communications.

Notre NI se distingue par sa capacité à permettre la réservation d'un slot par plusieurs communications. Ceci permet d'exploiter ce que nous appelons les communications mutuellement exclusives. Ce point sera expliqué dans la section 3.3.3. De plus, notre NI n'est pas limitée à deux canaux virtuels.

Elle possède un seul port réseau dans notre implantation et possède un ou plusieurs ports supplémentaires appelés *NIports*. Un *NIport* est destiné à connecter la NI avec un *wrapper* ou une IP.

La figure 3.4 montre l'exemple d'une NI qui a deux ports *NIports* pour permettre la connexion avec deux *wrappers* ou deux IPs. Nous voyons qu'elle dispose de deux

types de canaux (*chanel*) que nous appelons *ExtraNIchannel* et *IntraNiChannels*. Une connexion entre deux NIs se réalise par l'association d'un *ExtraNIchannel* de l'une avec un l'*ExtraNIchannel* de l'autre. Les *IntraNIchannels* permettent à deux *NIports* d'une même NI de communiquer directement entre eux sans passer par le réseau. Le nombre d'*ExtraNiChannels* et de *IntraNiChannels* de chaque *NIport* de la NI est paramétrable avant sa synthèse VHDL.

Le *NIport* utilise un simple protocole du type FIFO (vide, pleine, lecture, écriture) et une commande pour indiquer quelle FIFO est adressée.

L'ordonnanceur contrôle le déroulement des réservations dans le cas du trafic GT et arbitre l'accès des *ExtraNiChannels* au réseau en fonction de leur canal virtuel.

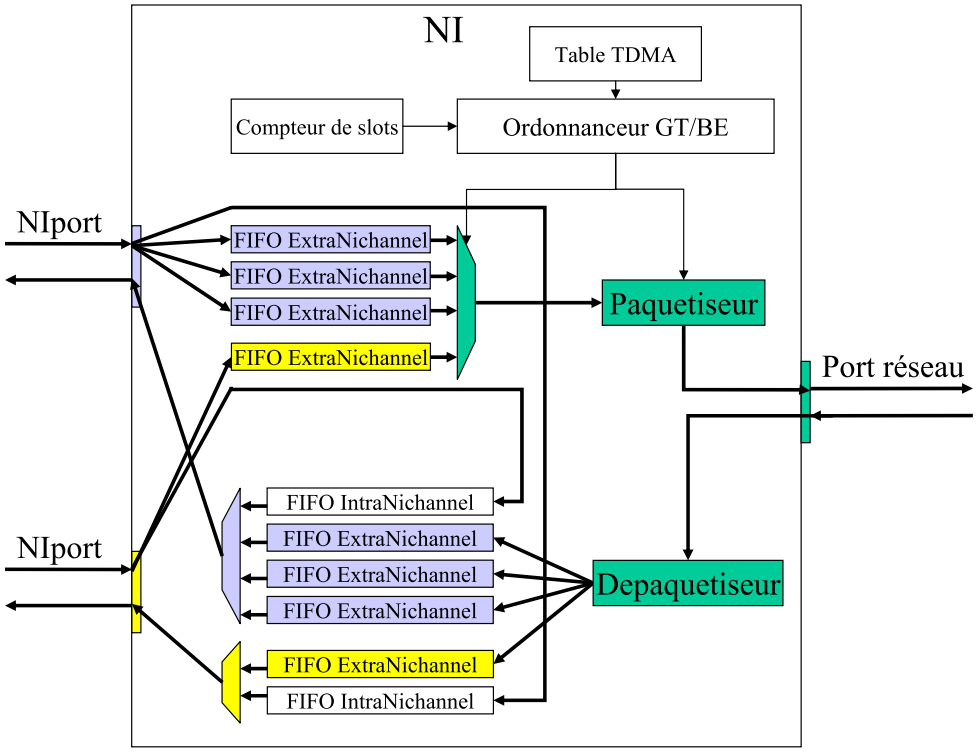


FIG. 3.4 – Une NI avec deux ports *NIports*

Notre NI peut manipuler un nombre générique de canaux virtuels. De plus, elle prend en charge le trafic GT par TDMA. Rappelons que la taille d'un slot de temps est de deux phits. Nous avons choisi cette valeur car elle correspond au temps de traversée de l'entête d'un paquet dans notre routeur. De plus, l'entête d'un paquet de trafic GT ne peut être envoyé que pendant le premier phit d'un slot réservé. C'est une taille plus petite que celle des slots de temps de Æthereal qui utilise des slots de 3 phits. Notre taille de slot plus petite nous permet d'avoir un découpage en slot plus fin et donc moins de slot sous utilisés (certains paquets peuvent ne pas occuper tous les mots du slot). De plus, cela permet d'avoir des tables dont la rotation est plus rapide, ce qui réduit le délai pour obtenir un slot réservé.

3.2.4.1 Les réservations des slots de temps dans les NIs

La technique de codage des slots réservés est la suivante. Nous disposons d'une table des réservations de slots, ainsi que d'une table des configurations des *ExtraNIchannels*.

Le nombre désiré de réservations dans la table des réservations de slots est paramétrable et peut aller jusqu'au nombre de slots de la table TDMA. Nous avons fait le choix qu'une zone de réservation soit inséquable dans la mesure du possible. En effet, il est plus efficace de créer un grand paquet plutôt que plusieurs petits, car un paquet implique à chaque fois un entête qui occupe de la bande passante.

De plus, les zones de réservations sont contigues. Pour définir la zone de réservation, seul le numéro du dernier slot inclus dans la zone réservée est indiqué. Ainsi, la zone débute juste après la fin de la zone de réservation précédente. Le champ « ExtraNIchannel ID » indique les *ExtraNIchannels* qui ont le droit d'émettre durant ces slots (le droit d'émission est codé par un bit à 1). Un bit est associé à chaque *ExtraNIchannel* (l' *ExtraNIchannel* numéro 0 correspond au bit de poids faible).

Le tableau 3.3 montre un exemple de table de réservations. La table est de 16 slots. Il y a 8 zones de réservations. Ainsi, nous voyons sur l'exemple que la première réservation (0) inclut les slots 2 à 3 et autorise le *ExtraNIchannel* 0 à émettre. La seconde réservation (1) va des slots 4 à 6 et autorise les *ExtraNIchannel* 0 ou 1 à émettre. On note que la quatrième réservation (3) n'autorise aucune émission (tous les bits sont à 0).

Réservation	Numéro du dernier slot réservé	ExtraNIchannel ID
0	3	000001
1	6	000011
2	8	000010
3	11	000000
4	12	000100
5	14	100001
6	15	010001
7	1	001000

TAB. 3.3 – Les réservations des slots de temps dans la table TDMA

3.2.4.2 La table de configuration des *ExtraNIchannels*

Chaque *ExtraNIchannel* peut être configuré indépendamment. Le tableau 3.4 nous montre un exemple de table de configuration des *ExtraNIchannels*.

Chaque *ExtraNIchannel* possède les paramètres suivants :

- Le crédit d'émission initial dont il dispose pour le contrôle de flux de bout-en-bout (il indique la place disponible dans la FIFO de réception de l'*ExtraNIchannel* de la NI distante vers laquelle ce *ExtraNIchannel* envoie des données) ;
- Le seuil d'envoi des données. Il permet de ne pas envoyer les données si elles sont trop peu nombreuses pour justifier l'envoi d'un paquet. Au dessus de ce seuil elles peuvent être envoyées.
- Le seuil d'envoi des crédits. Au dessus de cette quantité de crédits il est justifié de transmettre ce crédit.

– Le numéro de canal virtuel sur lequel ses communications évoluent.
– Les Instructions de chemin pour le routage au travers du NoC.
– L'ID du ExtraNiChannel de destination indique l'*ExtraNIchannel* de la NI distante auquel il s'adresse.

ID du ExtraNIchannel	Crédit d'émission initial	Seuil d'envoi des données	Seuil d'envoi des crédits	Numéro de canal virtuel	Instructions de chemin	ID du ExtraNiChannel de destination
0	0	0	5	0	00101111110	1
1	0	0	5	1	00101010110	0
2	0	0	5	0	00101000010	1
3	0	0	3	2	00000101010	2
4	0	0	3	2	00101111110	1
5	0	0	3	0	10111010110	3

TAB. 3.4 – La configuration des *ExtraNIchannels*

3.2.5 Adaptateur de protocole NoC-OPB

3.2.5.1 Présentation

Afin de pouvoir connecter au NoC des éléments qui utilisent un protocole différent de celui offert par les *NIports* de la NI, il est nécessaire de disposer d'un adaptateur de protocole appelé aussi *wrapper*.

Pour les besoins de nos expérimentations et la validation de nos concepts, nous avons mis en œuvre des *wrappers* pour permettre à des microprocesseurs MicroBlaze de communiquer au travers du NoC. Le MicroBlaze est un microprocesseur RISC que l'on peut synthétiser sur les FPGAs Xilinx à l'aide de l'outil EDK. Ce microprocesseur utilise pour communiquer le bus OPB et des interfaces FSL (*Fast Simplex Link*). Une interface FSL est une interface point à point d'une largeur de 32 bits. Elle utilise un protocole du type FIFO. Le Microblaze possède 8 ports FSL maîtres et 8 ports FSL esclaves. Ces interfaces extrêmement simples et rapides ont été utilisées par les auteurs de [65, 66] pour réaliser des communications entre plusieurs coprocesseurs et un Microblaze. Ce serait une façon simple de connecter le Microblaze au NIport de la NI, cependant, c'est une solution particulière, elle n'est pas générique. Nous avons préféré utiliser le bus OPB pour son mécanisme de type bus classique qui nous permet de connecter d'autres composants que le Microblaze, tels que des contrôleurs de RAM par exemple.

Le bus OPB (*On-chip Peripheral Bus*) est l'un des bus CoreConnect [10] conçu par IBM pour ses microprocesseurs PowerPC. Il permet de lier plusieurs maîtres à plusieurs esclaves. Ainsi, pour pouvoir réaliser l'interface entre un *NIport* de la NI et le bus OPB nous avons réalisé deux *wrappers*, un *wrapper* maître et un *wrapper* esclave. Ceux-ci intègrent des éléments appelés IPIF qui sont fournis par l'outil EDK de Xilinx et aident à la connexion au bus OPB.

Sur la figure 3.5, nous pouvons voir deux processeurs MicroBlazes connectés chacun à un bus OPB différent. Un NoC réalise la communication entre les deux bus OPB au moyen de *wrappers*. La liaisons entre une NI et un *wrapper* est appelée « pap » (pour point à point).

Le *wrapper* Maître (WRM) est un maître sur le bus OPB auquel il est connecté et peut donc communiquer avec des composants esclaves (RAM, ROM) reliés à ce bus. Sur la figure 3.6, nous pouvons voir l'architecture d'un *wrapper* Maître. Il possède deux FIFOs qui lui permettent de stocker le *burst* le plus long admissible entrant ou sortant. Ainsi, il n'envoie les données sur le bus OPB que lorsqu'il possède l'intégralité du *burst*. De la même façon, il ne fait la requête de lecture sur le bus OPB que quand il dispose de l'espace suffisant pour le recevoir. Ceci permet d'éviter de conserver longtemps le bus.

Le *wrapper* Esclave (WRS) est un esclave sur le bus OPB auquel il est connecté et peut donc être utilisé par un microprocesseur pour communiquer avec le réseau. Sur la figure 3.7 nous pouvons voir l'architecture d'un *wrapper* Esclave. Le *wrapper* Esclave possède des registres de statut qui permettent à un composant maître sur le bus OPB de consulter le statut des FIFOs des *channels* (*ExtraNIchannel* ou *IntraNIchannel*) de la NI pour savoir si des données sont disponibles pour être lues ou s'il est possible d'écrire. La partie basse de l'adresse spécifiée par le composant maître sur le bus OPB permet de sélectionner le *channel* sur lequel doit se réaliser une lecture ou une écriture.

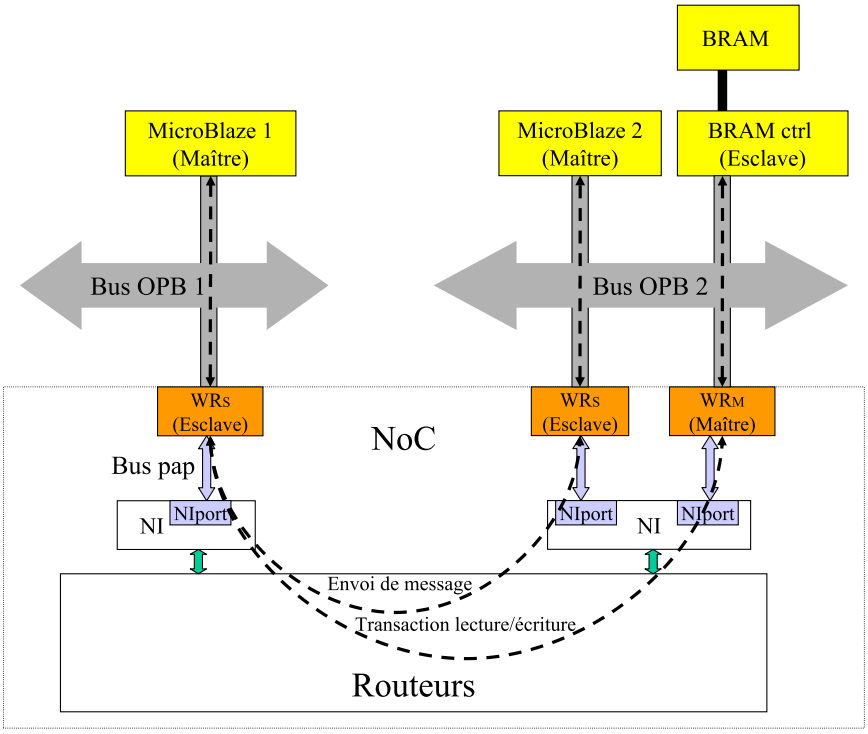


FIG. 3.5 – Deux microprocesseurs MicroBlaze et un contrôleur de RAM connectés par le NoC au travers de leurs bus OPB

Il existe deux façons de concevoir l'adressage des composants du système (*mapping mémoire*), soit un adressage mémoire global, soit un adressage local à chaque bus OPB. Nous avons choisi de réaliser des adressages locaux. Cela nous permet d'avoir une solution plus flexible. L'adressage sur chaque bus est ainsi indépendant.

Notre mise en œuvre permet d'utiliser deux types distincts de transmissions, le simple envoi de message et les transactions de lecture/écriture. L'envoi de message

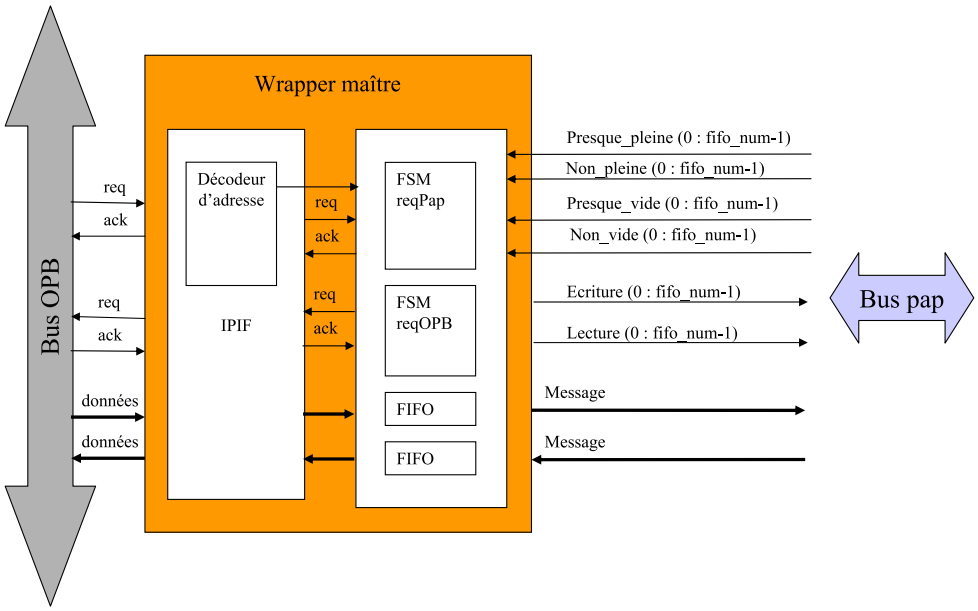


FIG. 3.6 – Le *wrapper* maître

est utilisé pour faire communiquer ensemble deux composants maîtres. Ce système fonctionne comme une boite aux lettres, les messages sont déposés dans les *wrappers* esclaves. Les transactions sont elles initiées par un composant maître qui envoie une requête à un composant esclave.

Pour réaliser une transaction de lecture ou d'écriture au travers du NoC, le MicroBlaze réalise une écriture dans le *wrapper* esclave. La partie basse de l'adresse sélectionne le *channel* à utiliser. Le contenu de l'écriture consiste en une requête de lecture ou d'écriture. Celle-ci sera transportée en tant que message dans un ou plusieurs paquets pour que le *wrapper* maître puisse réaliser cette requête sur le bus auquel il est connecté. Le format de codage de ces informations pour une requête de lecture ou d'écriture est spécifié dans les tableaux 3.5 et 3.6

Le *wrapper* maître a ainsi besoin de savoir s'il doit réaliser une requête de lecture ou d'écriture, le nombre de mots à lire ou écrire et l'adresse à laquelle cette requête doit être adressée sur le bus OPB. L'adresse indique l'adresse où se trouve l'esclave sur le bus distant et l'adresse locale visée dans ce composant.

$W/R = 0$																		nombre de mots du burst
adresse sur le bus OPB																		

TAB. 3.5 – Requête de lecture

$W/R = 1$																		nombre de mots du burst
adresse sur le bus OPB																		
donnée sur 32 bits																		
donnée sur 32 bits																		

TAB. 3.6 – Requête d'écriture

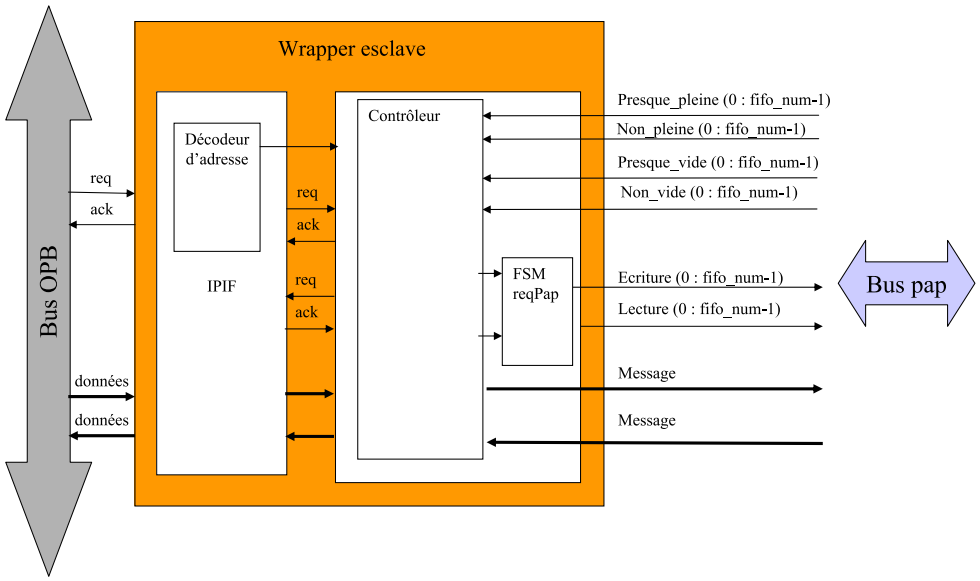


FIG. 3.7 – Le *wrapper* esclave

Il est nécessaire d'empêcher le MicroBlaze d'envoyer des données sur le *wrapper* esclaves lorsque la FIFO du *channel* d'émission sélectionnée dans la NI est pleine. De la même manière, il ne faut pas que le MicroBlaze puisse faire des demandes de lecture alors que celles-ci ne peuvent pas être satisfaite. De plus, nous voulons permettre au MicroBlaze de réaliser des écritures ou des lectures en continu sans avoir à vérifier continuellement le statut des FIFOs dans le *wrapper* esclave, pour savoir si la FIFO d'émission est pleine ou si la FIFO de réception qu'il utilise est vide. Pour ce faire, les registres de statuts de l'esclave indiquent également si les FIFOs d'émissions sont presque pleines et si les FIFOs de réceptions sont presque vides. Ainsi, la valeur qui fixe le « presque » permet au MicroBlaze de réaliser une série de lecture/écriture d'un nombre égal à cette valeur sans vérifier continuellement s'il a atteint la limite de la FIFO (vide ou pleine). Il doit cependant faire cette vérification lorsque la FIFO est entre presque vide et vide ou presque pleine et pleine.

Nous travaillons actuellement sur une nouvelle version du *wrapper* qui au lieu d'indiquer vide, presque vide, pleine et presque pleine, indiquera la valeur de la capacité libre/occupée dans les FIFOs dont le MicroBlaze désire connaître le contenu. Ceci réduira au maximum la consultation du statut des FIFOs. Les interruptions émisent par le wrapper esclave vers le MicroBlaze signaleront uniquement le passage d'une FIFO de réception de vide à non vide, et le passage d'une FIFO d'émission de pleine à non pleine.

3.2.5.2 Avertir le microprocesseur

Il y a deux solutions pour avertir le processeur d'un changement de statut des FIFOs dans le *wrapper* esclave, le *polling* et l'interruption. Dans le cas du *polling*, le microprocesseur doit aller interroger périodiquement les registres de statut dans le *wrapper* esclave. Cette solution est coûteuse en temps et en consommation pour le microprocesseur et le bus. Dans le cas de l'interruption, le microprocesseur reçoit une

interruption uniquement lorsque le statut a changé sur le *wrapper* esclave. La routine d'interruption lui fait alors réaliser une lecture du nouveau statut sur l'esclave. Nous avons choisi cette dernière solution, plus simple.

3.2.5.3 Chronogrammes

Nous montrons les trois types de transmissions sur trois chronogrammes. Deux mots d1 et d2 sont transportés.

Nous utilisons les notations suivantes pour décrire les échanges :

- (Rc+S) : Le MicroBlaze réalise une lecture sur le bus à l'adresse du registre de statut des FIFOs dans le *wrapper* esclave.
- (Rc+D) : Le MicroBlaze réalise une lecture sur le bus à l'adresse d'un *channel* du *wrapper* esclave lire une ou des donnée(s).
- (Wc+D) : Le MicroBlaze réalise une écriture sur le bus à l'adresse d'un *channel* du *wrapper* esclave pour transmettre une ou des données.
- (Wc+rqL) : Le MicroBlaze réalise une écriture sur le bus à l'adresse d'un *channel* du *wrapper* esclave pour transmettre une requête d'écriture.
- (Wc+rqE) : Le MicroBlaze réalise une écriture sur le bus à l'adresse d'un *channel* du *wrapper* esclave pour transmettre une requête de lecture.
- (R+D) : Le wrapper maître applique la requête de lecture sur le bus OPB2.
- (W+D) : Le wrapper maître applique la requête d'écriture sur le bus OPB2.

La figure 3.8 montre une transaction d'écriture depuis le MicroBlaze1 situé sur le bus OPB1 à destination d'un contrôleur de RAM situé sur le bus OPB2.

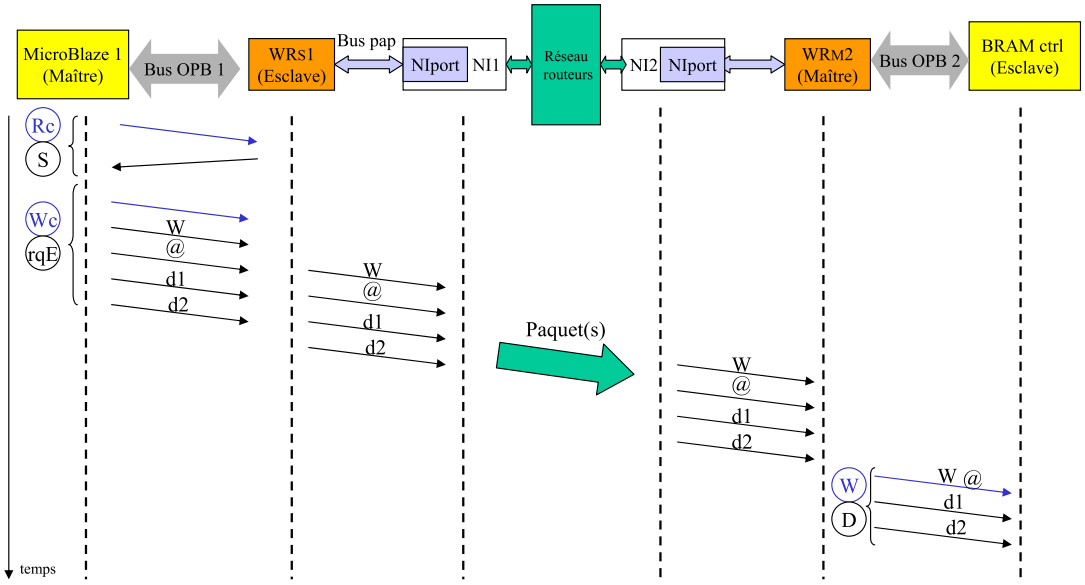


FIG. 3.8 – Écriture au travers du NoC et des deux bus OPB

La figure 3.9 montre une transaction de lecture faite par le MicroBlaze1 situé sur le bus OPB1 à destination d'un contrôleur de RAM situé sur le bus OPB2.

La figure 3.10 montre un échange de message d'un MicroBlaze1 situé sur le bus OPB1 à destination d'un MicroBlaze2 situé sur le bus OPB2.

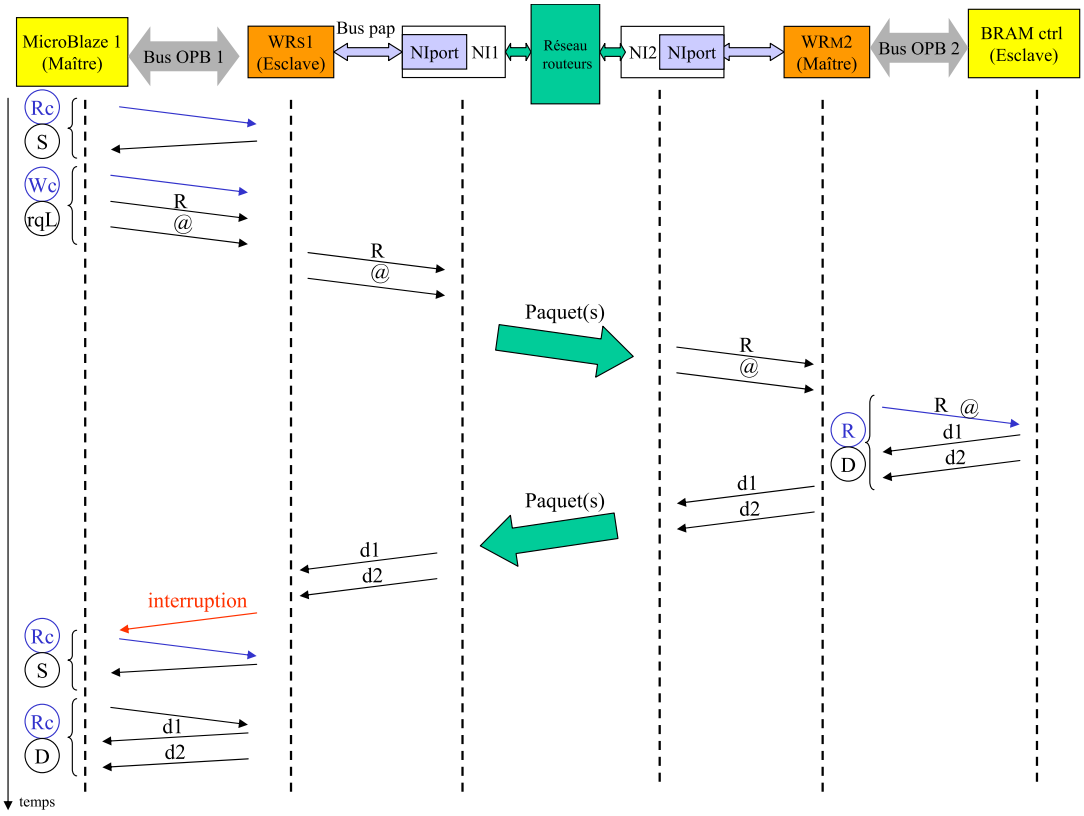


FIG. 3.9 – Lecture au travers du NoC et des deux bus OPB

3.2.5.4 Couche Logicielle

Nous avons développé des routines spécifiques en C qui permettent au MicroBlaze de dialoguer avec son *wrapper* esclave et de transmettre des requêtes au *wrapper* maître distant.

Le MicroBlaze peut ainsi :

- Consulter les registres de statuts du *wrapper* esclave (Rc+S).
- Réaliser au travers du NoC une requête de lecture ou d'écriture sur un composant situé sur un autre bus OPB (Wc+rqL) ou (Wc+rqE).
- Réaliser une lecture/écriture sur l'un des *channels* (*ExtraNIchannel* ou *IntraNIchannel*) du *wrapper* esclave (Rc+D) ou (Wc+D).

Les commandes sont :

- (Rc+S) : Lecture du registre de statut qui indique l'état vide ou presque vide des FIFOs de lecture et l'état plein ou presque plein des FIFOs d'écriture du *wrapper* esclave :
  `extern int WRAPPER_S_Read_Statut(BaseAddress);`
- (Wc+rqL) : Requête de lecture au travers du NoC :
  `extern void Wrapper_S_Read_(int BaseAddress, int channel, int remoteAddress, int *Data, int n);`

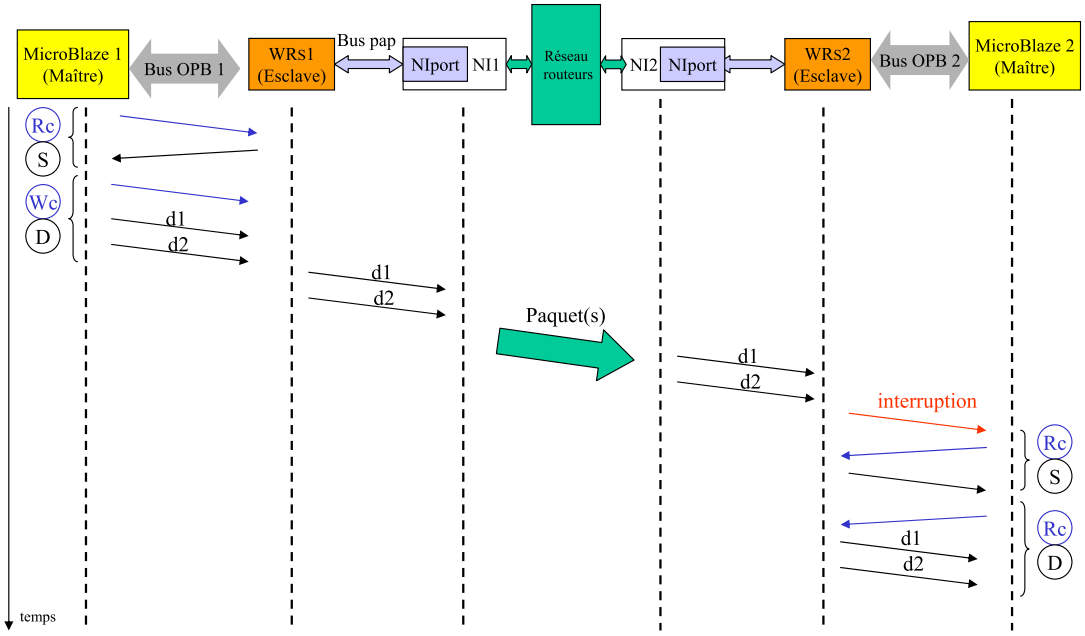


FIG. 3.10 – Envoi d'un message au travers du NoC et des deux bus OPB

- (Wc+rqE) : Requête d'écriture au travers du NoC :
  `extern void Wrapper_S_Write_BRAM(int BaseAddress, int channel, int remoteAddress, int *Data, int n);`
- (Rc+D) : Lecture de données dans l'un des canaux (*EXtraNIchannel* ou *IntraNIchannel*) du *wrapper* esclave :
  `extern void Wrapper_S_Recvn(int BaseAddress, int channel, int *Data, int n);`
- (Wc+D) : Écriture de données dans l'un des canaux (*EXtraNIchannel* ou *IntraNIchannel*) du *wrapper* esclave :
  `extern void Wrapper_S_Sendn(int BaseAddress, int channel, int *Data, int n);`

Les attributs de ces commandes sont les suivants :
- *BaseAddress* est l'adresse du *wrapper* que le MicroBlaze adresse sur son bus OPB.
- *channel* est le numéro du *channel* (*EXtraNIchannel* ou *IntraNIchannel*) que le MicroBlaze adresse dans la NI de ce *wrapper*.
- *remoteAddress* est l'adresse que devra utiliser le *wrapper* maître pour réaliser la transaction sur le bus auquel il est connecté.
- *Data* est la donnée transportée.
- *n* est le nombre de mots de données échangés.

3.2.6 Conclusion

Nous disposons d'une architecture complète, avec des routeurs, des interfaces et des *wrappers* OPB. De plus, cette architecture offre de nombreux paramètres :

- nombre de canaux virtuels ;
- technique d'arbitrage de routage et de garantie de service sur chaque canal virtuel ;
- largeur des données en nombre de bits ;
- nombre de ports de chaque routeur, ce qui permet de décrire toutes les topologies ;
- nombre de *NIports* de chaque NI ainsi que le nombre d'*ExtraNIchannels* ;
- choix des chemins ;
- choix des slots de temps pour les communications du trafic GT ;
- profondeur de chaque FIFO dans les routeurs et les NIs.
- utilisation ou pas du contrôle de flux de bout-en-bout.

Nous voyons qu'il est nécessaire de procéder avec méthode pour réaliser ces choix. C'est l'objet de la section suivante que d'expliquer le flot de conception associé.

3.3 Le flot de conception du NoC

3.3.1 Introduction

Un NoC doit être adapté aux besoins requis. Or, un NoC offre beaucoup de paramètres (topologie, fréquence d'horloge, mémorisation, stratégies de routage, arbitrage, contrôle de flux, etc.). De plus, il faut déterminer les chemins que les paquets doivent emprunter. Le choix des paramètres d'un NoC est un problème très complexe. La méthode retenue est donc une heuristique et s'appuie sur un ordre de choix avec un impact minimum. Les choix sont réalisés des moins contraignants aux plus contraignants. L'ordre des décisions dans le flot est donc important. Nous allons détailler les solutions que nous proposons ainsi que le flot que nous avons mis en œuvre. Nous nous intéresserons principalement au trafic GT car c'est celui qui présente la plus grande difficulté puisqu'il doit permettre de respecter des contraintes de temps réel.

3.3.2 Considération de la taille des FIFOs

Le coût du NoC est principalement dominé par la taille des FIFOs, si bien que leur minimisation doit être une priorité à chacune des étapes de la conception du NoC. Le coût en FIFO dans les NIs est particulièrement important lorsque des techniques visant à garantir le service sont intégrées. Cependant, cette observation nécessite d'être pesée car nous observons que l'utilisation du TDMA pour obtenir un trafic GT permet d'une part de réduire au minimum la profondeur des FIFO dans les routeurs puisqu'aucun conflit ne peut avoir lieu entre les paquets ; mais d'autre part, il implique un coût important en FIFO dans les NIs.

Les FIFOs dans les NIs ont deux rôles. Le premier est un rôle de découplage entre le NoC et l'IP connectée (productrice ou consommatrice). Le second est de cacher le voyage de retour du crédit d'émission.

La profondeur de FIFO requise pour le découplage est égale au nombre de mots réservés pour cette communication dans un tour de table de TDMA.

La FIFO chargée de masquer le voyage de retour du crédit d'émission (*round trip buffer*) doit avoir une profondeur égale ou supérieure au nombre de mots maximum que le producteur a besoin d'envoyer avant de récupérer des crédits, de sorte que

la contrainte de bande passante soit garantie. Cela dépend du temps de traversé du réseau dans les deux sens entre le producteur et le consommateur.

Ainsi, la taille de la table TDMA et la longueur des chemins sont des facteurs qui impactent le coût en FIFO et par conséquent la surface et la consommation d'énergie.

Le problème du dimensionnement de la taille des FIFOs des NIs dans le contexte d'un TDMA a été traité par les auteurs de [67, 68].

Les tailles de FIFOs nécessaires sont calculées automatiquement par notre outil à la fin du flot de conception, juste avant la génération du code VHDL.

3.3.3 Les communications mutuellement exclusives

3.3.3.1 Définition de l'exclusion mutuelle et de ses règles

L'exclusion mutuelle (ME pour *Mutual Exclusion*) de certaines communications permet de mutualiser les ressources qu'elles utilisent et donc d'augmenter les chances de trouver une solution d'allocation des chemins dans le réseau. Nous définissons deux types d'exclusions mutuelles dans le contexte des NoCs :

- **Fortes** : Les communications C_i et C_j sont fortement exclusives si les données de C_i (respectivement C_j) sont entièrement consommées avant l'émission des données de C_j (respectivement C_i). Au niveau du NoC, cela signifie que les crédits d'émissions sont totalement retournés et que la FIFO de réception est disponible.
- **Faibles** : Les communications C_i et C_j sont faiblement exclusives si elles ne peuvent pas se chevaucher dans le temps, cependant, cela ne garantie pas que les données de C_i (respectivement C_j) soient entièrement consommées avant l'émission des données de C_j (respectivement C_i).

Dans les deux cas, ces communications ME peuvent réserver les mêmes ressources de communication à des slots de temps identiques et ainsi faire une meilleure utilisation des liens, conduisant à une meilleure utilisation du NoC (figure 3.11).

Les communications ayant des exclusions mutuelles sont spécifiées par le concepteur durant la première étape du flot de conception et automatiquement organisées en cliques durant la seconde étape, avant le dimensionnement de la table TDMA. Une même communication peut appartenir à plusieurs cliques.

Durant l'allocation des chemins, la règle suivante est appliquée :

Plusieurs communications ne peuvent réserver un même lien durant un même slot de temps que si elles appartiennent à au moins une clique de ME commune.

3.3.3.2 La gestion particulière des crédits d'émission dans le cas des exclusions mutuelles

Le contrôle de flux de bout-en-bout se fait par la technique des crédits d'émission. En présence de communications mutuellement exclusives, la FIFO *round trip buffer* [68] dans la NI de destination peut être partagée par les communications mutuellement exclusives.

Cependant, des compteurs de crédits distincts doivent être utilisés quand des exclusions mutuelles faibles sont considérées. La figure 3.12 nous montre le partage de l'utilisation de la FIFO de réception par deux producteurs dont les communications sont mutuellement exclusives.

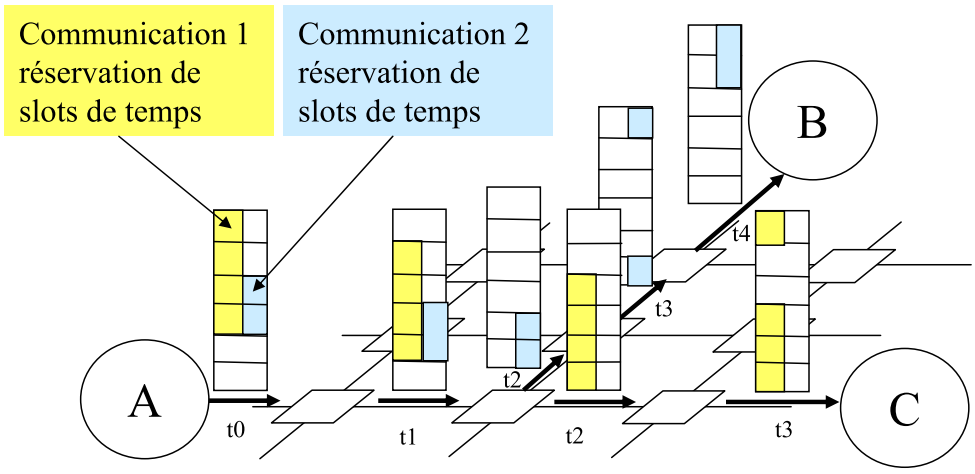


FIG. 3.11 – Réservations multiples de slots de temps sur les liens par des communications mutuellement exclusives

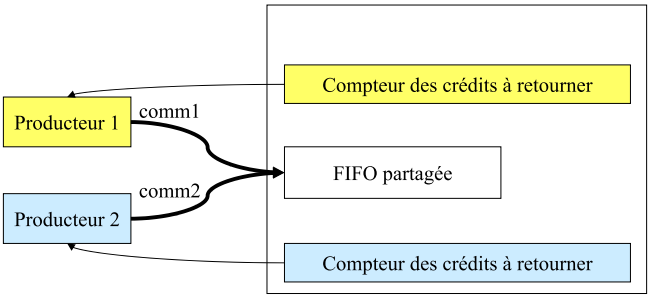


FIG. 3.12 – Partage de la FIFO de réception et utilisation de compteurs de crédits distincts pour des communications faiblement mutuellement exclusives

L'utilisation des exclusions mutuelles permet de mieux utiliser les ressources. Par exemple, si nous considérons des communications mutuellement exclusives entre un processeur et trois mémoires, RAM1, RAM2 et RAM3 avec des contraintes de bande passante BW1, BW2, BW3 respectivement. Avec l'approche d'Æthereal [22], la bande passante requise pour satisfaire ces trois communications serait la somme de ces trois bandes passantes (BW1 + BW2 + BW3). De plus, des fenêtres d'envoi distinctes seraient nécessaires dans la table de slots du TDMA. L'utilisation de « cas d'utilisation » [69, 25] permettrait d'utiliser des configurations distinctes pour chaque communication, mais la transition d'un cas d'utilisation à un autre impliquerait une perte de temps pour reconfigurer les tables TDMA des NIs. Avec une approche considérant les exclusions mutuelles entre les communications, la bande passante requise est inférieure à leur somme. Elle peut en effet être réduite pour atteindre la valeur de la plus grande bande passante des trois. De plus, nous pouvons utiliser des slots en commun pour ces communications, ce qui permet de réduire la taille de la table de TDMA.

La définition de communications fortement mutuellement exclusives peut être considérée comme une alternative à l'approche par cas d'utilisation. Avec notre ap-

proche, les communications des différents cas d'utilisations peuvent être vues comme mutuellement exclusives et placées simultanément.

3.3.4 Présentation de notre flot de conception

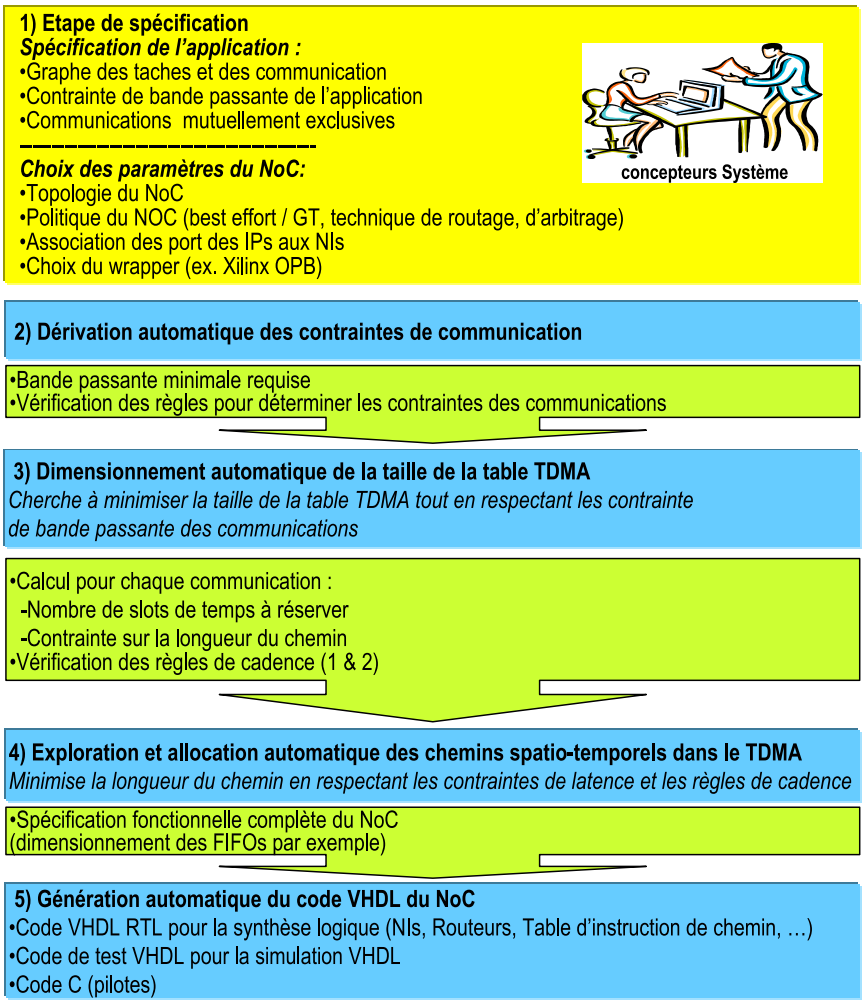


FIG. 3.13 – Le flot

Notre flot de conception est illustré sur la figure 3.13. Il se compose des étapes suivantes :

1) Étape de spécification

La première étape est une étape de spécification. Le concepteur doit alors décrire les spécifications de l'application qu'il désire implanter et faire un certain nombre de choix pour les paramètres du NoC. Ainsi, le concepteur renseigne un graphe représentant les tâches et les communications de son application. Il indique la contrainte de bande passante (contrainte temps réel) que le système devra satisfaire. Ce point sera détaillé dans le chapitre 4. Le concepteur signale de plus les cliques de communications mutuellement exclusives. De plus, il indique ses choix en terme de canaux virtuels, de service GT ou BE, de politique de routage, d'arbitrage et de contrôle de flux. Il fixe

la topologie du NoC et l'allocation des ports des IPs avec les NIs (*mapping*). Cette étape n'est pas automatisée car nous considérons que la connexion des IPs au NoC et donc leur placement ne peut pas être faite sans la connaissance de leurs formes et de leurs tailles. Trop souvent, les outils qui réalisent le placement des IPs dans le NoC, considèrent les IPs comme étant interchangeables en dépit des différences de tailles de celles-ci et des contraintes physiques des interfaces du SoC [70]. Nous notons tout de même que les auteurs de [38, 39] proposent une solution qui prend en compte ces aspects.

2) Dérivation automatique des contraintes de communication

Cette étape automatisée permet d'extraire les contraintes des communications à partir de la description de l'application. Elle vérifie des règles pour déterminer les contraintes des communications. Cette étape sera détaillée dans la section 4.3.

3) Dimensionnement automatique de la taille de la table TDMA

Nous cherchons à minimiser la taille de la table TDMA tout en respectant les contraintes de bande passante des communications. Cette étape permet d'obtenir pour chaque communication, le nombre de slots de temps à réserver et la contrainte sur la longueur du chemin. Ce point sera détaillé dans le chapitre 4.

4) Exploration et allocation automatique des chemins spatio-temporels dans le TDMA

Cette étape explore les chemins dans la topologie en tenant compte de la notion de slots propre à l'utilisation du TDMA. Elle vise à trouver pour chaque communication GT une solution de routage respectant ses contraintes de latence et de bande passante (règle de cadence). Ce point sera également détaillé dans le chapitre 4.

Nous obtenons alors la spécification fonctionnelle complète du NoC pour laquelle les communications GT ont un chemin réservé pour des slots de temps donnés. De plus, la profondeur de chacune des FIFOs est déterminée.

5) Génération automatique du code VHDL du NoC

Enfin, le code VHDL RTL pour la synthèse logique du NoC (NIs, Routeurs, Table d'instruction de chemin) est généré. De plus, notons que dans le cas de l'utilisation du MicroBlaze, nous fournissons également le code C qui permet d'utiliser des routines d'accès avec le NoC.

3.4 L'outil de CAO µSpider

3.4.1 Motivation

L'outil µSpider rassemble et met en œuvre les concepts exposés dans ce mémoire. Il nous a permis d'automatiser les tâches fastidieuses et ainsi de gagner en productivité en terme de conception de NoC. De plus, cet outil est un environnement ouvert pour l'exploration de nouvelles pistes. Enfin, celui-ci nous permet de disposer d'un démonstrateur, ce qui est un élément indispensable du point de vue dissémination.

La réalisation d'un tel outil nécessite de formaliser les éléments du problème et requière de la méthode. Sa conception a constitué une part importante du temps consacré à cette thèse. En effet, il a été nécessaire de formaliser le modèle du NoC, coder le VHDL des composants de celui-ci de façon à ce qu'il soit flexible, coder le générateur de code VHDL, coder l'outil qui réalise les décisions et les transformations

qui alimentent ce dernier, et enfin, valider le bon fonctionnement de l'ensemble sur une plate-forme FPGA.

3.4.2 Caractéristiques, performances

Une première version en langage PHP [71, 62] a été réalisée en 2004, elle ne traitait que la génération du code VHDL du NoC avec ses routeurs, mais sans les *Network Interfaces* et sans les adaptateurs de protocole. L'outil a ensuite été développé en Java pour disposer de tous les éléments et toutes les fonctionnalités qui seront décrites ci-après.

L'utilisation d'une approche objet est cruciale car elle est particulièrement adaptée à la modélisation de notre outil qui doit être évolutif. Le langage Java a été choisi pour sa simplicité à programmer, pour sa capacité d'évolution et le fait qu'il soit portable (multi-plate-formes).
Le format d'échange choisi est le XML (*eXtensible Markup Language*). C'est un format d'échange standard et facile à comprendre car humainement lisible. Il peut être édité par un simple éditeur ou dans des environnements XML.

De plus, pour chacun des modèles de description XML que nous manipulons, nous avons défini un document type appelé DTD (*Document Type Definition*). Celui-ci permet de définir la grammaire permettant de vérifier la validité du document XML. Nous vérifions ainsi que nos fichiers XML sont non seulement conformes (respect exact des règles de base de la norme XML), mais également valides (conforme à la grammaire que nous avons définie).

3.4.3 Modélisation dans µSpider

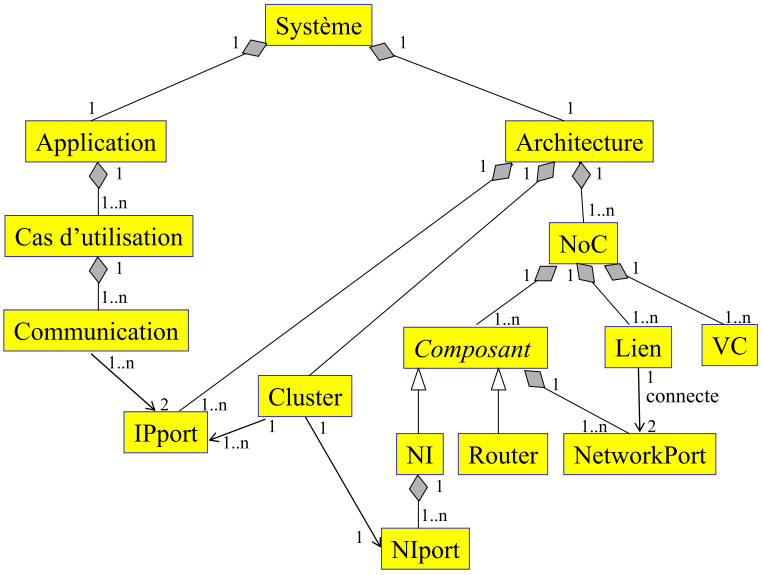


FIG. 3.14 – Relations entre les objets qui modélisent le système

La figure 3.14 montre de façon simplifiée les relations entre les objets principaux qui modélisent le système. Ce modèle s'inspire de UML. Les attributs des objets ne sont pas précisés, seules les relations sont indiquées.

Le système se compose d'une application et d'une architecture. L'objectif est de satisfaire les contraintes de l'application sur l'architecture.

Une application peut posséder plusieurs cas d'utilisation ou modes de fonctionnement. Un cas d'utilisation définit les communications qui lui sont propres. Deux ports d'IP (IPports) sont associés à chaque communication. Notre représentation permet de décrire plusieurs cas d'utilisation pour une application, mais pour le moment, nous n'en manipulons qu'un seul.

Une architecture est composée d'un ou plusieurs NoCs. Chaque NoC est composé de composants qui sont soit des routeurs soit des NIs. Chaque composant a des ports réseau *NetworkPort*. Le NoC est également composé de liens qui connectent une paire de *NetworkPort*. Le *cluster* associe des IPports avec les NIports d'une NI.

La figure 3.15 montre la façon dont les contraintes de bande passante et de latence des communications d'un cas d'utilisation peuvent être importées dans l'outil sous la forme d'un fichier *Excel*. De plus, il est possible de spécifier le canal virtuel sur lequel chacune de ces communications évoluera. Si une communication spécifie un canal virtuel qui assure le trafic GT, alors la communication bénéficiera de ce service.

	A	B	C	D	E	F	G	H	I
1									
2			READ			WRITE			
3	Initiator port	Target port	Bandwidth (MBytes/s)	BurstSize (Bytes)	Latency (nano sec)	Bandwidth (MBytes/s)	BurstSize (Bytes)	Latency (nano sec)	VC QoS
4	Am	Vs	100	4	1500				0
5	Am	Ws	50	4	1500				0
6	Am	Xs				30	16	1500	0
7	Bm	Zs	2	80	1500	10	8	1500	0
8	Cm	Xs	90	160	1500				0
9	Cm	Ys				3	4	1500	0
10	Cm	Zs				10	4	1500	0
11	Dm	Ys	50	8	1500	10	2	1500	0
12	Em	Zs	50	4	1500	70	8	1500	0
13	Em	Us	40	4	1500				0
14	Fm	Us	40	4	1500				0
15	Gm	Ys				30	8	1500	0
16									
17									

FIG. 3.15 – La spécification des communications dans le fichier *Excel*

La figure 3.16 montre un exemple de fichier XML qui décrit précisément une architecture. Nous retrouvons la représentation de la figure 3.14. C'est une description hiérarchique. Ainsi, elle permet de distribuer facilement les paramètres génériques du NoC à tous ses composants. Pour chaque NoC, ses canaux virtuels sont décrits et ordonnés selon leur ordre de priorité. Pour chacun de ces canaux virtuels sont précisés les politiques d'arbitrage, de routage et de contrôle de flux ainsi que le type de trafic GT ou BE. Un minimum d'un canal virtuel est requis.

Les composants (qui sont soit des routeurs soit des NIs) ainsi que leurs ports réseau, sont également décrient. Enfin, sont définis les liens qui par leurs extrémités relient les ports réseaux des composants.

Il est possible d'associer plusieurs ports d'IPs (IPport) avec une même NI. Un fichier au format XML du type *cluster* permet de définir ces associations entre les IPports et les NIs. De plus, il existe des fichiers XML qui décrivent les *cliques* des communications mutuellement exclusives, ainsi que des niveaux intermédiaires de la description du NoC.

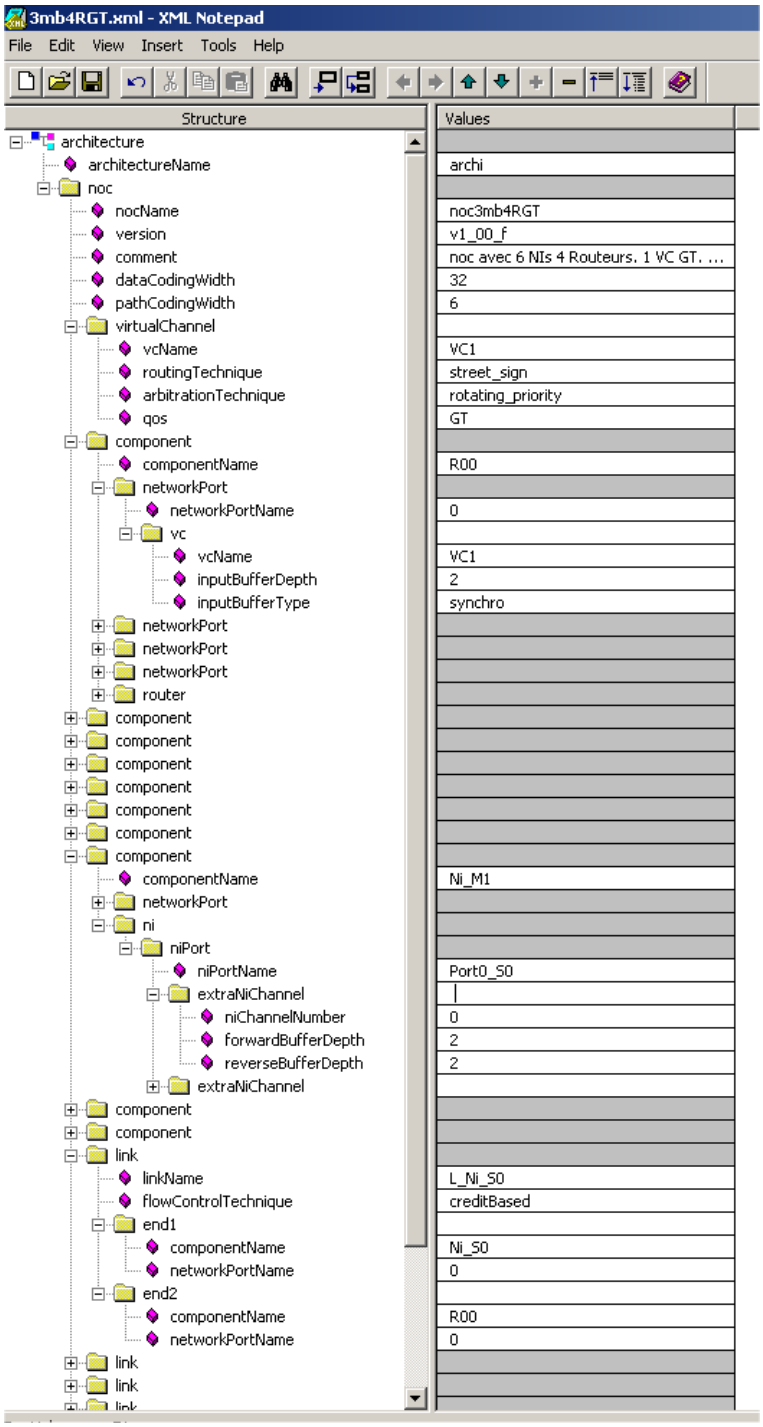


FIG. 3.16 – La description de l'architecture du NoC

3.4.4 Représentation du flot de conception dans µSpider

Le programme de cet outil possède des modèles qui décrivent ce que sont un NoC, un routeur, une NI, une communication, etc. Il possède aussi des modules. Ces modules

sont de trois types : importation, exportation et transformation. Ils permettent à l'utilisateur de rentrer ses spécifications architecturales et applicatives, de réaliser les transformations qui vont réaliser des décisions selon des paramètres, et enfin de sauvegarder les résultats de ces transformations (configuration, architecture du NoC enrichie, code VHDL du NoC)

De plus, il est possible d'ajouter facilement de nouveaux modules.

3.4.5 Étapes du flot de l'outil µSpider

Les principales étapes sont les suivantes :
- saisie/importation des contraintes de l'application ;
- génération ou importation d'une architecture de NoC ;
- connexion des IPs avec les NIs (*mapping*) définie par des *clusters* ;
- dimensionnement de la table TDMA ;
- affectation du nombre de slots à chaque communication ;
- exploration et affection des chemins et des fenêtres d'envoi (routage spatio-temporel) ;
- dimensionnement des FIFOs ;
- génération des configurations ;
- génération du code VHDL.

De plus, l'intégration du NoC dans l'environnement EDK est facilité. En effet, les fichiers VHDL ainsi que des fichiers spécifiques sont créés de manière à permettre à l'utilisateur d'ajouter le NoC en tant que simple bus par un simple copier-coller de ces fichiers dans la bibliothèque appropriée.

3.4.6 Les commandes de l'outil µSpider

Les commandes de l'outil µSpider sont entrées sous forme d'arguments. Il est ainsi facile d'utiliser ces commandes sous forme de script, de les sauvegarder et de les réutiliser. Ceci permet de gagner du temps pour entrer les paramètres. L'interface utilisateur pour spécifier les paramètres du NoC se fait à l'aide d'un éditeur XML.

Il est possible de générer une topologie régulière de NoC, d'exporter cette topologie en format XML, de la personnaliser et de la réimporter. L'outil propose la génération automatique de topologies en grille 2D avec des paramètres tels que le nombre maximum de ports des routeurs, nombre de lignes, nombre de colonnes, avec ou sans utilisation des bords extérieurs de la grille. Des NIs sont connectées aux ports locaux des routeurs et sur les bords extérieurs du NoC si cela est spécifié.

La figure 3.17 montre un exemple de script pour exécuter le flot.

Ces commandes réalisent les actions suivantes :
- -I_RWEXCEL *fileName* : Importe au format *Excel* les contraintes de lecture et écriture des communications de l'application ;
- -I_HW *fileName* : Importe au forme XML la description matérielle du NoC ;
- -T_RWFR *Yes/No* : Transformation des communications lecture et écriture en connections *Forward* et *Reverse*. *Yes/No* pour activer/désactiver le contrôle de flux de bout-en-bout ;
- -T_CIPPORT : Crée la liste des IpPorts depuis la liste des communications lecture et écriture

```
-I_RWEXCEL .\project\tracking\tracking03.xls
-I_HW .\project\tracking\trackingTopology.xml
-T_RWFR Yes
-T_CIPPORT
-I_CLUSTCONST .\project\tracking\IpPortClusterConstraint.xml
-T_CLUST_B
-T_SLOTPART_A Yes 1
-T_UCOMM
-O_CLUST .\project\tracking\exportCluster.xml
-T_CARC
-T_PALLOC 7 3 1 1 0 0 0
-T\_BUFF  Yes
-O_USLOT .\project\tracking\unidirComm.xml
-O_HW .\project\tracking\exportTopology.xml
-O_VHDL .\project\tracking\ x
-help
```

FIG. 3.17 – Exemple de script

- -I_CLUSTCONST *fileName* : Impose à des IpPorts des contraintes de regroupement en *clusters* et association à une NI.
- -T_CLUST_B : Avant le partitionnement en slots. Regroupement des IpPorts dans un *Cluster*. Chaque IpPort sera associé à une NI dans le NoC.
- -T_SLOTPART_A *Yes/No initialSlotTableSize* : Après le regroupement en *clusters*. Le partitionnement en slots calcule en nombre de slots les contraintes des communications unidirectionnelles entre les IpPorts situés dans des *clusters* différents. *Yes/No* pour activer/désactiver le contrôle de flux de bout-en-bout. *initialSlotTableSize* = taille initiale de la table de slot pour débuter la recherche de la taille de la table TDMA adaptée ;
- -T_UCOMM : Transformation des connections *Forward* et *Reverse* en communications unidirectionnelles
- -O_CLUST *fileName* : Sauvegarde les *clusters* d'IpPort ;
- -T_CARC : Crée un graphe orienté de la topologie spatio-temporelle, avec les arcs unidirectionnels correspondants aux liens entre les composants dans le réseau et les nœuds correspondants aux composants ;
- -T_PALLOC *schedulingStrategy* : Décide les chemins des communications dans la topologie spatio-temporelle ;
- -T_BUFF *Yes/No* : Dimensionne la profondeur des FIFOs dans les NIs. *Yes/No* pour activer/désactiver le contrôle de flux de bout-en-bout ;
- -O_USLOT *fileName* : Sauvegarde au format XML les bandes passantes et latences des communications unidirectionnelles en unité de slots ;
- -O_HW *fileName* : Sauvegarde au format XML la configuration matérielle du NoC ;

– -O_VHDL *fileName directory libraryName* : Génère le code VHDL RTL. Si *libraryName* = x alors le nom de la librairie utilisé est celui du NoC suivi de son numéro de version ;
– -help : Affiche l'aide de l'outil.

La commande -help affiche toutes les commandes et l'aide relative à chacune.
Il existe d'autres commandes et il est possible d'en rajouter de nouvelles.

3.4.7 L'interface graphique de µSpider

Affin de rendre plus facile l'utilisation de l'outil µSpider, une interface graphique a été développée. Celle-ci guide l'utilisateur dans le flot de conception en lui offrant une meilleure visibilité des actions possibles. Elle permet d'éviter des erreurs dans l'édition des commandes. De plus, une aide est disponible. Enfin, elle permet de sauvegarder des séquences de commandes pour pouvoir les rejouer.

La figure 3.18 montre une capture d'écran de l'interface graphique de µSpider.

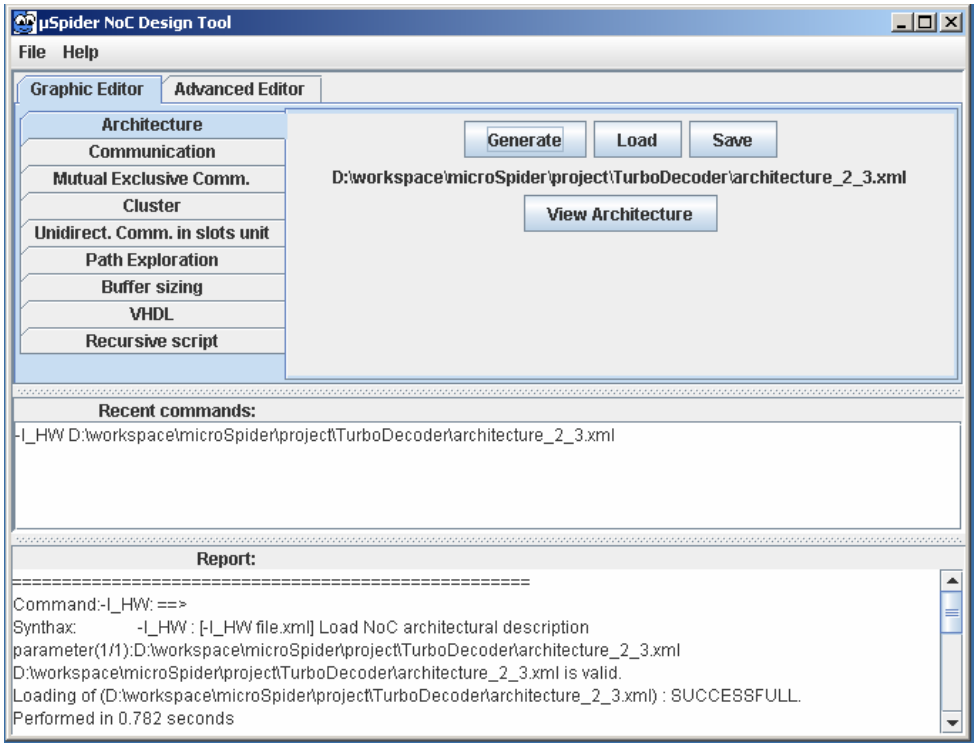


FIG. 3.18 – L'interface graphique de µSpider

3.4.8 µSpider en chiffres

A titre indicatif, le programme java de µSpider comporte environs 20.000 lignes de codes. Il génère du code XML et VHDL. Par exemple, la génération d'un simple NoC configuré avec un seul canal virtuel et composé d'un routeur de trois ports et de trois NIs, se traduit par la production de 9432 lignes de code VHDL.

3.4.9 Conclusion

Dans ce chapitre nous avons présenter notre outil. Il est capable de modéliser le NoC, de réaliser des décisions pour le paramétrer et de générer son code VHDL synthétisable. Cet outil est facilement évolutif. Il pourra également être utilisé pour générer du code systemC, ou encore réaliser des prédictions de surface et de consommation du NoC considéré.

Chapitre 4

Mise en œuvre de la qualité de service

4.1 Introduction

Ce chapitre détaille la mise en œuvre des étapes les plus complexes de notre flot. Il décrit comment dériver les données extraites de l'application, en données utilisables au niveau du NoC. Il présente notre technique d'exploration et d'affectation des chemins dédiés aux communications à garantir dans le NoC.

4.2 Préalable

Avec le principe du TDMA, un nombre de slots de temps doit être réservé pour satisfaire la bande passante de la charge utile de chaque communication.

Nous définissons les termes suivants :

L_p : Largeur d'un phit ou mot (en bits).
L_s : Largeur d'un slot de temps(en mots).
L_h : Largeur de l'entête d'un paquet (en mots).
F_{NoC} : Fréquence du NoC et de ses liens (en Hz).
B_l : Bande passante d'un lien (en bits/s)

$$B_l = F_{NoC} * L_p \tag{4.1}$$

$|S|$: Dimension de la table de slots (table TDMA) (en slots).
B_s : Bande passante associée à un slot réservé (en bits/s)

$$B_s = \frac{B_l}{|S|} \tag{4.2}$$

B_w : Bande passante associée à un mot réservé (en bits/s)

$$B_w = \frac{B_s}{L_s} \tag{4.3}$$

Un mot est utilisé soit pour transporter un entête soit pour transporter de la charge utile.

$B_{p,i}$: Bande passante requise par la charge utile pour la communication C_i (en bits/s).

H_i : Nombre d'entêtes dans une itération de la table de slots pour le ou les paquet(s) de la communication C_i (entêtes). Nous considérons qu'il n'y a généralement qu'un paquet et donc qu'un entête par itération de la table de slots pour chaque C_i. Ceci réduit la perte de bande passante.

$W_{h,i}$: Nombre de mots réservés dans une itération de la table de slots pour transporter les entêtes des paquets de la communication C_i (en mots).

$$W_{h,i} = L_h * H_i \tag{4.4}$$

$W_{p,i}$: Nombre de mots réservés dans une itération de la table de slots pour la charge utile de la communication C_i (en mots).

$$W_{p,i} = \left\lceil \frac{B_{p,i}}{B_w} \right\rceil$$

$$W_{p,i} = \left\lceil B_{p,i} * \frac{L_S}{(L_p * f_{NoC})} \right\rceil \tag{4.5}$$

S_i : Nombre de slots réservés dans une itération de la table de slots pour la communication C_i (en slots).

$$|S_i| = \left\lceil \frac{W_{p,i} + W_{h,i}}{L_s} \right\rceil$$

$$|S_i| = \left\lceil \frac{W_{p,i}}{f_{NoC}} * |S| + \frac{H_i * L_h}{L_s} \right\rceil \tag{4.6}$$

$W_{reel_{p,i}}$: Le nombre de mots obtenu par la réservation dans le TDMA pour pour la charge utile de la communication C_i (en mots).

$$W_{reel_{p,i}} = |S_i| * L_S - H_i * L_h \tag{4.7}$$

Du fait de l'arrondi supérieur pour déterminer un nombre entier de slots à réserver, nous avons :

$$W_{reel_{p,i}} \geq W_{p,i}$$

$B_{reel_{p,i}}$: La bande passante obtenue pour transporter la charge utile de la communication C_i (en bits/s).

$$B_{reel_{p,i}} = W_{reel_{p,i}} * B_w \tag{4.8}$$

La bande passante obtenue peut donc être supérieure à celle requise.

$$B_{reel_{p,i}} \geq B_{p,i} \tag{4.9}$$

4.3 Dérivation des contraintes applicatives en contraintes de communications

4.3.1 Introduction

Dans les outils traditionnels de conception de NoC, les contraintes de bande passante et de latence des communications sont généralement considérées comme connues et constituent le point d'entrée de leurs flots [45]. Or, la confrontation aux applications réelles nous montre qu'il s'avère très difficile de fournir de telles contraintes. La nécessité de les spécifier conduit souvent le concepteur à surestimer les exigences des communications, ce qui a pour conséquence un surdimensionnement de l'architecture mise en œuvre, ou bien au contraire, cela peut conduire à de mauvaises évaluations des besoins de communication et ainsi à des applications qui ne respectent pas leurs contraintes de fonctionnement.

Nous proposons une technique de dérivation des contraintes applicatives en contraintes de communications pour les NoCs. En effet, nous montrons que les contraintes des communications peuvent être extraites à partir des spécifications de l'application et de ses tâches.

4.3.2 Problématique

D'une part, les applications des domaines du traitement d'image et des télécommunications sont généralement spécifiées comme un ensemble de tâches avec des contraintes temporelles globales aux entrées et sorties.

D'autre part, il est nécessaire d'exprimer les contraintes de bande passante et de latence de chacune des communications afin que le NoC puisse les garantir. Ainsi, des contraintes locales relatives aux communications doivent être extraites des contraintes globales de l'application. Cependant, cette extraction n'est pas triviale car les décisions de conception sont fortement dépendantes. Pour résoudre le problème du dimensionnement de la taille de la table TDMA, il est nécessaire de connaître les contraintes de bande passante et de latence de chaque communication, cependant ces dernières dépendent de la taille de cette même table TDMA. Nous sommes face au problème du choix de l'ordre des décisions à prendre. Ce type de problème est classique dans le contexte des outils de CAO. Notre approche repose sur des hypothèses de départ qui visent à réduire le coût en FIFOs, car c'est un facteur de coût déterminant du NoC.

4.3.3 Principe

Les applications orientées flot de données peuvent être spécifiées par un graphe de tâches s'échangeant des données au travers du NoC. La figure 4.1 montre l'exemple simple d'une chaîne de quatre tâches (1-4) qui doivent être exécutées en moins d'une période pour respecter la contrainte de temps de l'application. T_i est une tâche de traitement. C_ij représente une communication entre deux tâches T_i et T_j.
En pratique, une application peut être spécifiée comme un ensemble de chaînes de tâches ayant des périodes spécifiques, mais pour plus de clarté, nous illustrerons ici le calcul des contraintes dans le cas d'une seule chaîne.

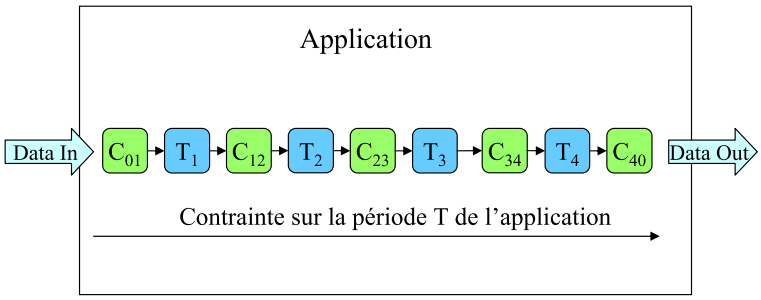


FIG. 4.1 – Une chaîne de tâches

4.3.3.1 Temps nécessaire pour qu'une quantité de données accède au réseau et le traverse

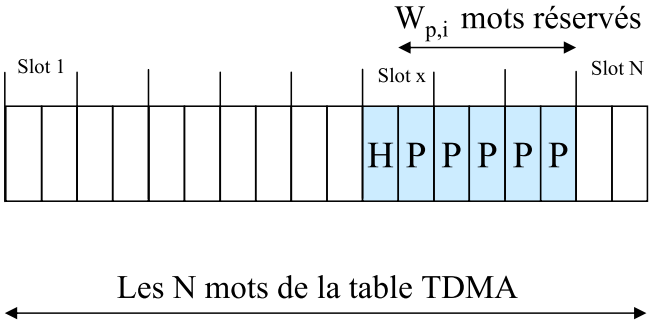


FIG. 4.2 – La réservation de slots dans une table TDMA

La figure 4.2, nous montre les slots réservés par une communication C_i dans une table TDMA. Six slots sont réservés. Les slots ont ici une capacité de deux mots. Ces réservation permettent d'émettre un entête (H) et 5 mots de charge utile (P).

N est le nombre de mots dans la table de TDMA. $(N = L_S * |S|)$;
n_i est le nombre de mots de données qui doivent être transmis pour la communication C_i.
s_i est le temps pour parcourir la longueur du chemin (en cycles) emprunté par C_i.

Nous pouvons déduire de la figure 4.2 le temps maximum nécessaire pour qu'une communication obtienne l'accès à un nombre de slots suffisant pour être entièrement expédiée dans le réseau.

$$delaiMaxEmission = \left\lceil \frac{n_i}{W_{reel_{p,i}}} \right\rceil (N - W_{reel_{p,i}})$$

À cela s'ajoute le temps pour acheminer ces données dans le NoC. Ce temps est fonction du nombre de mots de données à transmettre et de la longueur du chemin.

$$delaiDeTransport = n_i + s_i$$

Ainsi, le délai total maximum nécessaire pour acheminer les données est

$$FT_i(n_i, W_{reel_{p,i}}, s_i) = \left\lceil \frac{n_i}{W_{reel_{p,i}}} \right\rceil (N - W_{reel_{p,i}}) + n_i + s_i \tag{4.10}$$

Pour déterminer ce temps, il nous faut non seulement connaître la quantité de mots de données qui doit être transportée, mais aussi :

– la taille de la table TDMA,
– le nombre de slots réservés dans cette table pour la charge utile,
– la longueur du chemin.

Or nous ne possédons pas encore ces informations.

4.3.3.2 Algorithme

Afin de résoudre le problème des décisions croisées, nous avons opté pour une approche itérative favorisant la minimisation des FIFOs et une convergence au plus tôt. Notre approche consiste en les étapes suivantes (celles-ci seront détaillées ci-après) :

- ok = faux ;
- Tant que (!ok et !limite)
  - Affecter une portion de la bande passante au trafic BE en accord avec les choix du concepteur
  - Considérer la longueur du chemin le plus court de la source à la destination pour chaque C_i : s_{min_i} pour déterminer la latence L_i. C'est un choix optimiste, cette hypothèse n'est faite que pour la première itéréation.
  - Calculer la bande passante requise $B_{p,i}$ pour chaque communication C_i du trafic GT avec l'Éq. 4.16 ;
  - Calculer la taille de la table TDMA $|S|$ et le nombre de slots à réserver ($|S_i|$) pour chaque communication C_i (voir la section 4.3.6). On en déduit la bande passante réellement obtenue ($B_{reel_{p,i}}$) pour chacune des communications. Les bandes passantes obtenues sont égales ou supérieures à celles requises en raison de l'arrondi supérieur pour déterminer un nombre entier de slots ;
  - Vérifier le respect de la règle 1 (initialisation) ;
  - Vérifier le respect de la règle 2 (cadence) ;
  - Si succès alors ok = vrai
  - Sinon les contraintes de latence L_i sont mises à jour avec les valeurs courante S'_i sur la base de l'équation 4.16 dans laquelle $B_{min_{p,i}}$ est remplacée par la bande passante réellement obtenue $B_{reel_{p,i}}$.
- Fournir à l'étape d'allocation de chemin (routage spatio-temporel) la taille de la table TDMA et les contraintes de bande passante et de latence de transport (N,$W_{p,i}$,L_i).

La figure 4.3 montre l'évolution de la contrainte sur la longueur du chemin vis à vis de la bande passante durant le processus de dérivation. Nous débutons par l'hypothèse que les chemins obtenus seront de longueur minimale. Nous déterminons les bandes passantes requises. Le calcul de la table TDMA conduit à l'obtention de bandes passantes supérieures à celles requises, ce qui permet de relâcher la contrainte sur la longueur des chemins (nous pourrons alors nous satisfaire de chemins qui seront plus long que ceux minimums). Néanmoins, la latence d'accès dans le TDMA réduit quelque peu cette marge.

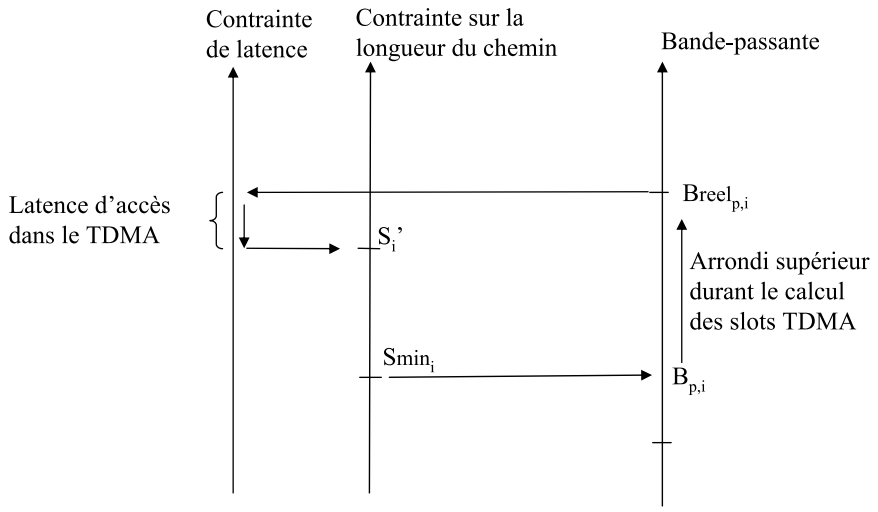


FIG. 4.3 – Contrainte sur la longueur du chemin et bande passante durant le processus de dérivation des contraintes

4.3.4 Les règles à respecter

Notre étude nous conduit à une formalisation du problème selon deux règles, la règle d'initialisation et la règle de cadence (contraintes en régime continu).

4.3.4.1 La règle d'initialisation

La règle d'initialisation traite du délai nécessaire pour lire la quantité minimale de données requises avant que la tâche T_i ne puisse commencer à s'exécuter. Cette initialisation est calculée en additionnant les délais d'initialisation (qui incluent traitement et communication) de toutes les tâches séquentielles depuis l'entrée jusqu'à la sortie de la chaîne.

La figure 4.4 nous montre une IP_i qui réalise une tâche T_i, et ses communications avec deux mémoires. La tâche T_i peut nécessiter 3 types de communications comme illustré.

- CFR_i est la communication relative à la requête de lecture faite par la tâche T_i ;
- CBR_i est la communication qui achemine les données que la tâche T_i consomme ;
- CFW_i est la communication pour écrire les données produites par T_i ;

Nous définissons les sigles suivants :

- T : la période de l'application ;
- nFR_i : le nombre de mots nécessaires à CFR_i ;
- nBR_i : le nombre de mots nécessaires à CBR_i ;
- nFW_i : le nombre de mots nécessaires à CFW_i ;
- P_i : le nombre d'itérations de T_i durant T ;
- I_i : le nombre de lectures avant que l'exécution de la tâche T_i ne puisse débuter ;
- ΔmemR, ΔmemW : temps accès à la mémoire en lecture et écriture respectivement ;

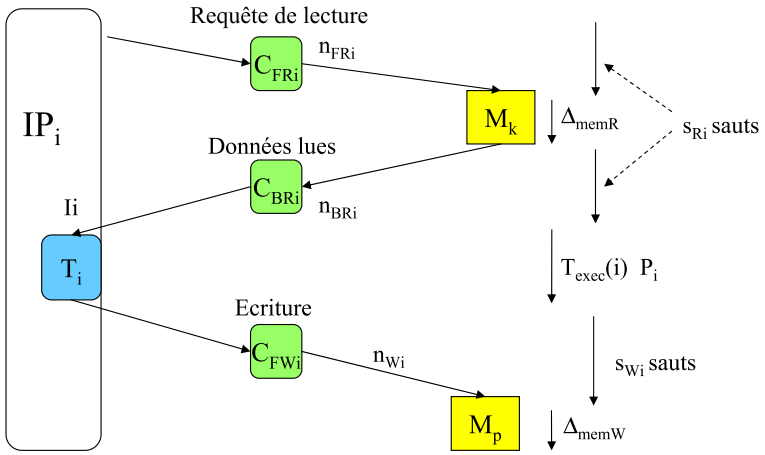


FIG. 4.4 – Chronogramme des communications autour d'une tâche

- sR_i : le nombre de sauts pour une opération de lecture ;
- sW_i : le nombre de sauts pour une opération d'écriture ;
- Texe(i) : Durée d'exécution de T_i pour une itération ;
- O_i : le nombre nécessaire d'itération de la tâche pour produire le nombre requis de mots pour permettre l'initialisation des tâches qui la suivent.

Pour chacune des tâches T_j consommant des données produites par la tâche T_i :

$$O_i = \left\lceil \frac{I_j nBR_j}{nFW_i} \right\rceil \tag{4.11}$$

Ainsi, pour une tâche T_i, nous pouvons définir son délai d'initialisation pour qu'elle puisse débuter. Nous noterons ce délai Tinit(i).

$$\begin{aligned} &T_{init}(i) = \\ &Max_{p \in Prev(i)}(T_{init}(p)) \\ &+F_T(nFR_i, K_{FRi}, sR_i) \\ &+\Delta_{memR} \\ &+F_T(I_i nBR_i, K_{BRi}, sR_i) \\ &+T_{exe} \\ &+\max\{Texe(i) - F_T(nFW_i, K_{FWi}, sW_i); 0\}(O_i - 1) \\ &+F_T(O_i nFW_i, K_{FWi}, sW_i) \\ &+\Delta memW(i) \end{aligned} \tag{4.12}$$

La règle 1 doit être vérifiée pour toutes les chaînes de l'application.

Règle 1

$$T_{init}(j) + F_T(nFW_j, (P_j - 1)nFW_j, SW_j) < T \tag{4.13}$$

Prev(i) est l'ensemble des tâches qui précèdent T_i. F_T est le délai de communication dans le NoC. F_T dépend de la taille de la table TDMA, de la longueur du chemin et du nombre de mots de données à transmettre. Il a été défini par l'équation 4.10.

La règle 1 exprime la dépendance entre les tâches. Elle se traduit sur les tâches de sortie de l'application par une contrainte de retard pour débuter qui est due à l'initialisation des tâches précédentes.

Ce modèle est générique et peut être appliqué à différents modèles de communication tels que les mémoires partagées ou les communications directes entre les registres des IPs. Dans ce dernier cas, nFR_i est nul.

La respect de la Règle 1 relative à l'initialisation est nécessaire, mais non suffisante pour garantir la contrainte de temps réel car la cadence doit être également vérifiée pour s'assurer que les données peuvent être produites dans les temps à chaque itération.

4.3.4.2 La règle de cadence

La vérification de la règle sur la cadence montre l'habileté du système à consommer et produire les données en accord avec la cadence de l'application pour toutes les communications.

Cela signifie que la bande passante qui permet à chaque tâche de se réaliser doit être en accord avec ses besoins de communication.

Nous faisons l'hypothèse que dans un flot de données régulier, la latence des requêtes est masquée puisqu'une requête de lecture (respectivement écriture) peut être émise avant que la précédente ait été traitée par celui qui l'a reçue.

Différents cas doivent être considérés, ils dépendent du modèle d'architecture de l'IP. Les communications transportant les lectures-écritures et l'exécution des tâches peuvent être réalisées séquentiellement ou simultanément. Pour des raisons de simplicité, nous présentons une seule règle, compatible avec le modèle complètement séquentiel. En pratique, une application pourra être composée de cas mixtes, certaines IPs pouvant fonctionner suivant un fonctionnement séquentiel et d'autres selon un fonctionnement parallèle.

Nous notons :

$$\begin{aligned} T_{read}(i) &= \Delta_{memR} + F_T(nBR_i, KBR_i, sR_i) \\ T_{write}(i) &= \Delta_{memw} + F_T(nFW_i, K_{FWi}, sW_i) \end{aligned}$$

Nous en tirons la règle suivante pour le cas séquentiel (pire cas).

Règle 2

$$T_{read}(i) + T_{write}(i) + T_{exec}(i) < \frac{T}{P_i} \tag{4.14}$$

La règle 2 exprime la contrainte de cadence individuellement pour chaque tâche.

4.3.5 Dimensionnement de la table TDMA

4.3.5.1 Présentation

La table TDMA est construite pour permettre une subdivision de la bande passante des liens sous forme de slots. Elle doit permettre de respecter les contraintes de latence (L_i en cycles) et de bande passante (BW_i en mots par cycle) des communications du type trafic GT. De plus, nous cherchons à minimiser sa taille. La taille de la table TDMA est exprimée en nombre de slots et notre flot a été conçu de sorte que nous devons la définir avant l'allocation des chemins.

4.3.5.2 Principe

La taille de la table TDMA doit être définie avant l'allocation des chemins et des slots aux communications du trafic GT. Sans la connaissance de la taille de cette table, les contraintes d'une communication C_i peuvent être spécifiées par les équations 4.15 et 4.16.

$$L_i + \frac{P_i n_i - 1}{B_{p,i}} < T - \delta_m P_i T_{exe}(i) \tag{4.15}$$

$$\Leftrightarrow \quad B_{p,i} > \frac{P_i n_i - 1}{T - \delta_m P_i T_{exe}(i) - L_i}$$

$$B_{min_{p,i}} \simeq \frac{P_i n_i - 1}{T - \delta_m P_i T_{exe}(i) - S_i} \tag{4.16}$$

n_i est le nombre de mots transmis durant chaque itération de C_i, δi vaut 1 (respectivement 0) si les traitements et communications sont réalisés séquentiellement (respectivement simultanément), S_i est le plus court chemin.

Les applications dans les domaines des télécommunications ou du traitement d'image sont généralement dominées par les communications et manipulent donc des communications périodiques ($P_i \gg 1$) et de grandes quantités de données ($n_i \gg 1$).

De plus, le temps d'attente pour accéder aux slots réservés dans la table de TDMA est masqué par les FIFOs de découplage. Cela signifie que L_i pourrait être négligé en comparaison du terme relatif à la bande passante, cependant par sécurité nous considérons $L_i = s_{min_i}$, qui est le délai d'acheminement de la communication C_i par le plus court chemin.

Ainsi, grâce à l'équation 4.16, nous pouvons obtenir une contrainte $BW_i min$ pour initialiser l'étape de conception du TDMA indépendamment de la valeur L_i qui est inconnue.

Cette hypothèse est renforcée par le fait que les valeurs de bandes passantes BW_i obtenues seront supérieures à celles demandées puisque les communications obtiendront un nombre entier (arrondi supérieur) de slots dans la table TDMA, pour satisfaire la bande passante requise.

4.3.6 Algorithme de calcul de la taille de la table TDMA

Comme mentionné précédemment, le dimensionnement de la table TDMA est une étape importante de notre méthode, son algorithme est basé sur un algorithme itératif rapide décrit ci-dessous :

- Notification des communications C_i ayant des exclusions mutuelles ;
- La taille de la table $|S|$ est initialisé à 0 ;
- Répéter
  - incrémentation de 1 de la taille de la table TDMA ; La recherche commence donc avec une latence minimum pour toute les communications ;
  - calcul du nombre de slots requis (et du nombre de mots $W_{p,i}$) pour satisfaire chaque communication C_i avec l'équation 4.6 ;
- Tant que (la somme des slots entrants/sortants dans une NI $>$ taille de la table TDMA & taille de la table TDMA $<$ taille maximale admissible). Fin du répéter ;

– La taille de la table est déterminée, les slots réservés pour contenir les $W_{p,i}$ mots pour chaque communications C_i sont calculés, ce qui permet le calcul de la longueur de chemin maximale permise S_imax avec la bande passante offerte (équation 4.16).

4.3.7 Validation

4.3.7.1 Cas d'étude et résultats

Une application de détection d'objets en mouvement est étudiée dans la section 7.3. C'est un exemple typique où les contraintes d'initialisation doivent être prises en compte. La plupart des tâches nécessitent de disposer d'une quantité minimum de données produites par les tâches qui les précèdent pour pouvoir commencer à s'exécuter pour la première fois. Par exemple, la tâche d'érosion requière 2 lignes (640 pixels) et 3 pixels de la ligne suivante avant de pouvoir réaliser sa première itération. Un autre point est la longue chaîne de dépendances entre les tâches, qui implique des calculs récursifs des délais d'initialisation qui s'imbriquent. Cette vérification repose principalement sur la règle 1.

4.3.7.2 Résultats

Grâce à notre approche, les contraintes des communications ont été extraites et une solution satisfaisant ces contraintes a été trouvée. Cette solution est détaillée dans la section 7.3. À partir des contraintes applicatives (cadence de 25 images par secondes) et du graphe des tâches de l'application ainsi que des caractéristiques des IPs (temps d'accès des mémoires), nous avons appliqué notre méthode pour extraire les contraintes de bande passante et de latence de chacune des communications et vérifié les règles d'initialisation et de cadence de l'application.

4.3.8 Conclusion

La conception d'un NoC optimisé requière la spécification des contraintes de bande passante et de latence pour chacune des communications de l'application. En pratique, ces informations sont généralement inconnues, car les communications appartiennent à un flot complexe avec de nombreuses dépendances et seules les contraintes applicatives de haut niveau sont généralement disponibles. Ce type de problème nécessite la définition de règles permettant de vérifier le respect des contraintes d'initialisation et de cadence de l'application.

Ces règles sont vérifiées durant le flot de conception afin de relâcher de façon appropriée les contraintes qui peuvent l'être.

L'utilisation de cette méthode sur une application de traitement d'image a permis de montrer son intérêt et de la valider.

A l'heure actuelle, cette partie du flot est encore partiellement manuelle. Du travail d'ingénierie reste nécessaire pour l'inclure dans notre outil, cependant, la méthode a été validée.

4.4 Routage spatio-temporel

4.4.1 L'allocation des chemins dans l'espace et le temps

4.4.1.1 Introduction

Pour chacune des communications du trafic GT qui doivent traverser le NoC selon le principe de TDMA, nous désirons trouver un chemin et un instant d'envoi qui permettent de satisfaire les contraintes de latence et de bande passante.

L'échec dans l'allocation d'une communication est attribué soit à une allocation sous-optimale des précédentes communications, soit à une insuffisance des ressources mises à disposition. Les ressources en l'espèce sont les liens et les slots de temps de la table TDMA.

Or quand aucun chemin n'a été trouvé, il est nécessaire d'augmenter la taille du TDMA de façon à créer de nouvelles opportunités sur la dimension temporelle. Cependant, le redimensionnement du TDMA peut se traduire par une augmentation importante de celui-ci pour satisfaire une distribution de bande passante correcte, car le nombre de slots alloués est un nombre entier. Enfin, l'augmentation de la taille de la table TDMA implique une augmentation de la taille des FIFOs et donc un coût supplémentaire. Ainsi, il est crucial de disposer d'un algorithme d'allocation des chemins qui soit performant afin de trouver une solution avec une taille de table TDMA minimale. Les auteurs dans [45] présentent une approche pour allouer le placement des IPs et les chemins des communications. Leur approche favorise l'utilisation des liens ayant le moins de slots réservés, notamment, les liens qui à la fois réduisent à minima la fenêtre temporelle relative à la communication à placer et offre une courte longueur de chemin. C'est une approche gloutonne. Dans celle-ci, contrairement à la notre, il n'y a pas de pré-réservation pour guider vers des choix qui faciliteront la suite de la résolution du problème. Notre technique utilisant les pré-réservations est expliquée dans les sections suivantes.

4.4.1.2 Topologie avec une exploration sur les dimensions espace et temps

Nous utilisons une technique TDMA. Nous avons vu que dans ce cas le trafic GT est pré-ordonnancé grâce à ce TDMA pour éviter la moindre contention entre ses paquets. Grâce à cette absence de contention du trafic GT, aucun *deadlock* ne peut intervenir et donc aucune règle particulière de routage (technique de routage sans interblocage ou *deadlock-free*) n'est nécessaire pour les éviter, contrairement au trafic BE. Cependant, la technique TDMA ajoute une dimension temporelle en plus des classiques dimensions spatiales. Celle-ci doit être considérée lors de l'allocation des chemins. Ainsi, nous passons par exemple d'un espace 2D (X,Y) à une représentation espace-temps (X,Y,t). La figure 4.5 montre une simple topologie en grille 2D avec ses liens entre les routeurs et la version duale de cette représentation avec l'ajout de la dimension temporelle en unité de slots. Nous appellerons ce dernier, le graphe spatio-temporel. La taille de la table TDMA est ici de 3 slots. Un arc de la vue spatio-temporelle correspond à un lien unidirectionnel de la vue topologique durant un slot de temps donné. Les arcs sont orientés en accord avec l'évolution séquentielle de la table TDMA, du premier au dernier slot, modulo la taille de la table.

Pour des raisons de visibilité et compréhension, les arcs fermant la boucle en allant du dernier au premier slot ne sont pas tous représentés sur la figure 4.5. De plus, seuls les arcs correspondant aux flèches en gras de la vue topologique sont représentés sur le graphe spatio-temporel. Le trafic GT suit la règle de transport suivante :
Il n'y a pas d'arcs verticaux car un paquet de type GT ne peut jamais attendre dans un routeur (contrairement au BE), un paquet GT effectue un saut dans une dimension spatiale à chaque nouveau slot de la dimension temporelle. De plus, un paquet ne peut pas ressortir d'un routeur par le lien par lequel il est entré.

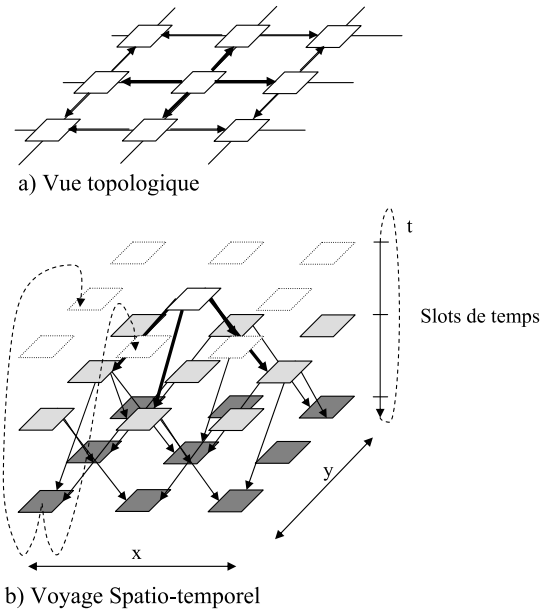


FIG. 4.5 – Topologie spatiale et vue spatio-temporelle

4.4.1.3 Technique d'allocation des chemins

Dans le graphe spatio-temporel, nous parlons de chemins spatio-temporels. Ils disposent d'un chemin spatial, d'un instant de départ et d'une durée. À partir de maintenant le terme chemin désignera le chemin spatio-temporel. Notre technique d'allocation des chemins manipule des communications ayant des contraintes de bande passante et de latence. La latence est exprimée en nombre de sauts et la bande passante en nombre de slots réservés. De plus, les cliques des communications mutuellement exclusives sont considérées et permettent la multi-réservation des slots de temps par ces communications. L'objectif est de trouver une solution permettant de satisfaire toutes les communications GT spécifiées et de réduire le coût en FIFOs. Pour réduire le coût en FIFOs, nous favorisons les plus courts chemins tout en respectant les contraintes.

La base de notre technique est la réservation des chemins et donc des arcs du graphe spatio-temporel en fonction des poids de pré-réservation faites par les communications non encore placées. L'objectif est de faire dans la mesure du possible des choix qui restreignent le moins possible les solutions pour les choix qui restent à faire. Notre algorithme est organisé en cinq étapes principales.

L'algorithme est le suivant :

Tant que les communications n'ont pas toutes un chemin réservé (elles ne sont pas satisfaites)

1. Extraction des chemins candidats pour chacune des communications non satisfaites.
2. Pré-réservation des arcs sur les chemins candidats pour chacune des communications non encore satisfaites.
3. Sélection de la communication Ci à satisfaire parmi toutes les communications non satisfaites.
4. Sélection d'un chemin P_i pour la communication C_i parmi les chemins candidats.
5. Réservation des arcs du chemin P_i par la communication C_i. C_i est marquée comme satisfaite.
6. Annulation de toutes les pré-réservations devenues obsolètes.

Chacune de ces étapes est détaillée dans les sections suivantes.

4.4.1.4 Vue détaillée de l'algorithme de décision des chemins

L'extraction des chemins candidats : Nous avons adapté l'algorithme de Dijkstra pour déterminer les chemins minimums dans notre graphe spatio-temporel. Pour chaque communication non encore placée et pour chaque slot de départ possible dans la table TDMA, le graphe spatio-temporel est exploré par une version modifiée de l'algorithme de Dijkstra. Les arcs ont un poids de 1. Un arc est disponible s'il n'est pas encore réservé ou s'il n'est réservé que par des communications appartenant toutes à des cliques de mutuelle exclusion dont fait également partie la communication considérée. De plus, le nombre de slots libres suivants doit être suffisant sur un lien pour permettre à toute la taille du paquet de passer sans être scindé. Enfin, il ne retient pas un chemin minimal, mais tous les chemins minimaux entre une source et une destination.

Pour chacun des slots de temps évalués comme instant de départ à la source, les distances les plus courtes de la source à la destination qui ont été trouvées peuvent être différentes. Pour la communication considérée, si la plus courte distance trouvée de sa source à sa destination respecte sa contrainte de latence, alors les plus courts chemins minimaux sont extraits de ce graphe afin d'être ajoutés à la liste des chemins candidats de cette communication. Notons que nous éliminons les chemins qui sortent d'un routeur par le même port par lequel ils sont rentrés car nous avons fait le choix de ne pas traiter ce cas rare pour simplifier l'architecture des routeurs.

La phase de pré-réservation : Cette étape a pour objectif de quantifier la probabilité d'utilisation de chacun des slots des arcs (c'est à dire de chacun des slots temporels de chacun des liens unidirectionnels du NoC) par les communications non encore placées. Le poids d'une pré-réservation est de (1 / nombre de chemins candidats). Une pré-reservation est faite par une communication pour chacun des slots de temps que son paquet peut occuper sur chacun des arcs de chacun de ses chemins candidats. Sur la figure 4.6, nous observons les poids des pré-réservations faites pour un paquet d'une communication qui nécessite deux slots. Ce paquet peut s'accommoder de trois positions différentes. Nous voyons que les slots libres au centre se voient affectés un poids de préservation plus important que ceux situés sur les extrémités.

En effet, ceux du centre offrent plus de mobilité au paquet de deux slots que l'on désire placer. Si une autre communication venait à se réserver les deux slots du centre, elle priverait cette communication de toute possibilité d'utiliser ce lien. Il est donc tout a fait judicieux de faire cette pondération des pré-réservations. Elle signalera aux autres communications qui seront placées avant celle-ci d'utiliser les slots les moins pré-réservés dans la mesure du possible.

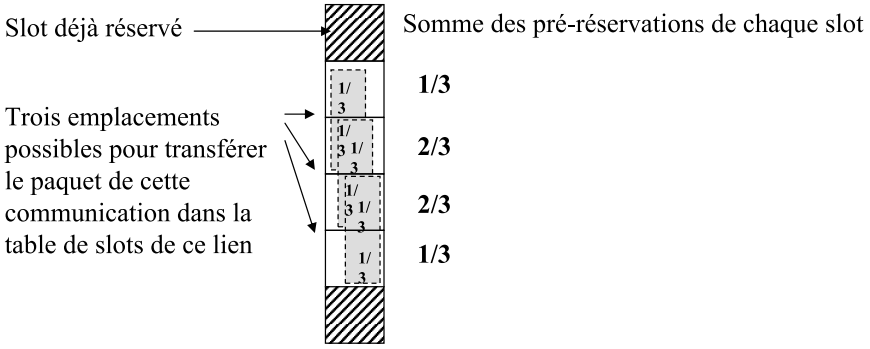


FIG. 4.6 – Répartition des Pré-réservations des slots

Pour chaque communication non allouée, une pré-réservation est faite pour tous les arcs, de tous ses chemins candidats, en accord avec la taille du paquet. Les pré-réservations s'additionnent lorsqu'elles sont effectuées sur un même arc.

La sélection de la communication Ci à satisfaire : Parmi toutes les communications pour lesquelles aucun chemin n'a encore été réservé, il nous faut décider laquelle il est le plus judicieux de satisfaire en première. Ce choix doit être fait de façon judicieuse. En effet, la réservation des arcs appartenant à son chemin spatio-temporel réduira la liberté d'exploration pour l'affectation des chemins des communications qui restent à placer.

Pour cette politique de sélection de la communication à placer en priorité, nous avions étudié un facteur que nous qualifions de « gène » ou de contention. Nous avions initialement imaginé deux stratégies très opposées qui sont les suivantes :

- Satisfaire en premier les communications qui ont un chemin candidat qui rencontre le moins de pré-réservations des autres communications (gène le moins les autres)
- Placer en premier celui qui rencontre le plus de conflits avec les autres, ainsi, pour lui sa réservation est faite dès le début plutôt que d'attendre que les possibilités ne se réduisent encore.

Ces solutions n'ont pas montré de résultats significatifs.

Nous avons décidé d'ordonner les communications selon le critère de priorité suivant :

$$Prior_{C_i} = \frac{Bandepassanterequise_{C_i}}{latencecourante_{C_i} - latencerequise_{C_i}}$$

Ainsi, la priorité est donnée à la communication nécessitant la plus grosse bande passante et ayant la laxité la plus faible (la plus petite marge pour satisfaire sa contrainte de latence).

La sélection d'un chemin : Une fois que l'on a sélectionné la communication à satisfaire, il reste à choisir parmi les chemins candidats retenus, le chemin P_i qui va finalement être celui qui sera alloué (réservé) à la communication C_i. Pour cela nous pouvons utiliser un facteur de coût qui pourra notamment être la gène qu'entraînerait l'utilisation d'un chemin plutôt qu'un autre pour l'allocation des communications qui restent à placer. Cette gène est ici représentée par les pré-réservations faites par ces dernières.

Différentes fonctions de coût ont été évaluées :

1. *Le choix naïf :*
   Le choix du chemin parmi ceux candidats est réalisé par un tirage aléatoire.
2. *Le plus court chemin :*
   La pré-réservation n'est pas utilisée, le plus court chemin est sélectionné. C'est en quelque sorte un choix égoïste, sans concertation. Le facteur de coût est la taille du chemin.
3. *Le chemin ayant la plus faible « somme des pré-réservations sur ses arcs » :*
   Le facteur de coût est la somme des pré-réservations sur chacun des arcs du chemin. Les pré-réservations faites par C_i, ainsi que les communications appartenant à l'une de ses cliques d'exclusion mutuelle, ne sont pas considérées car elles ne représentent pas un conflit. Notons ici que la pré-réservation d'un chemin par une communication C_j en conflit avec C_i n'est comptabilisée qu'une seule fois. En effet, que l'utilisation du chemin P_i condamne une ou plusieurs fois l'un des chemins de la communication C_j, la gène entraînée est la même.
4. *Le chemin ayant la plus faible « pré-réservation maximale » :*
   Le facteur de coût est la pré-réservation maximale faite sur l'un des arcs du chemin. Pour chaque chemin candidat, nous faisons la somme des pré-réservations sur chacun de ses arcs, puis nous retenons la plus grande valeur. Cette valeur est la somme des pré-réservations sur l'arc le plus pré-réservé de ce chemin. Le chemin sélectionné sera celui qui a la plus faible « plus grande pré-réservation ».

Beaucoup d'autres fonctions de coût peuvent être imaginées, mais nous nous limiterons à celles-ci dans le cas de cette étude, car elles se sont montrées les plus représentatives.

Quand des chemins candidats ont un facteur de coût identique, des critères supplémentaires peuvent être utilisés pour les départager, notamment, le nombre d'arcs déjà réservés par des communications appartenant à une clique d'exclusion mutuelle avec la communication C_i. Ceci permet de favoriser la multi-réservation des arcs par les communications appartenant à une même clique et donc de laisser plus d'arcs libres aux autres communications n'appartenant pas à cette clique.

La phase de réservation : Le chemin sélectionné est alloué à la communication sélectionnée. Les arcs (slots temporels des liens unidirectionnels du NoC) sont réservés en accord avec l'instant de départ, le chemin et la taille du paquet. Le chemin est marqué comme ayant été satisfait.

L'annulation de toutes les pré-réservations devenues obsolètes : Enfin, toutes les pré-réservations de toutes les communications sont effacées car certaines des réservations faites par la communication précédemment placée rendent certains chemins indisponibles.

4.4.2 Comparaisons des différents facteurs de coût de l'étape de sélection du chemin

Les techniques évaluées sont :

1. le choix naif ;
2. le plus court chemin ;
3. le chemin ayant la plus faible « somme des pré-réservations sur ses arcs » ;
4. le chemin ayant la plus faible « pré-réservation maximale ».

Toutes ces techniques sont des heuristiques.

4.4.2.1 Évaluation sur un exemple simple

La figure 4.7 montre un exemple d'une topologie en grille 4x4. Huit communications (de C0 à C8) doivent être satisfaites. Nous pouvons voire que la communication C0 est celle qui dispose du plus grand nombre de chemins minimum. Si, par l'étape de sélection de la communication à placer, C0 est la première communication pour laquelle un chemin va être réservé, alors les techniques présentés ci-dessus peuvent obtenir des résultats différents. En effet les expériences nous montre que les technique 3 et 4 trouvent le chemin "idéal pour C0" (celui indiqué sur la figure) , ce qui permet de satisfaire toutes les communications par les chemins les plus courts.

Les deux premières techniques conduisent soit à des chemins plus longs, soit à des échecs pour satisfaire toutes les communications ce qui conduit à augmenter la taille de la table TDMA pour disposer d'un graphe d'exploration plus large sur la dimension temporelle afin d'aboutir à un succès.

Les techniques 3 et 4 sont donc meilleures.

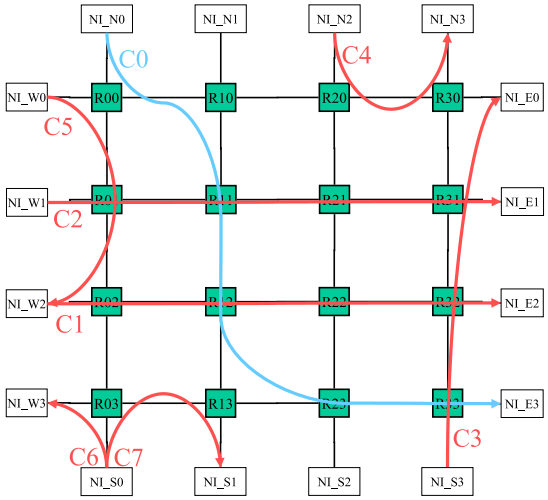


FIG. 4.7 – Répartition des communications sur une topologie

4.4.2.2 Étude plus large

Pour les différentes techniques de sélection des chemins présentées dans la section précédente, la figure 4.8 montre la longueur moyenne des chemins (en nombre de sauts)

trouvé pour 3 applications différentes. Afin de pouvoir comparer ces techniques, elles doivent trouver la solution avec la taille de la table TDMA minimum.

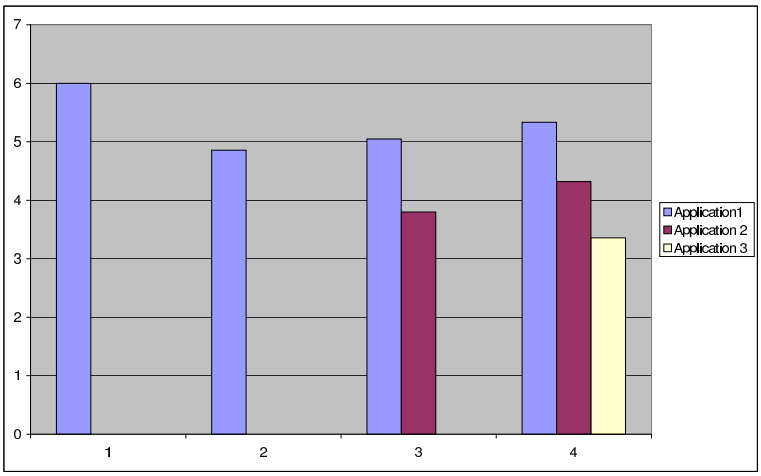


FIG. 4.8 – Longueur moyenne des chemins en fonction de la technique d'affectation des chemins

Le choix naif sélectionne arbitrairement un chemin qui satisfait les contraintes de la communication placée. Ce choix qui est non optimum pour la communication placée et pour la liberté qu'il laisse aux communications qui restent à placer, conduit à des chemins globalement plus longs que ceux obtenus par les autres techniques. De plus, il échoue pour trouver une solution pour les applications 2 et 3. Le facteur de coût 2 ne nécessite pas l'étape de pré-réservation et peut rapidement produire un résultat. S'il trouve une solution, il obtient paradoxalement parfois la solution qui offre les plus courts chemins et réduit le plus la taille des FIFOs. En effet chaque communication fait le choix égoïste de prendre le plus court chemin et si finalement tout le monde trouve une solution, cette technique est suffisante. Si ce facteur de coût ne permet pas de satisfaire les contraintes, les facteurs de coût 3 et 4 sont intéressant à utiliser car ils trouvent plus facilement une solution, mais celle-ci est moins optimisée, car chaque communication lors de son placement prend soin de ne pas déranger les communications restantes à placer et se contentait de satisfaire sa contrainte. C'est finalement le facteur de coût 4 qui se révèle trouver le plus souvent une solution respectant les contraintes. Il est ainsi plus préjudiciable d'utiliser un arc qui était fortement pré-réservé par une communication, que d'utiliser un chemin qui occupe des arcs sur lesquels plusieurs communications ont chacune fait une faible pré-réservation.

4.4.2.3 Conclusion

Pour obtenir un résultat rapide, nous préconisons d'utiliser le facteur de coût "du plus court chemin" la première fois, pour la taille de la table minimale qui satisfait les bandes passantes. Si cette technique échoue à satisfaire toutes les communications, alors ensuite nous utilisons le facteur de sélection du chemin ayant la plus faible « pré-réservation maximale », ceci jusqu'à ce que toutes les communications soient satisfaites ou que l'on dépasse la taille maximale admissible pour la taille de la table TDMA.

4.4.3 Complexité de l'algorithme et heuristiques possibles

4.4.3.1 Complexité

Nous considérons les grandeurs suivantes :
C : le nombre de communications
R : le nombre de routeurs
S : le nombre de slots de la table TDMA (c'est aussi la taille maximale d'un paquet en nombre de slots)
A : le nombre d'arcs partant d'un routeur
Cl : le nombre de cliques auxquelles peut appartenir une communication

Ainsi, le nombre de nœuds du graphe spatio-temporel est $R * S$

La complexité de l'algorithme Dijkstra modifié est

$O(S * R * A * S * C * Cl)$

La complexité de l'algorithme de routage est

$O(C^2 * S*$(complexité de l'algorithme Dijkstra modifié)) $= O(C^3 * S^2 * R * A * Cl)$

Le nombre d'arcs partant d'un routeur est faible, il vaut 5 au maximum dans une grille 2D (liens nord, sud, est, ouest et local).

En pratique, le nombre S se situe entre 4 et 128. En effet, pour que les communications aient un temps d'attente court pour accéder aux slots qui leurs sont réservés, il faut que la table de TDMA ait un temps de rotation suffisamment court et donc une taille raisonnable.

Ce calcul de la complexité donne une estimation pessimiste. En effet, nous ne pouvons pas avoir à la fois beaucoup de paquets dans une table de slots et ces paquets occupant tous les slots de la table. De plus, l'algorithme dispose de conditions d'échappement dans ses boucles, afin de ne pas poursuivre leurs exécutions lorsque cela est rendu inutile. Enfin, au fur et à mesure de l'exécution, le nombre de communications non allouées décroît, accélérant ainsi la vitesse d'exécution.

A titre indicatif, nous avons exécuté notre algorithme de recherche de chemin sur une machine équipée d'un processeur Intel Pentium III à 3GHz et de 1 Giga de RAM. Il trouve généralement la solution en quelques minutes, voir en quelques secondes. Dans un cas complexe tel que l'application du cas 6 de la section 7.6, qui utilise une topologie en grille 5x5 pour connecter 25 NIs, utilise les exclusions mutuelles, ainsi que le contrôle de flux de bout-en-bout (il implique ici 96 communications unidirectionnelles), une solution avec une table de 16 slots a été trouvée en un temps d'exécution de 8643 secondes soit 2h24. Ceci s'explique par la complexité de ce cas.

Cet algorithme est codé en java, langage qui ne tire pas partie de toute la vitesse de la machine. Notre algorithme n'a pas été prévu initialement pour être embarqué et exécuté pour trouver de nouvelles solutions durant le fonctionnement du système. Ce temps n'est donc pas préjudiciable.

4.4.3.2 Améliorations et heuristiques

Dans le cadre d'un système reconfigurable ou adaptatif, il peut s'avérer nécessaire d'exécuter en ligne l'algorithme de recherche de chemin pour qu'il trouve une nouvelle solution. Dans ce cas cette exécution doit être rapide.

Nous pouvons noter qu'il est possible de réduire le temps d'exécution de notre algorithme en faisant appel à quelques améliorations et heuristiques :

- Le nombre de chemins candidats retenu peut être limité. Les N premiers sont retenus, car en effet des centaines de chemins peuvent être trouvés avec la version actuelle.
- L'algorithme peut ne recalculer que les chemins candidats des communications dont tous les chemins candidats pré-réservés ont perdu un arc par la réservation du chemin faite par la communication qui vient d'être satisfaite.

Ces améliorations n'ont pas été introduites, mais s'intègrent dans des travaux en cours traitant notamment de la surveillance et du contrôle du NoC.

Enfin, seule l'allocation du trafic GT a été détaillée ici, puisque seul celui-ci doit bénéficier de garantie et fonctionner sur le principe du TDMA. Précisons que le trafic BE qui n'a pas été abordé ici, peut se voir affecter les chemins utilisant les liens qui ont été les plus délaissés par les réservations des communications GT, ceci afin, de souffrir au minimum des conflits avec le trafic GT qui a une priorité plus élevée. L'auteur dans [72] présente une approche dans ce sens. Pour le trafic BE, il faut en revanche veiller à utiliser des chemins qui empêcheront l'apparition d'un *deadlock*.

4.4.4 Conclusion

Nous avons présenté une technique de recherche de chemin pour les communications de type trafic garanti dans le contexte d'utilisation de tables de TDMA.

L'algorithme que nous proposons tire partie de la connaissance des potentielles exclusions mutuelles entre les communications. De plus, une technique de pré-réservation des chemins permet à l'algorithme d'obtenir des solutions avec des topologies et des tables de TDMA les plus petites possibles, ceci afin de réduire le coût en FIFO et donc en surface du NoC.

De plus, cet algorithme a été programmé en java et validé sur des applications.

Chapitre 5

Technique de codage des chemins pour la sécurité et la reconfiguration

5.1 Introduction à la sécurité

Un SoC est constitué d'un ensemble d'éléments IP, tels que, processeurs, mémoires et matériels dédiés. Ainsi, le SoC sera amené à faire cohabiter un ensemble d'applications dont certaines pourront avoir des exigences en termes de sécurité.

Les NoCs fournissent aux concepteurs de SoCs un moyen flexible pour contrôler les communications entre un grand nombre de blocs IP, ainsi que des capacités de reconfiguration. Cependant, cette flexibilité introduit de nouveaux points faibles dans le système et offre des opportunités à de potentielles attaques.

De plus, la complexité des applications, l'hétérogénéité de l'architecture et les besoins en reconfiguration, qui peuvent justifier l'utilisation d'un NoC au sein d'un SoC augmentent la gravité de ces attaques potentielles. Il est donc nécessaire de prendre en compte la sécurité du NoC. Ce sujet a jusqu'à présent été négligé, mais deviendra certainement une préoccupation importante dans l'avenir compte tenu de l'importance grandissante que prend la sécurité dans les systèmes embarqués.

Enfin, nous verrons que la technique de codage que nous proposons offre des propriétés intéressantes pour d'autres aspects que la sécurité, notamment, la reconfiguration [73].

5.2 Analyse des attaques sur un NoC et points faibles actuels

Nous dissocions la sécurité et la fiabilité. Nous nous attachons ici plus particulièrement à l'aspect sécurité même si les solutions présentées pourrait être utilisée pour détecter un comportement anormal. La tolérance aux fautes dans les NoCs est traitée dans [74]. Une solution de sécurité des échanges à base de « clé privée »/« clé publique » est proposée par les auteurs de [75].

Nous définissons ici un élément qui contrôle le NoC, le CCM (pour *central configuration module*). Il est unique dans le réseau et est en charge de l'initialisation, la configuration et la reconfiguration du NoC. Le NoC de Philips, Æthereal, utilise un module similaire [76] dans le but de configurer les tables TDMA des NIs uniquement. Des fonctionnalités telles que la supervision et la défense peuvent être ajoutées au CCM, dans une optique de sécurité. Ces points seront discutés dans les sections suivantes.

5.2.1 La sécurité

Il peut être nécessaire de faire cohabiter des données sécurisées avec des composants IP et des interfaces non sécurisées. Par exemple, la lecture dans une mémoire peut être autorisée à toutes les IPs, mais l'opération d'écriture peut être restreinte à une IP donnée.

Nous définissons deux zones dans un système, la zone sécurisée ou sûre et la zone non sécurisée ou non sûre. La zone sécurisée stocke, calcule ou transporte des informations critiques. La zone non sécurisée est relativement ouverte et vulnérable. Typiquement, la zone non sûre peut être un FPGA qui est facilement reprogrammable, alors que la zone sécurisée peut être un ASIC (figure 5.1). En pratique, une solution entièrement basée sur un ASIC ne peut pas toujours être sélectionnée. Pour des raisons de flexibilité, de consommation d'énergie et de performances, la reconfiguration devient un point clé pour les applications futures telle que la radio logiciel par exemple [77]. Dans ce type de domaine, les zones sécurisées et non sécurisées peuvent être associées aux zones noire et rouge respectivement.

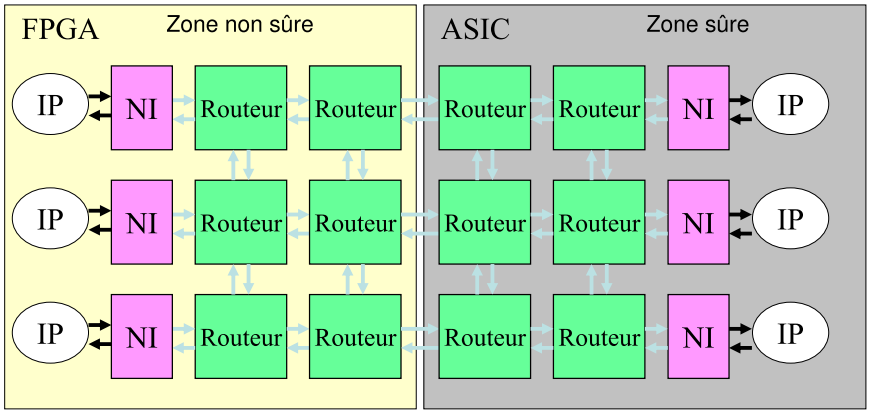


FIG. 5.1 – Un réseau distribué sur un ASIC et un FPGA

Nous pouvons ainsi distinguer trois types d'intégrations du réseau.

– Intégration entièrement sur un ASIC. Le NoC bénéficie de la protection intrinsèque de la puce. Les potentiels points faibles sont les interfaces du système, notamment les accès de lecture et d'écriture depuis ces interfaces. Le NoC est un moyen de contrôler les entrées/sorties, mais aussi d'étendre la vulnérabilité du SoC.
– Intégration entièrement sur un FPGA. En plus des aspects précédents, la capacité de reconfiguration du FPGA offre de nouveaux points faibles. Cependant,

deux sous-classes peuvent être distinguées puisque l'encryptage du *bitstream* peut être partiellement ou pleinement utilisé.
– Intégration partielle sur ASIC et FPGA. Il faut être capable de contrôler les accès entre l'ASIC et le FPGA

5.2.2 Les scénarios d'attaque

L'agresseur peut avoir différents objectifs et donc différents scénarios. Différents types d'attaques peuvent être identifiés et sont classés dans la table 5.1. Ces types d'attaques sont expliqués ci-après.

Attaque	Type d'attaque	Scénario d'attaque
Distante	Déni de service	Sur-utilisation du réseau
		Chemin incorrect
		Deadlock
		Livelock
	Extraction d'informations secrètes	Lecture non autorisée
	Altération du comportement	Écriture non autorisée ou reconfiguration
De proximité	Rétro-ingénierie	Extraction d'informations de conception
	Extraction d'information secrètes	Observation en fonctionnement

TAB. 5.1 – Types et scénarios d'attaques

5.2.2.1 Le déni de service

Ce type d'attaque a pour but d'altérer les performances du système. Une sur-utilisation du réseau peut dégrader la disponibilité et les performances du système. Des requêtes fréquentes peuvent être utilisées pour gaspiller de la bande passante et causer une augmentation de la latence des transferts dans le système, ce qui peut conduire à dépasser les délais admissibles.

Les trois scénarios d'attaque suivants sont plus pénalisants car ils ont pour but d'obstruer les canaux du NoC. Ils consistent à introduire dans le réseau un paquet dont les instructions de chemin sont incorrectes.

– Chemin incorrect
  Le paquet est piégé dans un chemin qui ne débouche pas, ceci dans le but de le piéger dans le réseau. Le corps de ce paquet occupe ainsi quelques canaux et les rend indisponibles pour les autres paquets.
– *Deadlock*
  Le paquet introduit ne respecte pas les règles qui garantissent l'absence de *deadlock*, ceci dans le but de créer un interblocage. Ceci conduit à la contention des canaux et par conséquent, condamne une partie, voir tout le réseau.
– *Livelock*
  Le paquet introduit n'atteint aucune destination et reste tourner indéfiniment dans le réseau, causant un gaspillage de bande passante, latence et consommation d'énergie.

5.2.2.2 L'extraction d'informations secrètes

Elle a pour but de lire des informations protégées dans une cible sécurisée et non autorisée. Les information dérobées peuvent être des données sensibles, des instructions d'un programme critique, des registres de configuration et bien d'autres.

5.2.2.3 L'altération du comportement

Elle consiste en une écriture dans le but de modifier le comportement ou la configuration du système. Cette pratique est appelée *Hijacking*. Elle peut avoir des fins malveillantes, tel que l'espionnage ou la détérioration du système.

5.2.2.4 La rétro-ingénierie et l'extraction d'informations secrètes par accès de proximité

La rétro-ingénierie (ou *Reverse engineering*) est l'activité qui consiste à étudier un système pour en déterminer le fonctionnement. Par un accès physique au circuit, l'attaquant peut tenter de dérober des informations de propriété intellectuelle par des lectures non autorisées, pour obtenir des parties du *firmware*. L'extraction d'informations secrètes par accès de proximité peut aussi être réalisée par des analyses différentielles de puissance (DPA pour *Differential power analysis*) dans le cas d'extraction de clé cryptée [78].

5.3 Les stratégies de protection

Les techniques classiques de codage et d'authentification ont un coût en surface, temps et consommation d'énergie qui ne sont pas compatibles avec le domaine des NoCs. Nous proposons ici des techniques originales.

5.3.1 Garantir le trafic contre le déni de service

Une simple solution contre l'attaque de déni de bande passante consiste à utiliser des canaux virtuels (VCs)[20] pour séparer les communications sûres et non-sûres. Les canaux virtuels permettent de multiplexer plusieurs trafics sur un même canal physique. Chaque canal virtuel peut se voir affecter un niveau de priorité, ce qui permet d'avoir plusieurs classes de priorités de trafics.

Nous pouvons utiliser 2 canaux virtuels, un premier de haut niveau de priorité pour assurer les échanges sécurisés dans la zone sûre et un second canal virtuel de faible niveau de priorité et de classe de trafic de faible sécurité. La zone sécurisée est donc traversée par deux types de canaux virtuels et donne la priorité aux paquets évoluant sur le canal virtuel prioritaire (figure 5.2).

Si aucune technique TDMA (*Time division multiplexing access*) n'est disponible pour allouer des slots au trafic non sûre alors une intégration simple consiste à n'assigner que le VC de faible priorité à la zone non sûre. Les communications entre les deux zones n'utilisent alors que le VC de faible priorité. Ainsi, les communications sécurisées de la zone sûre sont isolées des potentielles attaques de déni de service.

Cependant cette solution ne protège que du déni de bande passante et ne traite pas les droits d'accès.

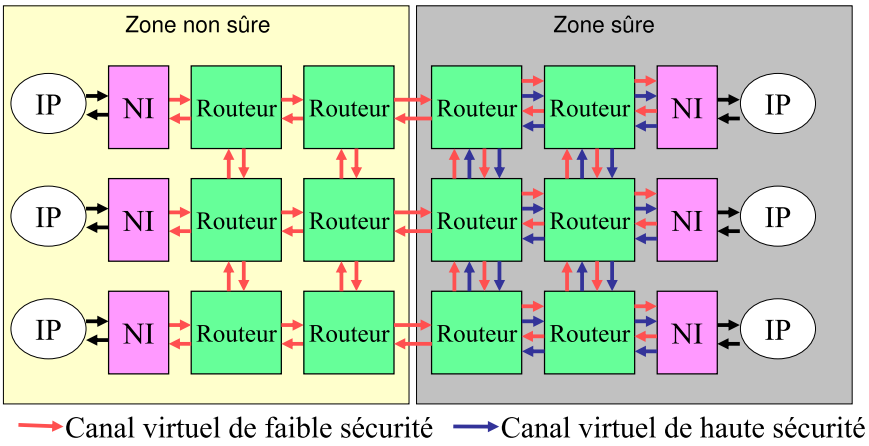


FIG. 5.2 – Séparation des trafics sur des canaux virtuels différents

5.3.2 Filtre multi-frontières

Trois frontières de sécurité peuvent être considérées entre les zones sûre et non sûre. Chacune peut être équipée d'une protection spécifique (figure 5.3). La première protection est une vérification d'autorisation à l'accès aux interfaces du NoC. La seconde est un filtre de chemin inséré dans le réseau. La troisième protection est une vérification par la NI destinataire de l'identité de l'émetteur qui lui adresse un paquet. Ces trois frontières de sécurité sont expliquées dans les sections suivantes.

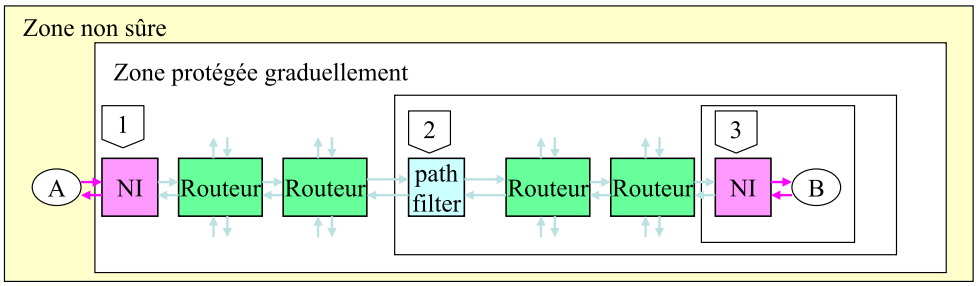


FIG. 5.3 – Trois frontières de sécurité

5.3.2.1 Frontière 1

Tout le NoC est dans la zone sécurisée. Certaines IPs sont aussi dans la zone sécurisée parce qu'elles sont critiques, mais d'autres IPs sont elles dans la zone non sûre et doivent pouvoir n'accéder qu'à certaines informations dans des fenêtres de temps limitées. La figure 5.4 présente un exemple.

Les détails de notre NI sont montrés sur la figure 5.5. Les NIs contiennent de nombreux paramètres de configuration. Ces paramètres sont configurables au travers du NoC lui même. Nous distinguons les éléments suivants dans la NI :

- Un adaptateur de protocole pour communiquer de façon appropriée avec le protocole du port de la NI connectée.
- Une table d'espace d'adressage mémoire qui spécifie les droit d'accès. De plus, elle permet de convertir l'adresse logique demandée en l'identité de l'IP cible.

- Une table de chemins, fournissant les instructions de chemin nécessaires pour atteindre une destination atteignable (autorisée).
- Une table des slots de temps alloués pour le GT.
- Une instruction spécifiant la bande passante maximale pouvant être allouée au BE. Ceci permet de maintenir la bande passante d'une communication dans une limite spécifiée.

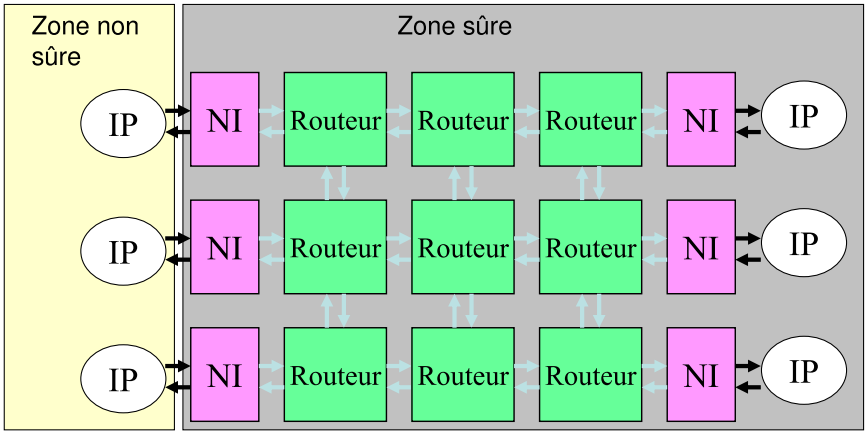


FIG. 5.4 – Exemple de NoC entièrement dans la zone sûre

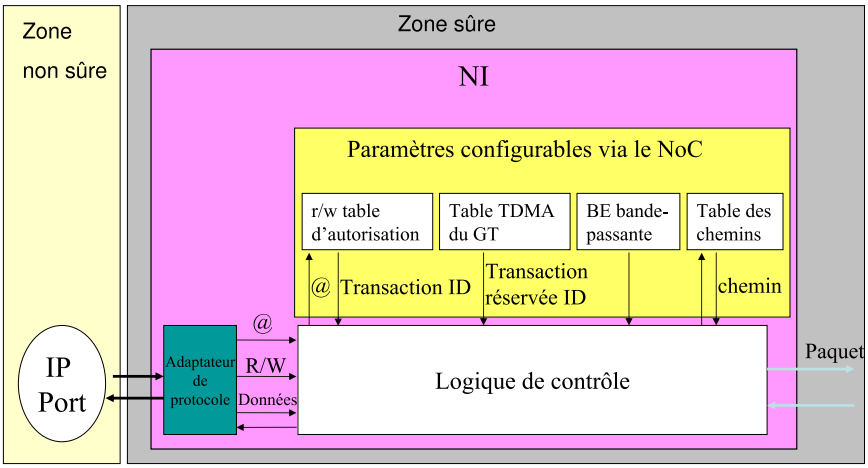


FIG. 5.5 – Network Interface

Le CCM délivre aux NIs des autorisations d'accès mémoire. Les NIs sont limitées par ces droits. Une NI doit solliciter le CCM pour demander une nouvelle autorisation. Le CCM peut retirer une autorisation.

Ceci nous prémunit des attaques de type déni de service, grâce à la limitation de la bande passante et nous permet de contrôler les droits d'accès.

5.3.2.2 Frontière 2

Seule une partie du NoC est dans la zone sécurisée. Dans ce cas, nous ne pouvons pas avoir confiance dans les NIs se trouvant dans la zone non sûre. Ainsi, ici, l'information

qui sera vérifiée ne sera non pas l'adresse mémoire logique comme précédemment, mais le contenu du paquet. En effet, le champ d'instruction de chemin dans l'entête du paquet indique comment atteindre la destination, ce qui permet de déduire la destination et correspond donc à l'adresse logique globale précédente.

Des filtres appelés *path filter* sont insérés dans la zone sûre à la frontière avec la zone non sûre (figure 5.6).

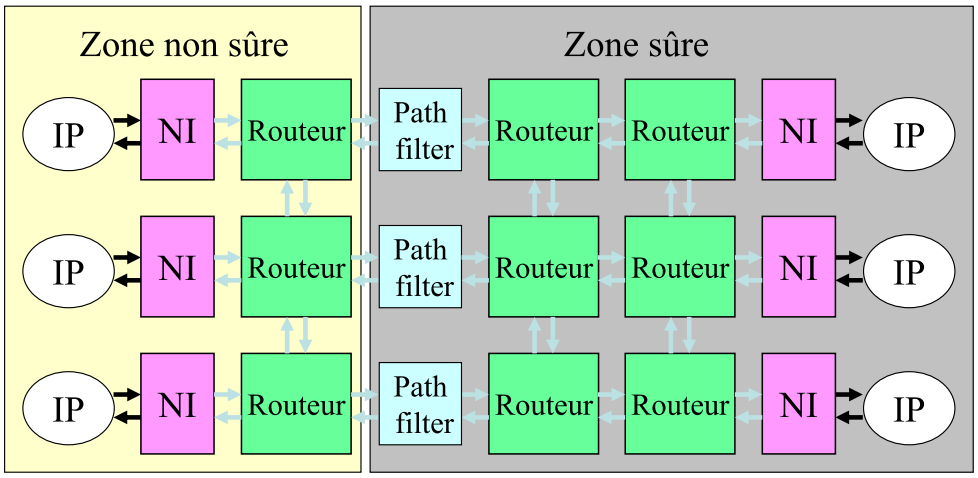


FIG. 5.6 – Noc dans la zone sûre et non sûre

Dans ce cas, le CCM délivre des autorisations relatives à des chemins (et non des espaces mémoires logiques comme précédemment) aux *path filters*. Un *path filter* reçoit des paquets de la zone non sûre et n'autorise à pénétrer dans la zone sécurisée que ceux qui ont des instructions de chemin autorisées. Cette solution empêche un émetteur de la zone non sûre d'accéder à un destinataire auquel il n'est pas autorisé. De plus, cette vérification par les instructions de chemin nous prémuni des *deadlock* et *livelock*.

Un *path filter* ressemble à une NI, excepté qu'il n'est pas connecté à une IP et n'a donc pas d'adaptateur de protocole, et qu'il possède une table de chemins à la place de la table d'espace d'adressage mémoire (figure 5.7) En pratique, pour éviter de perdre du temps, le *path filter* peut être intégré dans le port du routeur de la zone sûre. Cela permet de procéder simultanément à la vérification du chemin et à l'étape de décodage.

5.3.2.3 Frontière 3

Hypothèse de cette étude : Nous proposons ici de faire les vérifications sur les informations de la couche réseau. Ainsi, on ne sécurise pas une application vis à vis d'une autre mais assurons l'authentification d'un composant connecté au NoC afin de lui attribuer des droits d'accès avec les autres. Pour cela nous nous servons des instructions de chemin comme des marques indiquant au récepteur le chemin emprunté par le paquet. L'intérêt de se servir des instructions, plutôt que de marques qui seraient ajoutées par les routeurs traversés, est que nous économisons de l'espace dans l'entête du paquet.

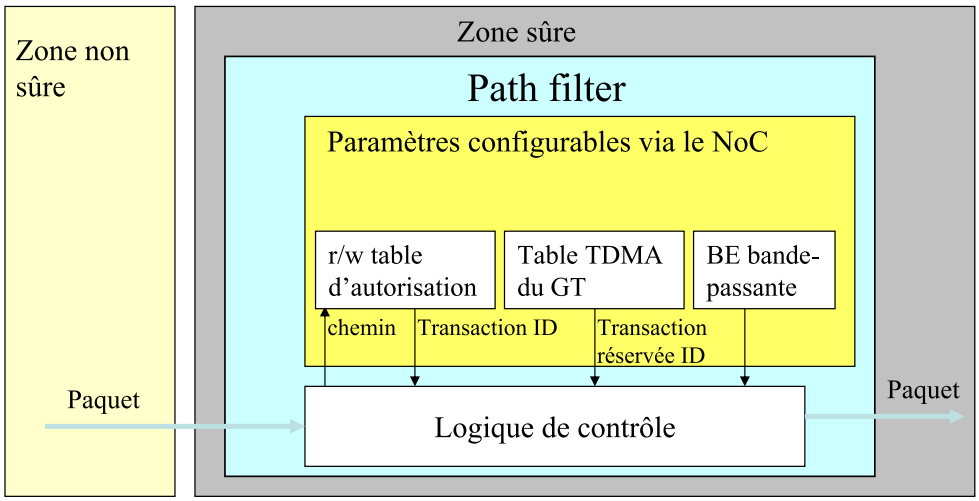


FIG. 5.7 – Le *path filter*

De cette façon, un composant destinataire peut vérifier l'identité de l'émetteur. Ainsi, l'accès à une mémoire sera autorisé depuis certains processeurs et certaines interfaces mais pas depuis d'autres.

Cette authentification du composant source pourrait être réalisée par un identifiant qui serait placé dans l'un des champs de l'entête du paquet lors de sa création dans la NI. Cet identifiant serait alors consulté à la réception afin de vérifier l'identité de l'émetteur. Seulement, cet identifiant pourrait être usurpé, pour permettre à un composant de se faire passer pour un autre.

Nous proposons ici de nous appuyer sur la couche transport pour plus de sécurité. Ainsi, nous utilisons une technique de codage des instructions de routage qui permet à un destinataire de remonter vers l'identité de l'émetteur. En effet, la seule information qui ne puisse être corrompue pour tenter d'usurper une identité est l'information de chemin pour atteindre la destination, à condition que celle-ci soit unique et dans l'hypothèse que l'intégrité du NoC est garantie bien entendu. Nous appelons cette technique SPA (pour *Source Path Authentication*)

5.4 Les techniques classiques de codage des instructions de routage

5.4.1 Codage pour le routage ordonné par dimensions X-Y

Le champ des instructions de chemin du paquet contient deux valeurs, X et Y. Ces valeurs sont exprimées soit en relatif à la coordonnée de l'émetteur (le nombre de sauts à réaliser dans chaque dimension pour arriver au destinataire), soit en absolu (les coordonnées de la destination). Les routeurs acheminent le paquet sur la dimension X puis sur la dimension Y (dans le cas du routage relatif, à chaque saut, ils décrémentent la valeur correspondant à cette dimension). Ainsi arrivée à destination, ces valeurs sont nulles dans le cas du routage relatif et valent les coordonnées de la destination dans le cas du routage absolu.

5.4.2 Codage *street-sign*

Le codage *street-sign* utilise une liste d'instructions du type Nord, Sud, Est, Ouest et descendre, indiquant la direction à prendre à chaque routeur, ou à des routeurs identifiés. Une fois exécutée, l'instruction courante est supprimée et les instructions restantes sont décalées pour que la suivante prenne sa place en tête de liste. Ainsi, à la réception, les instructions ont toutes été supprimées. Conserver ces instructions ne serait pas suffisant pour identifier l'émetteur car ce type d'instruction ne permet de connaître que le port de sortie du routeur et pas le port par lequel le paquet est entré.

5.4.3 Analyse

Ces codages ne permettent pas au destinataire d'authentifier la provenance du paquet. Nous proposons donc l'approche suivante.

5.5 Exploitation de l'information de routage

5.5.1 Principe du codage relatif pour l'authentification

Nous expliquons ici comment nous pouvons utiliser l'information de routage pour son utilisation au niveau de la destination.

Nous avons vu qu'avec les techniques classiques, les informations de routage ne sont pas conservées ou sont inexploitables au niveau de la destination.

Notre technique de codage des instructions de chemin doit offrir une clef unique à destination permettant d'identifier l'émetteur. Nous proposons une technique de codage avec des perfectionnements qui permettent de tirer parti de cette information pour deux objectifs, la sécurité et la reconfiguration.

Notre codage fonctionne sur le modèle du codage *street-sign*, mais de manière relative. Le port de sortie d'un routeur est indiqué relativement au port par lequel le paquet est entré. Ainsi, dans le routeur, comme cela peut se faire dans le cas d'un rond point, la sortie est indiquée par son occurrence à partir de l'entrée par laquelle le paquet est arrivée. Comme le montre la figure 5.8, nous avons choisi arbitrairement de compter le numéro des sorties dans le sens inverse des aiguilles d'une montre dans le cas d'une topologie 2D. Le paquet sortira par la $n^{ème}$ sortie. De plus, les instructions doivent être préservées. Ainsi, les instructions ne sont plus supprimées après avoir été exécutées, mais décalées de façon circulaire.

Le champ des instructions de chemin du paquet indique ainsi de façon unique l'identité de l'émetteur. L'émetteur ne peut se faire passer pour un autre auprès d'un destinataire car en mentant sur ses instructions de chemin, il n'arriverait pas à ce destinataire.

Aucun champ supplémentaire n'est utilisé. C'est donc une façon simple et sans surcoût de disposer d'une trace indiquant la provenance d'un paquet, une sorte de passeport ou certificat d'identité.

Propriété 1 : Le champ des instructions de chemin du paquet constitue un certificat permettant au destinataire de vérifier l'identité de la NI émet-

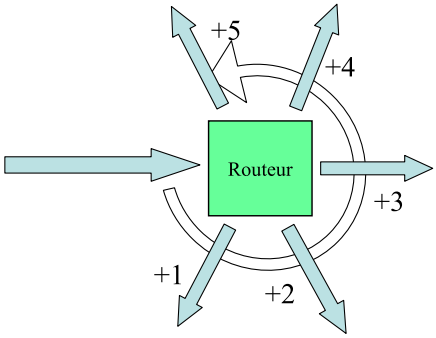


FIG. 5.8 – Le codage des instructions de chemin selon un codage relatif à la source

trice.

5.5.2 Le complément automatique du chemin (SCP)

Notre technique de codage peut aussi permettre de construire le chemin inverse de façon automatique. Nous appelons cela le codage SCP (*Self Complemented Path*). En effet, le chemin retour peut être déduit du chemin aller et de l'arité (nombre de ports) de chacun des routeurs traversés.

Le nombre de ports de chaque routeur n'est pas connu du récepteur, en revanche chaque routeur est bien placé pour connaître son nombre de ports. Ainsi, le remplacement des instructions aller par les instructions retour est fait au fil de la traversée des routeurs (figure 5.9).

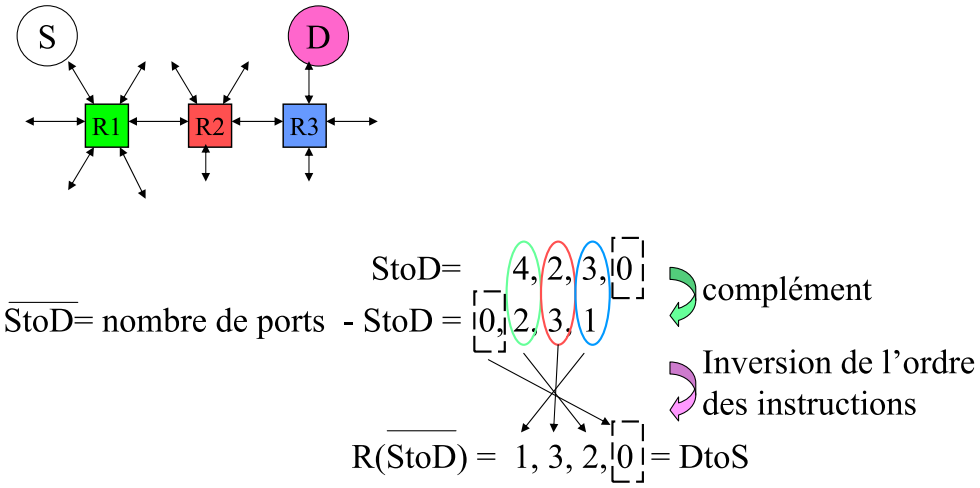


FIG. 5.9 – Les instructions de chemin aller et retour

Propriété 2 : À chaque routeur, la somme de l'instruction courante dans le sens aller et de l'instruction correspondante dans le sens retour est égale à l'arité du routeur.

Quand le manager (CCM) crée une connexion entre deux NIs, NIa et NIb, il ne leur

donne que les instructions aller. Lorsqu'ils s'échangent des paquets, chaque routeur traversé exécute l'instruction de chemin courante et la complémente en fonction de son arité et décale les instructions de façon circulaire comme expliqué dans la figure 5.9. La restriction que nous nous fixons ici est que les paquets aller et retour empruntent exactement les mêmes chemins mais dans des directions opposées.

Propriété 3 : Les instructions d'un chemin, une fois complémentées et organisées dans l'ordre inverse, sont identiques aux instructions de chemin dans le sens opposé. Nous notons ceci de la façon suivante :

$$R(\overline{AtoB}) = BtoA$$

Ainsi, chaque NI vérifie la propriété 3, pour authentifier l'identité de l'émetteur.
Si nous voulons nous prémunir des *livelocks*, il existe deux solutions.

- Un champ supplémentaire dans l'entête du paquet réalise un comptage du nombre de sauts effectués (nombre de liens traversés) et permet aux routeurs de supprimer ce paquet quand ce nombre dépasse une limite fixée.
- Une instruction de fin de chemin peut être utilisée (égale à 0 dans la figure 5.9).
  Propriété 4 : Dans la liste des instructions de chemin, une instruction de fin de chemin signale que le paquet est arrivé à destination.
  Dans le cas d'un transfert correct, l'instruction de fin de chemin ne devient l'instruction courante que lorsque le paquet arrive à la NI destinatrice. Cette dernière vérifie alors que cela est bien le cas. Si ce n'est pas le cas le paquet est considéré incorrect et est supprimé. De plus, chaque routeur vérifie deux choses. Premièrement il vérifie que l'instruction courante n'est pas l'instruction de fin de chemin. Deuxièmement, il vérifie que l'instruction fin de chemin est bien présente parmi les instructions. Si l'une de ces deux conditions n'est pas respectée, alors le routeur supprime le paquet.

La figure 5.10 montre un exemple. B autorise uniquement A à lui émettre des paquets. C tente d'envoyer un paquet à B. Lorsque la NI de B reçoit le paquet, elle vérifie le champ des instructions de chemin et le compare à ceux autorisés, elle détecte ainsi l'émission non autorisée.

De plus, le champ des instructions de chemin du paquet reçu est celui nécessaire pour répondre par le même chemin excepté le fait que l'ordre des instructions est inversé. Le coût mineur dû à l'opération de complément dans les routeurs est contre balancé par l'économie en mémoire et communication, car le CCM ne fournit que les instructions de chemin pour émettre et un bit de statut par chemin pour indiquer si le sens réception est autorisé.

Notons que dans le contexte de la sécurité l'opération de complément n'est pas obligatoire. Cependant, elle est intéressante puisqu'elle rend l'authentification dépendante du chemin emprunté et de l'arité des routeurs traversés.

Cette technique est efficace en termes de délai et d'intégration puisque l'instruction complémentée est codée en parallèle de l'étape de décodage de l'instruction et que la soustraction optimisée pour l'arité du routeur peut être facilement câblée.

Le champ d'instructions de chemin pour répondre au CCM est stocké dans toutes les NIs, de sorte qu'elles puissent l'identifier et que l'on ne puisse usurper son identité.

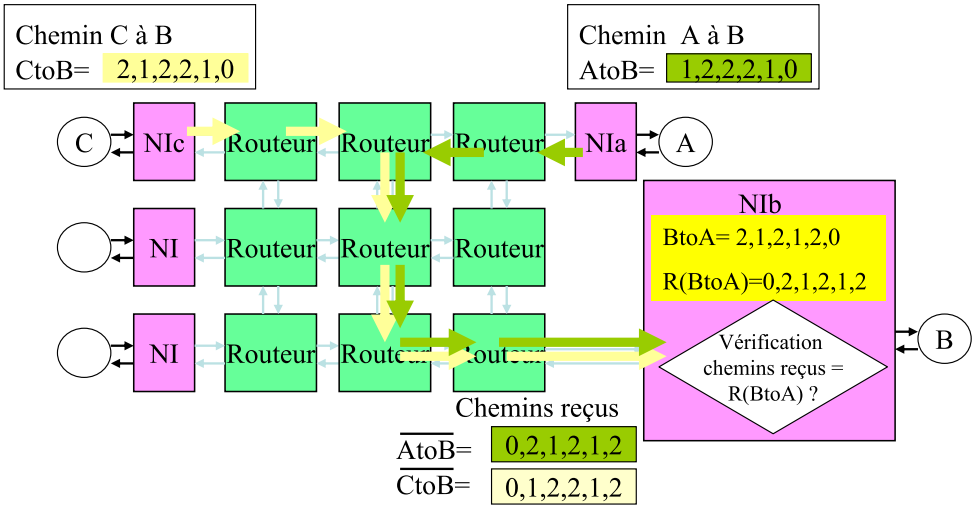


FIG. 5.10 – Exemple d'authentification par le chemin

5.5.3 Utilisation pour la reconfiguration

Notons que la propriété 2 a un autre grand intérêt que nous appelons *Trusted Boomerang Path* (TBP). Si la sécurité n'est pas le but recherché, la propriété de transformer le chemin aller en chemin retour est utile pour permettre à un bloc IP esclave de répondre à une requête de lecture émise depuis n'importe quel bloc IP maître. Ainsi, aucune table de chemin n'est nécessaire pour indiquer le chemin des esclaves vers les maîtres, ce qui revêt un intérêt majeure en termes de configuration et d'économie de taille mémoire. Le paquet reçu par l'esclave fournit directement les instructions de chemin nécessaires pour répondre au maître en question. Ainsi, le grand avantage de cette technique est que l'esclave peut ignorer la localisation du maître, ce qui permet la mobilité des IPs dans un système reconfigurable [79].

Une transaction entre un maître et un esclave est décrite sur la figure 5.11. Le maître NIa utilise l'authentification SPA alors que l'esclave NIb utilise le TBP pour répondre. La technique SCP réalise le complément des instructions de chemin.

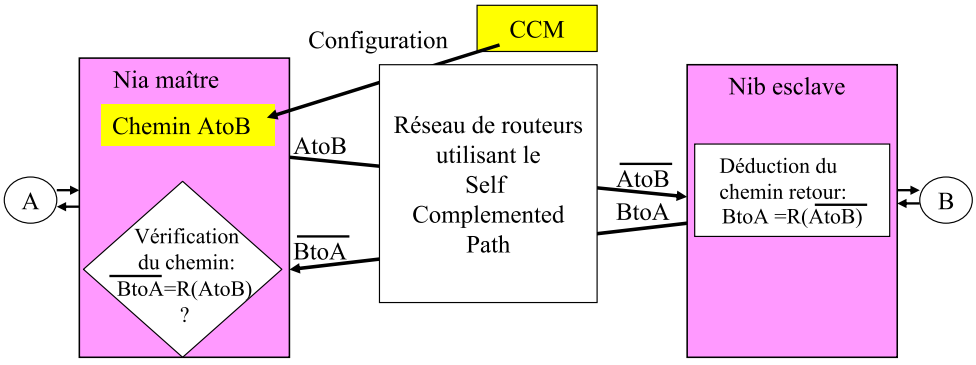


FIG. 5.11 – Boomrang

5.5.4 Le réarrangement binaire automatique des instructions de routage

Le nombre de ports de chaque routeur varie et le nombre de bits nécessaires au codage des instructions peut donc varier aussi. Pour réduire le codage du champ d'instructions, il est donc souhaitable de ne pas utiliser un codage de largeur fixe car cela conduirait à utiliser la largeur la plus grande et donc un champ de grande taille dans l'entête. Si les instructions n'ont pas la même largeur, un problème se pose pour inverser l'ordre des instructions au niveau du destinataire, car celui-ci ne connaît pas la taille de chacune et ne voit qu'un champ de bits.

Nous proposons alors la solution suivante. Elle consiste à ce que le routeur, en plus de complémenter l'instruction courante, inverse le poids des bits de l'instruction. L'instruction se retrouve ainsi retournée. Le destinataire n'a alors plus qu'à retourner l'ensemble du champ des instructions de chemin, pour que l'ordre des instructions soit inversé. C'est donc une façon simple d'inverser l'ordre des instructions indépendamment du fait qu'elles aient des codages de largeurs variées (en nombre de bits) et inconnues du destinataire. Ce principe, conjugué aux SPA, TBP et SCP a fait l'objet d'un brevet en 2005.

La figure 5.12 montre un exemple. Une source et une destination sont séparées par 4 routeurs (R1 à R4). Les ports des routeurs ne sont pas tous représentés. La source dispose des instructions aller destinées aux routeurs R1 à R4. Au cours du cheminement du paquet les routeurs traversés remplacent l'instruction aller par l'instruction retour et inversent l'ordre des bits (poids fort - poids faible). Le destinataire qui n'a pas connaissance de la taille de chaque instruction retourne l'ordre des bits de tout le champ d'instruction. Il obtient ainsi le chemin de retour, avec les bonnes instructions de retour et dans le bon ordre.

5.5.5 Encryptage du *bitstream*

La configuration du CCM doit être sûre, car c'est le point clé de la sécurité du NoC. Pour empêcher un accès mal intentionné au CCM, une technique d'encodage standard peut être utilisée. La configuration du CCM n'est généralement pas fréquente, donc l'encryptage et le décryptage ne constitueront pas une perte de temps significative.

Quand une zone non sûre est intégrée dans un FPGA, une authentification classique peut être utilisée pour vérifier l'identité d'une IP et éviter le remplacement de l'IP autorisée par un usurpateur. L'authentification de l'IP peut être protégée avec une technique telle que l'encryptage du *bitstream* [80].

5.6 Contre-attaque

5.6.1 Réaction face à un routage erroné

Quand une NI de la zone sécurisée reçoit un paquet d'un émetteur non autorisé, elle avertit le CCM. De multiples raisons peuvent être à l'origine de ce résultat, si bien que le CCM peut réagir de différentes façons :

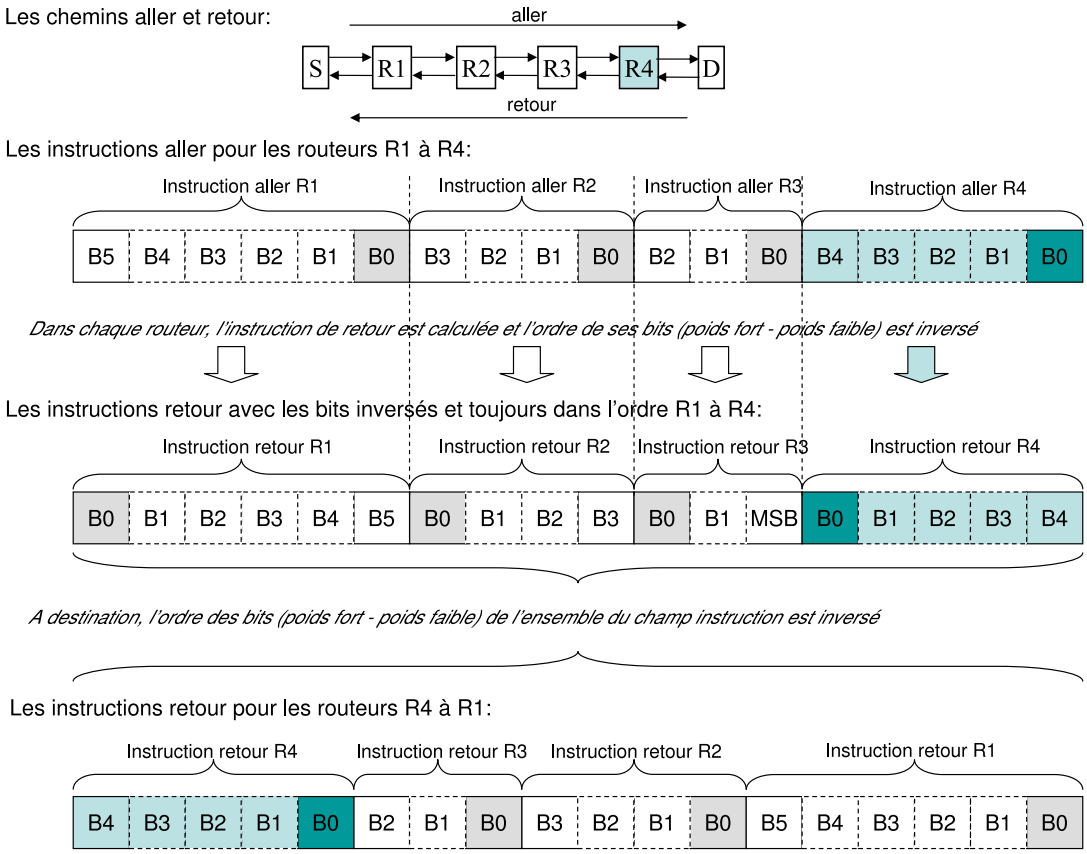


FIG. 5.12 – Double inversion pour inverser l'ordre des instructions

- Premièrement, le problème peut être causé par une erreur de transmission entre l'émetteur et le récepteur. Le CCM demande à l'émetteur suspecté de réémettre la donnée perdue.
- Le problème peut aussi être le résultat d'une erreur de configuration. Le CCM procède alors à la reconfiguration de l'émetteur et du récepteur.
- Si l'erreur persiste et qu'un compteur d'erreurs dans le CCM atteint un seuil spécifié, le CCM identifie cela comme une attaque et réduit à zéro la bande passante de la NI de l'IP en faute.

5.6.2 Rétro-ingénierie et extraction d'informations secrètes par analyse différentielle de puissance

Le NoC peut aussi être utilisé dans l'intention d'augmenter la sécurité du système. En changeant alternativement les chemins utilisés, l'observation des échanges devient plus difficile. De plus, ceci peut être complété par le changement de place des IPs maîtres avec la technique TBP présentée. Ceci peut préserver le système contre des extractions de clés par DPA [78].

5.6.3 Supervision de bande passante et de consommation

Le contrôle de la consommation d'énergie peut être réalisé par le CCM pour contrôler le système en fonction de son espérance de vie. Si la consommation de puissance augmente au dessus d'un seuil, le trafic BE est réduit pour prévenir le risque d'une attaque ayant pour but de vider les batteries [81]. Le CCM est en charge de la configuration, de la surveillance et du contrôle des NIs. Cela suppose de la part du CCM une certaine connaissance à priori du comportement attendu de l'application. Ainsi, un profil doit être réalisé et certains seuils doivent être identifiés pour permettre au CCM de détecter un comportement irrégulier du NoC et pour lui permettre par conséquent de lancer des réactions en ajustant des paramètres dans les NIs afin que le système retrouve un comportement régulier. Cet aspect n'est pas traité ici et fait l'objet d'un nouveau thème de recherche au laboratoire LESTER.

5.7 Conclusion

Nous avons vu les différentes attaques que peut subir un NoC. Nous avons également vu comment un NoC peut être protégé au moyen de *path filter*, ou NI sécurisée. Une technique de codage des instructions de chemin appelée SCP permet à un destinataire d'authentifier l'identité de l'émetteur du paquet. Dans l'hypothèse ou l'intégrité de la topologie du NoC est assurée cette technique est fiable. Et surtout elle ne comporte pas de coût supplémentaire en terme de contenu dans l'entête du paquet contrairement à un identifiant qui serait ajouté. Nous avons vu comment cette technique peut permettre d'offrir de la mobilité aux IPs maîtres.

Le tableau 5.2 montre les différentes attaques et indique par un X les protections adaptées.

Scénario d'attaque	Stratégies de protection						
	NIs dans la zone sûre	path filter	SPA	VCs	Bitstream encodé	Surveillance et contrôle des bandes passantes	Mobilité des IP avec TBP
Déni de bande passante				X		X	
Chemin incorrect	X	X	X				
Deadlock	X	X	X				
Livelock	X	X	X				
Lecture non autorisée	X	X	X		X		
Ecriture non autorisée	X	X	X		X		
Extraction d'information					X		X
Observation en fonctionnement							X

TAB. 5.2 – Les stratégies de protection aux attaques

Chapitre 6

NoC avec trafic garanti dans une approche de type GALS

6.1 Introduction

Les futurs systèmes sur puce ne permettront plus la distribution d'une horloge globale sur l'ensemble du circuit. Le réseau doit donc prendre cette contrainte en compte. Nous proposons ici une solution qui vise à conserver la garantie de service offerte par les TDMAs, tout en étant compatible avec des décalages d'horloges et des horloges hétérogènes du circuit [82].

6.2 La synchronisation et la QoS

6.2.1 Problématique

Pour s'intégrer avec les règles de conception des systèmes synchrones, l'intégration traditionnelle du NoC repose généralement sur une horloge système globale avec des déphasages négligeables (*skew*). Cette hypothèse de conception n'est pas compatible avec les futurs SoCs dans lesquels il faudra plusieurs cycles d'horloge pour transporter un signal au travers de la puce [7]. Dans le cas idéal, un délai de 100 pico secondes est nécessaire pour traverser en diagonale une puce de 22mm de coté avec une technologie à 50nm. En pratique, avec une horloge à 10GHz, il est estimé qu'il faut entre 6 et 10 cycles d'horloge pour traverser le circuit. Ceci pause un problème de synchronisation.

De plus, la conception d'un arbre d'horloge distribuant celle-ci de façon uniforme sur l'ensemble du circuit deviendra très difficile [5]. On ne pourra donc plus assurer une synchronisation totale de l'horloge sur l'ensemble de la puce.

Un autre défaut de l'intégration des NoCs traditionnels, est celui de l'utilisation d'un domaine d'horloge unique, alors que les larges SoCs tendent à se répartir en plusieurs domaines d'horloge, notamment pour des raisons de consommation. Enfin, des circuit multi-cœurs (*multi-core*) ou bien système multi-puce, impliquent intrinsèquement des horloges distinctes.

Tous ces cas compliquent la conception et impliquent des challenges au concepteur de NoC qui devra réaliser des circuits de grande taille où les besoins sont grandissants

en terme de flexibilité, de capacité à s'étendre, de QoS et de gestion des communications.

6.2.2 Synchroniser un système

Il existe plusieurs méthodes pour synchroniser un système :

- L'approche traditionnelle est celle du système synchrone, qui consiste à utiliser une horloge globale sur tout le système avec un décalage de phase pouvant être négligé.
- La seconde approche est la réalisation d'un système totalement asynchrone. Les communications se font alors par un protocole de poignée de mains, ou bien par des mécanismes spéciaux au niveau logique (2 ou 4 phases). L'asynchrone a pour avantage de réduire la consommation dynamique.
- La troisième méthode consiste à utiliser des blocs synchrones qui communiquent entre eux de façon asynchrone. Ce concept est appelé GALS (*Globally asynchronous, locally synchronous*). Il a pour avantage de permettre d'intégrer ensemble des domaines d'horloges différents.
- Une autre approche consiste en la distribution d'horloges mesochrones [83]. Une horloge commune est distribuée sur tout le système. Les différentes parties du système ont une horloge de même fréquence, mais des phases différentes. Ainsi, chaque partie peut fonctionner comme un systèmes synchrone. Les communications entre ces parties nécessitent une adaptation pour s'accommoder du décalage de phase.
- Enfin, il existe les systèmes plesiochrones [(1)]. Les systèmes sont presque parfaitement synchronisés, mais diffèrent légèrement en fonction de leurs horloges qui sont locales. La différence est maintenue dans une limite spécifiée.

6.2.3 État de l'art de la gestion de la QoS et des horloges dans les NoCs

Les NoC peuvent être classés par catégories en fonction de leur utilisation de l'horloge et de la qualité de service qu'ils assurent à leurs communications.

L'approche traditionnelle considère un NoC synchrone avec un domaine d'horloge unique. Elle a pour avantage de permettre d'obtenir un système parfaitement déterministe. Ainsi, elle permet l'utilisation d'un TDMA au travers du circuit pipeliné pour assurer un fonctionnement qui a été pré-ordonnancé pour satisfaire la QoS [22] [21]. Cela signifie que tous les éléments routeurs et NIs ont une horloge commune. De plus, le délai d'acheminement sur un lien du réseau doit se faire en moins d'un cycle d'horloge afin que le pipeline des routeurs traversés soit respecté. Notons qu'il y a généralement tout de même une frontière de synchronisation entre les NIs et IPs, réalisée à l'aide de FIFOs double horloge.

Une autre approche consiste à utiliser des communications asynchrones entre des composants synchrones (GALS). Dans [84], les auteurs proposent un crossbar asynchrone, seulement la contrainte de temps réel n'est pas considérée.

L'approche de type GALS est utilisée par la solution industrielle fournie par Arteris [13]. Des IPs sont regroupés en clusters. Les échanges dans les clusters sont synchrones

[(1)] Le terme plesiochrone est dérivé du Grec *plesio* (proche) et *chronos* (temps)

et garantis. Un réseau d'interconnexion asynchrone permet de transporter des paquets entre des groupes ou ensembles (clusters) d'IPs synchrones. Les échanges inter-clusters réalisés par des échanges asynchrones n'offrent eux pas de garantie « stricte ».

La conception d'un système sans aucune horloge est la version ultime de l'asynchrone, c'est la solution intégrée par [85] et [86]. Dans le premier cas, des canaux virtuels et des routeurs avec de faibles latences sont utilisés, cependant l'architecture du NoC n'offre pas de garantie de bande passante. Dans le second cas, un ordonnancement appelé ALG (*Asynchronous Latency Guarantee*) est utilisé. La garantie est obtenue sous certaines conditions. Il faut en effet que le délai maximum de transmission d'un flit soit connu et que l'intervalle inter-flits soit borné. Cette approche est intéressante, cependant, l'intégration d'un grand nombre de trafics à garantir, se traduit par l'ajout de nombreux canaux virtuels ce qui implique une augmentation importante du coût en surface au niveau des routeurs.

Les avantages principaux des solutions sans horloge sont l'insensibilité à la latence et la réduction de la consommation car il n'y a pas de transition quand le système est inactif. Cependant, ces solutions requièrent une implantation spécifique utilisant généralement un protocole double rail à quatre phases, où chaque bit est codé sur quatre fils. C'est le cas des solutions [85] et [86], où les exemples d'implantations ne sont donnés que pour un seul routeur. La consommation dynamique est réduite avec un NoC asynchrone, mais la surface nécessaire pour son implantation peut augmenter sa consommation statique.

6.2.4 Contexte dans lequel nous nous plaçons

Du point de vue de la qualité de service : La technique par TDMA est une technique efficace et simple pour assurer la qualité de service, mais nécessite une notion de synchronisation pour que le comportement pré-ordonnancé soit respecté.

Du point de vue des horloges : La réutilisation de ce qui a déjà été conçu impose d'être compatible avec les architecture synchrones et les outils de conceptions construits autour des protocoles de communication standards. De plus, les larges SoCs tendent à se répartir en plusieurs domaines d'horloges. Le concept de GALS semble donc être la solution la plus adaptée.

Enfin, il est possible d'avoir des parties du circuit qui ont une horloge de même fréquence mais pas de même phase. On doit aussi faire face a des liens qui ont un temps de traversée supérieur à un cycle d'horloge. Il est donc aussi nécessaire de traiter l'aspect des parties mésochrones.

Nous cherchons donc à la fois à conserver la QoS par l'utilisation de TDMAs tout en se mettant dans le contexte de domaines d'horloges différents et de sous parties mésochrones.

6.2.5 Résumé de l'approche développée

Notre objectif a été de conserver la QoS à l'aide du TDMA, mais dans un contexte de GALS qui répond mieux aux futures contraintes d'implantation de la distribution de l'horloge dans les SoCs. La figure 6.1 illustre notre approche.

Cette solution consiste en le découpage du réseau global du circuit en zones de même horloge appelées sous-réseaux ou sub-NoCs. Chacun des sub-NoCs fonctionne de

manière synchrone (ou vu comme synchrone), ainsi son comportement est prédictible et permet d'utiliser un TDMA pour l'ordonnancer et garantir le trafic qui le parcoure. De plus, un sub-NoC peut être lui même divisé en sous parties mésochrones.

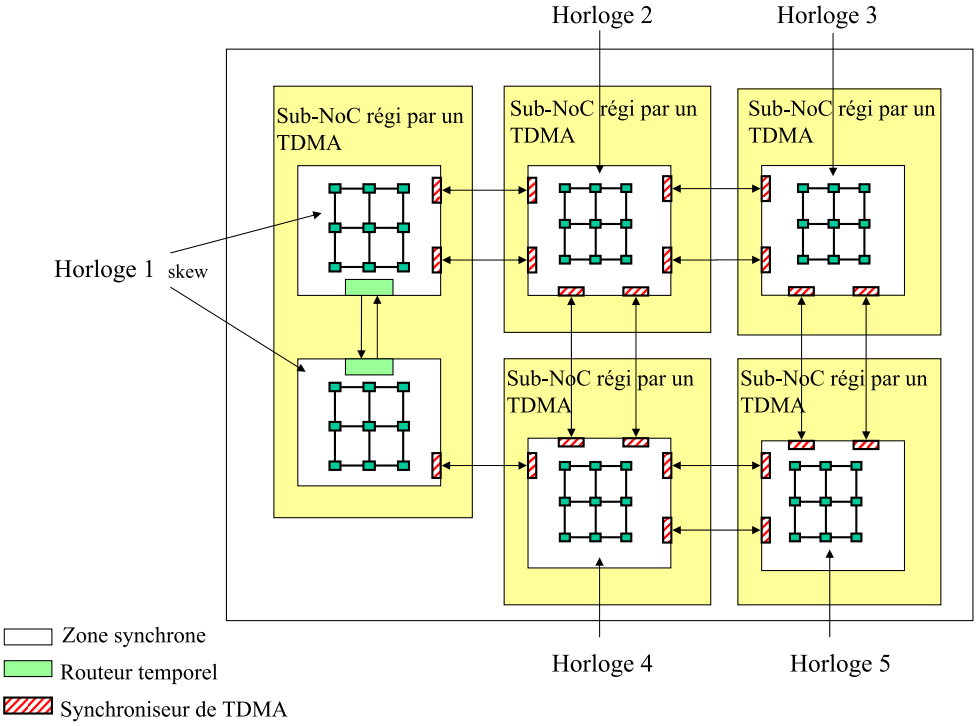


FIG. 6.1 – NoC GALS avec QoS

Il y a donc deux difficultés à surmonter :

- *À* l'intérieur d'un sub-NoC, il faut pouvoir masquer le problème de liens qui ont un délai de traversée trop long pour permettre le pipeline de la communication au travers du sub-NoC. De plus, il faut pouvoir s'accommoder des parties mésochones. Pour cela, on introduit le concept de routage temporel (TR pour *time-routing*) qui réalisera l'interface.
- De plus, il faut pouvoir garantir le trafic au niveau global, c'est à dire sur tout le circuit. Nous proposons des interfaces de synchronisation qui vont permettre de faire tampon entre les différents sub-NoCs qui sont ordonnancés individuellement avec leur propre table TDMA.

À l'aide de nos deux types d'interfaces, la QoS du trafic est assurée sur toute la puce malgré l'hétérogénéité des horloges.

6.3 Le routage temporel

6.3.1 Problématique

Pour que l'ordonnancement des communications par le TDMA soit correct, il est impératif que les flits des paquets arrivent à un moment bien déterminé. Le routage temporel répond au problème posé par les liens avec des délais supérieures à la durée

d'une période d'horloge, ainsi qu'au problème du décalage de phase (skew) des horloges entre deux parties d'un sub-NoC (parties mésochrones) (figure 6.2).

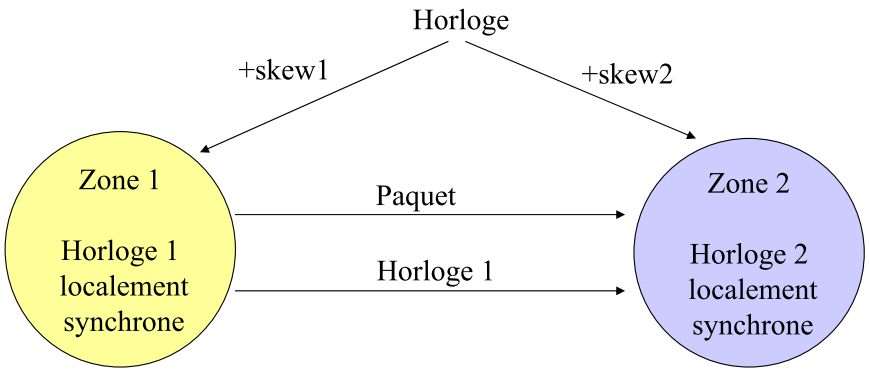


FIG. 6.2 – Le *skew* entre les horloges

Le bon verrouillage des données à l'extrémité de réception du lien dépend du délai de transmission qui est lui-même fonction des caractéristiques du lien physique sur lequel les données circulent et du décalage (skew) entre les horloges à chaque extrémité du lien.

Un mauvais verrouillage rompt le pipeline instauré par le TDMA pour le cheminement des paquets dans le sub-NoC et par là même la garantie de service.

La solution classique utilisant des stations relais [87] où le pipeline sur le lien nécessite la connaissance à priori du délai sur le lien et d'ajouter des éléments le long de ce lien pour le pipeliner. Le problème est que cet ajout peut remettre en cause le placement et le routage et ainsi déplacer le problème. De plus, dans le cas extrême ou le sub-NoC se trouverait réparti sur deux puces distinctes, ces solutions ne pourraient pas s'intégrer sur le lien entre les deux circuits du silicium.

6.3.2 Architecture

Pour se prémunir des variations de *skew* et des délais des liens, ces deux valeurs sont bornées par des valeurs maximales. Pour pouvoir garantir le pré-ordonnancement, le délai considéré est le pire cas, de sorte que ce délai soit toujours le même (ni inférieur, ni supérieur) quelles que soient les variations de ces deux valeurs dans les limites fixées.

Ce contrôle est réalisé en intégrant deux éléments, le codeur temporel (TC pour *Time Coder*) et le routeur temporel (TR pour *Time Router*) dans le port de sortie du routeur émetteur et dans le port d'entrée du routeur récepteur respectivement (figure 6.3).

6.3.3 Calcul de la taille de la FIFO

Pour réaliser l'interface entre les deux domaines temporels, nous utilisons une FIFO double horloge et transportons le signal d'horloge du TC vers le port d'écriture de la FIFO dans le TR. L'architecture de la FIFO double horloge n'est pas détaillée ici.

Le TC ajoute une instruction de numéro de slot temporel dans l'entête des paquets qui le traversent. L'instruction de numéro de slot temporel indique au TR le slot temporel durant lequel il devra émettre ce paquet. Le TR n'a pas de table TDMA,

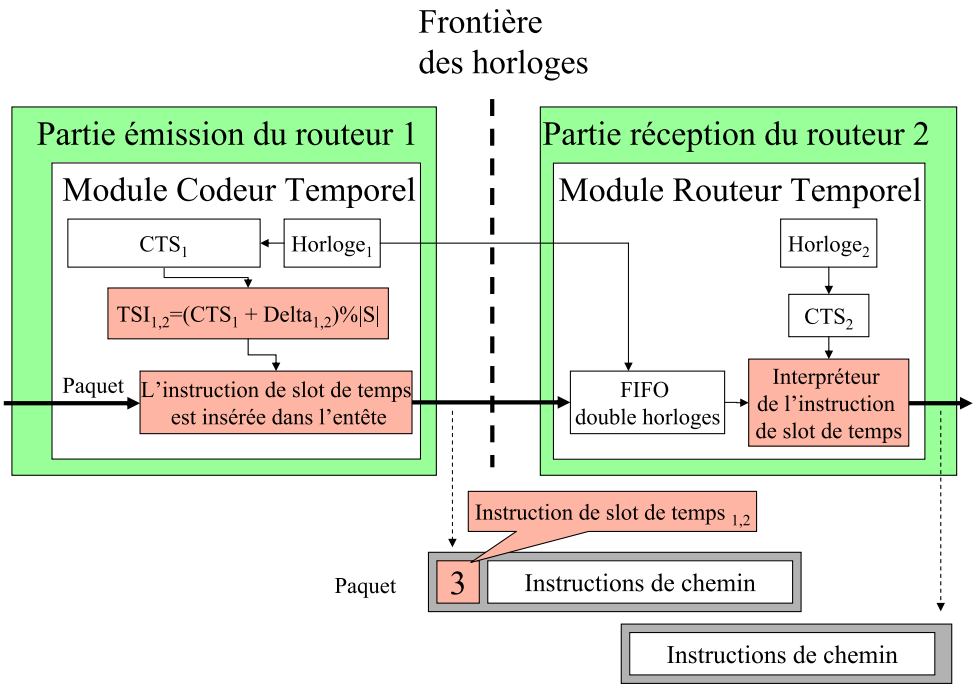


FIG. 6.3 – Le Routage temporel

cependant il est conscient du slot temporel courant grâce à un simple compteur qui incrémente les slots qui s'écoulent (dans son domaine temporel propre). Le paquet reçu par le TR est stocké dans une FIFO et expédié uniquement quand le slot temporel courant et le slot temporel de l'instruction sont identiques.

La figure 6.3 montre l'architecture du codeur temporel et du routeur temporel, de chaque coté d'un lien unidirectionnel dont le délai est incertain. Notons que l'exécution des TC et TR se fait en parallèle avec l'exécution des routeurs, ainsi, il n'y a pas de délai supplémentaire introduit sinon celui spécifié.

Nous allons déterminer la différence maximale des slots de temps afin de spécifier l'instruction de slot de temps et de calculer au plus juste la profondeur de FIFO requise.

Rappel : nous notons L_s la largeur d'un slot en nombre de phits (ou mots) et $|S|$ la taille de la table TDMA en nombre de slots.

De plus, nous définissons les sigles suivants :

- CTS_x : le slot de temps courant dans le routeur émetteur x.
- CTS_y : le slot de temps courant dans le routeur récepteur y.
- $TSI_{x,y}$: l'instruction de slot de temps fournie par le domaine d'horloge x et à exécuter dans le domaine d'horloge y : $0 \leq TSI_{x,y} < |S|$.
- $SkewUp_{x,y}$: limite supérieure du décalage (skew) en nombre de cycles entre les horloges du routeur émetteur x et du routeur récepteur y.
- $SkewLow_{x,y}$: limite inférieure du décalage (skew) en nombre de cycles entre les horloges du routeur émetteur x et du routeur récepteur y.
- $TUp_{x,y}$: temps maximum pour un phit pour traverser le lien (en nombre de cycles).

– $TLow_{x,y}$: temps minimum pour un phit pour traverser le lien (en nombre de cycles). Si cette valeur est inconnue, nous prenons la valeur 1.

Le nombre de cycles maximum de décalage entre les deux routeurs est la somme des temps maximum pour traverser le lien et leur *skew* :

$$CUp_{x,y} = TUp_{x,y} + SkewUp_{x,y} \tag{6.1}$$

Le nombre de cycles minimum de décalage entre les deux routeurs :

$$CLow_{x,y} = TLow_{x,y} + SkewLow_{x,y} \tag{6.2}$$

Le nombre de slots de temps de différence entre les deux routeurs :

$$Delta_{x,y} = \left\lceil \frac{CUp_{x,y}}{L_s} \right\rceil \tag{6.3}$$

L'instruction de slot de temps pour le récepteur y est la somme du slot de temps courant du domaine x et du nombre de slot de différence entre les deux domaines :

$$TSI_{x,y} = (CTS_x + Delta_{x,y})\%|S| \tag{6.4}$$

La profondeur de la FIFO :
si $(CLow < 0)$ alors

$$FDEPTH = |S| * L_s \tag{6.5}$$

sinon

$$FDEPTH = \min(Delta_{x,y} * L_s, |S| * L_s) \tag{6.6}$$

La valeur de $Delta_{x,y}$ peut aussi bien être ajoutée dans le TR que dans le TC. De plus, cette valeur peut être statique et préconfigurée.

6.4 Synchroniseur de TDMAs

6.4.1 Principe

Le synchroniseur de TDMAs s'adresse aux liens entre les sub-NoCs. Il étend le concept de NI à celui d'interface de sub-NoC, en permettant d'interconnecter deux sub-NoCs par un lien, tout en respectant leurs TDMAs respectifs.

L'objectif est de garantir la bande passante et la latence globale en traversant différents sub-NoCs.

Pour illustrer notre concept, nous considérons le NoC de la figure 6.4. Le sous réseau 2 offre une garantie de trafic aux communications qui le traversent.

Pour garantir la bande passante et la latence pour la communication entre A et B, il faut que la bande passante réservée dans chacun des sub-NoCs soit suffisante. De plus, la latence maximale tient compte de l'attente maximale pour accéder aux slots réservés dans la table TDMA de chaque sub-NoC.

Une paire de synchroniseurs, détaillée figure 6.5, est introduite entre les sub-NoCs tel un pont entre les deux domaines d'horloge et de TDMA. Chaque synchroniseur a

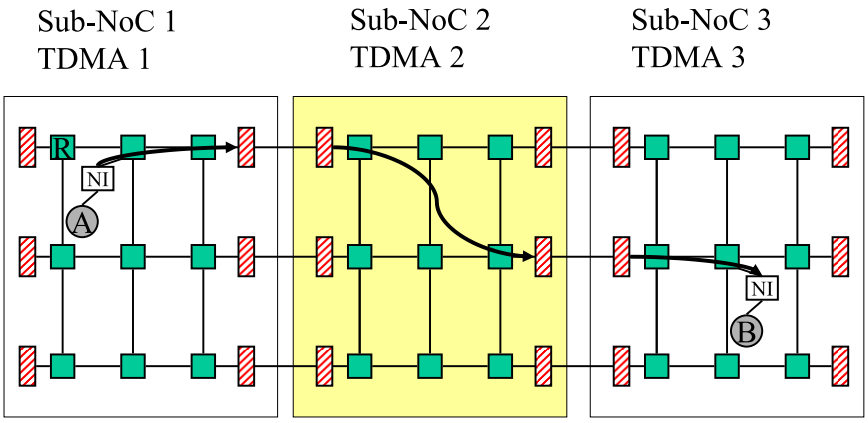


FIG. 6.4 – Un NoC formé de trois sub-NoCs interconnectés

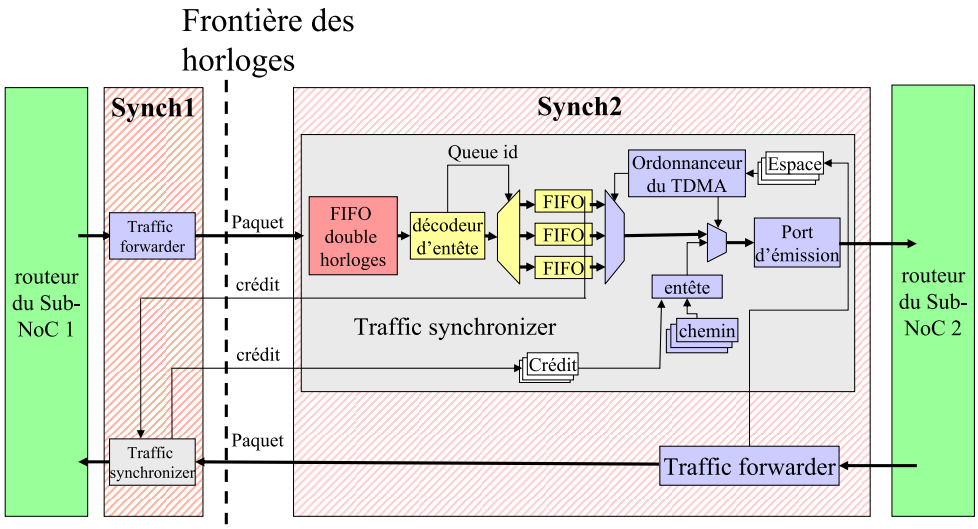


FIG. 6.5 – Une paire de synchroniseurs avec le détail de l'architecture du second

deux parties, la première synchronise le trafic (*traffic synchronizer*) alors que la seconde fait suivre le trafic dans le sens opposé (*traffic forwarder*). Les synchroniseurs sont connectés têtes bêches. La partie *traffic forwarder* du synchroniseur 1 est connectée au *traffic synchronizer* du synchroniseur 2 et vice versa. Seule l'architecture du synchroniseur 2 est détaillée sur la figure 6.5.

Un synchroniseur appartient au sub-NoC vers lequel son module *traffic synchronizer* envoie les communications. Il est vu comme une NI de ce sub-NoC. De même qu'une NI classique, la table de TDMA contenue dans le synchroniseur a été précalculée pour être en accord avec l'ordonnancement de ce sub-NoC. Ainsi, chaque sub-NoC voit un sub-NoC voisin comme une NI classique.

Un paquet quitte un sub-NoC en traversant le module *traffic forwarder* de son synchroniseur. Quand il arrive au synchroniseur distant, il est stocké dans la FIFO double horloge. Le décodeur d'entête utilise l'identifiant de FIFO de l'entête du paquet pour déterminer la FIFO dans laquelle il doit stocker le paquet. Le nombre de FIFOs

et l'architecture du synchroniseur dépend des paramètres du problème et des choix du concepteur qui seront décrits dans les paragraphes suivants.

L'ordonnanceur du TDMA sélectionne la FIFO à lire en fonction des réservations de slots dans la table de TDMA. De plus, en fonction de l'identifiant de la FIFO qui avait été sélectionnée, le champ relatif aux instructions de routage de l'entête du paquet reçoit les instructions de routage appropriées pour traverser le prochain sub-NoC.

La garantie de service de ce trafic est possible si et seulement si la bande passante offerte par les slots réservés pour cette communication dans ce sub-NoC permet de faire passer à la fois le débit de ses données et les entêtes de ses paquets. Nous préciserons ce point dans la section 6.4.4.2.

6.4.2 Le contrôle de flux de bout-en-bout

Pour s'assurer qu'aucun dépassement de capacité ne puisse avoir lieu dans l'interface de réception, un contrôle de flux de bout-en-bout est utilisé. Il repose sur un système de crédit d'émission. Les crédits représentent la quantité de places libres dans la FIFO de réception. À l'initialisation de la connexion entre un émetteur et un récepteur, la totalité de la profondeur de la FIFO de destination est allouée à l'émetteur. Cette FIFO est appelée *round trip buffer*, elle a pour rôle de cacher le délai d'acheminement des données et le délai de retour des crédits. L'émetteur peut uniquement envoyer des données au récepteur dans la limite des crédits dont il dispose. De plus, l'émetteur décrémente cette valeur chaque fois qu'il envoie une donnée vers cette destination.

Le récepteur retourne des crédits d'émission à l'émetteur lorsqu'un nombre suffisant de données a été consommé et donc qu'un nouvel espace libre est disponible dans sa FIFO de réception.

La politique de contrôle de flux peut être introduite à différents niveaux hiérarchiques, soit au niveau global soit au niveau sub-NoC. Le contrôle de flux entre les NIs au travers du NoC complet est appelé global. Le contrôle de flux indépendant dans chaque sub-NoC est local. Le choix entre ces deux politiques dépend de conditions expliquées dans les paragraphe suivants.

6.4.2.1 Le contrôle de flux global, niveau NoC

Au niveau global, le contrôle de flux se fait réellement entre les NIs qui ont instauré une communication, ceci au travers de tout le réseau. Ainsi, plusieurs sub-NoCs peuvent être traversés par cette communication.

Cela signifie que des communications qui partagent un même chemin de part en part d'un sub-NoC sont considérées comme une seule communication du point de vu de ce sub-NoC comme le montre le sub-NoC2 de la figure 6.6. De plus, les modules relatifs au contrôle de flux par crédit d'émission (figure 6.5) sont ôtés des synchroniseurs.

Dans un trafic synchroniseur, une seule FIFO est associée à chaque (sous)destination dans ce sub-NoC, indépendamment de l'émetteur originel de cette communication. Ainsi, le sub-NoC2 de la figure 6.6 regroupe les communications provenant de A et B car elles vont toutes les deux au même synchroniseur. En revanche, le sub-NoC3 place ces deux communications dans des FIFOs différentes car elles vont à des destinations différentes.

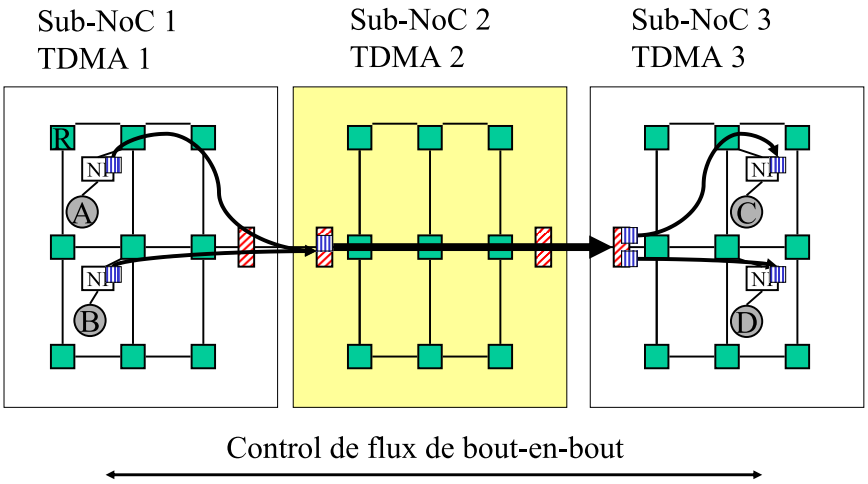


FIG. 6.6 – Contrôle de flux de bout-en-bout au niveau global, niveau NoC

Le contrôle de flux au niveau global, c'est à dire au travers de tout le NoC, peut nécessiter des FIFOs (*round trip buffer*) de grandes profondeurs dans les NIs de destination pour pouvoir masquer le temps d'aller des données et de retour des crédits. Ceci est particulièrement vrai si la distance parcourue par une communication au travers du NoC est longue et parsemée de nombreux sub-NoCs.

Un aspect qui peut être intéressant est celui de la cohérence entre les tables TDMA des sub-NoCs interconnectés. Les tables TDMAs de deux sub-NoCs sont dites cohérentes si les paquets traversant de l'un vers l'autre sont entièrement émis et consommés dans le même ordre. Dans ce cas, une seule FIFO est nécessaire pour stocker les paquets dans la partie *traffic syncronizer* des synchroniseurs au lieu d'une FIFO par destination.

6.4.2.2 Le contrôle de flux au niveau local, niveau sub-NoC

Au niveau sub-NoC, le contrôle de flux se fait localement, à l'intérieur de chacun des sub-NoCs traversés. Les synchroniseurs sont alors considérés comme des NIs à parts entières.

Dans un trafic synchroniseur, une FIFO est assignée non pas à chaque destination de ce sub-NoC comme dans le cas précédent, mais à chaque paire émetteur-destinataire (au niveau du NoC) de cette communication.

Les messages sont dépaquetés quand ils quittent un sub-NoC et rempaquetés pour entrer dans un autre sub-NoC.

Le contrôle de flux local à chaque sub-NoC se traduit par une multitude de petits contrôles de flux (figure 6.7). Les aller-retours des données et crédits sont courts mais nombreux. Le coût en FIFO induit par le contrôle de flux par crédit est distribué sur l'ensemble des TDMAs synchroniseurs traversés, au lieu d'être uniquement dans les NIs de destination. De plus, chaque sub-NoC se suffit à lui même en taille de FIFO, indépendamment de ses voisins. Ainsi, un sub-NoC peut être vu comme une seule et même IP.

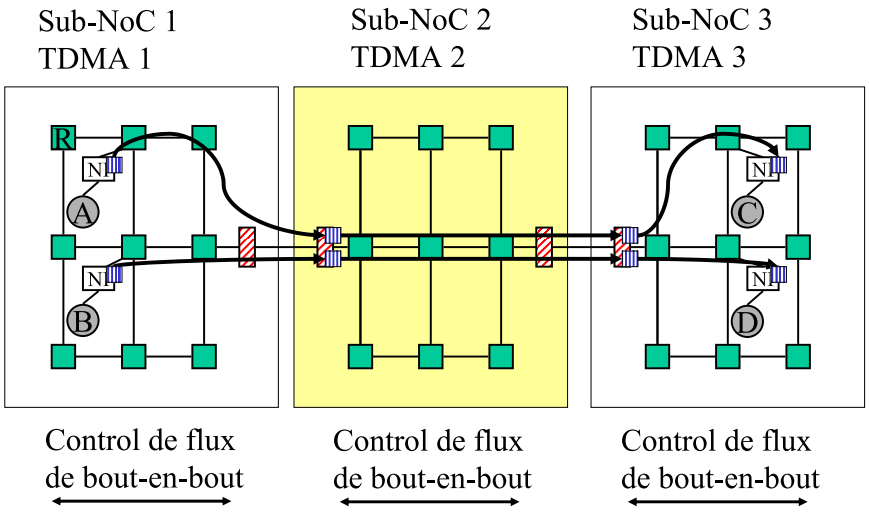


FIG. 6.7 – Contrôle de flux de bout-en-bout au niveau local

6.4.2.3 Comparaison entre les contrôles de flux global et local

Le choix entre les deux types de contrôle de flux dépend de la nature des contraintes. Pour une application donnée, la somme des profondeurs des FIFOs requisent par le contrôle de flux local est égale ou légèrement supérieure à celle du contrôle de flux globale. Cette légère différence peut être causée par la sub-division de la FIFO (*round trip buffer*) dans le cas du contrôle de flux local car la profondeur d'une FIFO étant un nombre entier, par arrondi, cela peut conduire à une légère différence.

Le contrôle de flux local utilise donc des FIFOs plus petites mais plus nombreuses, cela implique donc aussi plus de contrôles (incluant les compteurs de crédits), de plus un problème de la distribution des profondeurs des FIFOs apparaît (une FIFO par paire émetteur-destinataire au lieu d'une seule par destinataire). L'avantage du contrôle de flux local réside dans la subdivision qu'il implique, facilitant la réutilisation et la reconfiguration indépendante des sub-NoCs.

Dans le cas où il y a de nombreuses communications différentes entre deux sub-NoCs, le contrôle de flux global est préférable. Si un faible nombre de communications est spécifié, alors le contrôle de flux local est plus approprié puisque le nombre de FIFOs reste raisonnable. De plus, la nécessité de redimensionner la taille du paquet n'existe que dans le cas du contrôle de flux global comme nous le verrons dans la section 6.4.4.

6.4.3 Instructions de routage

Les instructions de routage indiquent le chemin que le paquet doit suivre pour atteindre sa destination au travers des éléments du NoC. Nous nous plaçons dans le cas d'un routage par la source. Les instructions sont codées dans un des champs de l'entête du paquet.

Le paquet est ici amené à traverser plusieurs sub-NoCs et donc à avoir une longue liste d'instructions, cependant, le champ instruction de l'entête du paquet doit conserver une taille raisonnable. De plus, il est préférable que les chemins au travers de chaque

sub-NoC soient les plus indépendants possible de sorte que l'on puisse opter pour des chemins alternatifs sans en référer au reste du NoC.

Pour toutes ces raisons, nous codons le chemin du paquet sous la forme d'identifiants qui indiquent à un synchroniseur la destination à atteindre dans le prochain sub-NoC à traverser (figure 6.8). Le synchroniseur consulte alors une table de chemin et remplace l'identifiant par les instructions de chemin à suivre pour traverser ce sub-NoC et atteindre soit un autre synchroniseur amenant sur un autre sub-NoC, soit vers la NI destinatrice finale.

De plus, notons que lorsqu'un paquet quitte un sub-NoC, les instructions précédemment utilisées pour le traverser sont effacées, ce qui offre de la place pour insérer les instructions relatives à la traversée du sub-NoC suivant.

Cette distribution de la connaissance du chemin réduit donc la taille du champ d'instruction de routage, permet une plus grande souplesse pour la reconfiguration et permet l'extension, ce qui est donc dans l'esprit du concept de NoC.

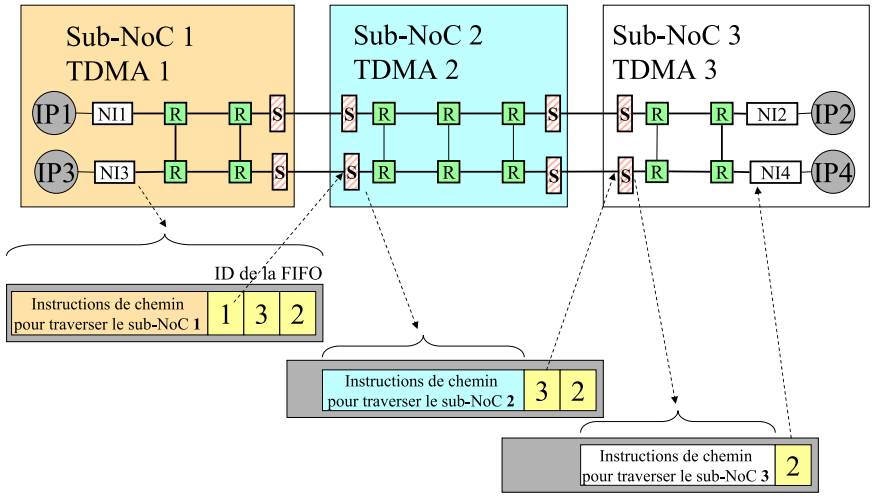


FIG. 6.8 – Les instructions de routage dans l'entête du paquet au travers des sub-NoCs

6.4.4 Redimensionnement de la taille des paquets dans le cas du contrôle de flux global

6.4.4.1 Formulation du problème

Dans le contexte de l'utilisation d'une technique TDMA, le nombre de slots consécutifs réservés (fenêtre d'envoi) pour une communication peut ne pas être le même dans les différents sous-NoCs traversés. Ceci est dû aux tailles différentes des tables de TDMA des sub-NoCs ou encore à la différence des débits de ces sub-NoCs. Nous considérerons ici qu'une communication ne dispose que d'une fenêtre d'envoi par tour de table, ce qui a du sens car cela réduit le coût de l'entête et de plus simplifiera l'explication.

Ceci nous conduit au problème du redimensionnement des paquets, avec des opérations de division et de fusion.

En effet, il peut être nécessaire de diviser, voir à l'inverse de fusionner des paquets pour que leurs tailles soient appropriées à la taille des fenêtres d'envoi. Ceci suppose un coût en contrôle pour réorganiser les paquets.

Quand un paquet doit être divisé en deux parties, son entête est copié pour servir d'entête à la seconde partie. Cette duplication de l'entête occupe de la bande passante. Il faut donc veiller à ce que la bande passante des données transportées soit toujours assurée. La bande passante occupée par les entêtes est d'autant plus importante que les paquet sont petits. La reconstruction ou fusion de paquets appartenant à une même transaction, implique, elle, de supprimer un entête. Cependant les paquets intercalés avec d'autres paquets n'appartenant pas à la même transaction ne peuvent pas être fusionnés, excepté si des FIFOs distinctes sont utilisées pour séparer les paquets appartenant à une même transaction. Ceci parait extrêmement complexe.

Notons que si on est dans le contexte d'un contrôle de flux global, le transport de crédit dans les entêtes implique quelques aménagements. Lors de la division de paquet, l'information de crédit ne doit pas être dupliquée lors de la duplication de l'entête. De même, lors de la fusion, si l'entête à supprimer transporte des crédits, il faut évidemment les sauvegarder et les attribuer au prochain entête. De plus, le dimensionnent de la FIFO (*round trip buffer*) doit avoir pris cet aspect en considération.

Le redimensionnement des paquets peut introduire une dégradation de la bande passante, cela peut être acceptable dans certain cas, cependant le mieux est de pouvoir éviter d'y recourir.

6.4.4.2 Solutions

Considérant une communication allant du sub-NoC$_i$ vers le sub-NoC$_j$, nous comparons dans le tableau 6.1 les tailles des fenêtres d'envois entre ces deux sub-NoC voisins et donnons les règles à respecter.

taille de la fenêtre d'envoi du sub-NoC$_i$ comparée à celle du sub-NoC$_j$	Règles
$>$	La bande passante offerte par le sub-NoC_j doit être suffisamment grande pour supporter la bande passante offerte par le sub-NoC_i et pour pouvoir transporter le ou les entêtes supplémentaires introduits par la division du paquet
$= 1$	La bande passante offerte par le sub-NoC$_j$ doit être au moins égale à la bande passante offerte par le sub-NoC$_i$
$\leq$	Le paquet ne peut débuter son émission que durant le premier slot de la fenêtre d'envoi. On s'assure ainsi qu'il ne sera pas divisé. La bande passante offerte par le sub-NoC$_j$ doit être au moins égale à la bande passante offerte par le sub-NoC$_i$

TAB. 6.1 – Taille des fenêtres d'envoi et règles

Une solution simple pour obtenir le même nombre de slots réservés et la même bande passante pour les communications dans chaque sub-NoC consiste à conserver le même rapport fréquence (f_{NoC_i}) sur taille de table TDMA (S_{NoC_i}). La figure 6.9

montre un exemple avec deux sub-NoCs de tailles de tables TDMAs différentes et fonctionnant à des fréquences différentes.

En effet la bande passante du lien est

$$B_L = L_p * f_{NoC} \tag{6.7}$$

et la bande passante d'un slot est

$$B_S = \frac{B_L}{|S|} \tag{6.8}$$

ainsi, la bande passante d'un slot réservé dans un NoC_i est

$$B_{S_{NoC_i}} = \frac{L_p * f_{NoC_i}}{|S_{NoC_i}|} \tag{6.9}$$

et la bande passante d'un slot réservé dans un NoC_j est

$$B_{S_{NoC_j}} = \frac{L_p * f_{NoC_j}}{|S_{NoC_j}|} \tag{6.10}$$

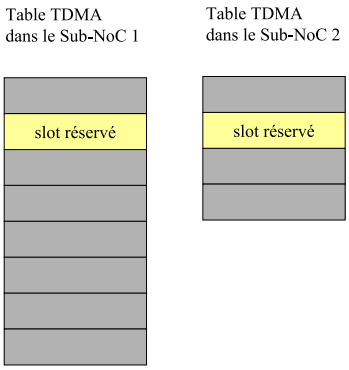
donc,

$$B_{S_{NoC_i}} = B_{S_{NoC_j}} \tag{6.11}$$

si

$$\frac{f_{NoC_i}}{|S_{NoC_i}|} = \frac{f_{NoC_j}}{|S_{NoC_j}|}$$

ainsi, les bandes passantes d'un slot dans chacun de ces deux sub-NoCs sont égales si ces sub-NoCs ont le même rapport fréquence sur taille de table TDMA.



	Sub-NoC 1	Sub-NoC 2
Fréquence des slots	100MHz	50MHz
Taille de la table de TDMA	8	4
Fréquence du slot réservé = (fréquence des slots/ Taille de la table de TDMA)	12.5MHz	12.5MHz

FIG. 6.9 – Le rapport fréquence sur taille de la table TDMA

6.5 Application et résultats

6.5.1 Contexte de l'application

Nous considérons une application de télécommunication de $4^{ème}$ génération. Il s'agit d'un codeur/décodeur intégrant une technique de communication MC-CDMA et MC-SS-MA en bande de base. La contrainte de l'application est une cadence de 665,6μs pour traiter une *frame* composée de 32 symboles.

Le codeur MC-SS-MA requière 18 Ports d'IP et le décodeur MC-CD-MA nécessite lui 32 ports d'IP. Les ports d'IP sont regroupés en 22 *clusters*. Chacun de ces *clusters* est connecté à une NI du réseau.

Cette application requière 29 transactions sous forme de communications unidirectionnelles entre les éléments matériels. Nous utilisons un contrôle de flux de bout-en-bout, le retour des crédits d'émission rend donc les communications bidirectionnelles.

Les bandes passantes requises sont très hétérogènes. En effet, les communications proches de l'antenne dans le graphe de l'application, qui sont donc les données encodées, nécessitent une bande passante très supérieure à la bande passante requise pour les données décodées ou pas encore codées.

Cette application est architecturée sur une topologie composée de deux parties que nous appellerons zone 1 et zone 2. Cette division est faite de telle sorte que les communications nécessitant une faible bande passante sont regroupées dans la zone 1 alors que celles nécessitant une plus grande bande passante sont regroupées dans la zone 2. Trois communications sont échangées entre les zones 1 et 2, elles seront appelées communications inter-zones. Celles-ci représentent donc 10% des communications totales. Cette proportion semble représentative des communications que nous pouvons espérer dans les futures systèmes sur puce dans la mesure où nous devrions avoir une grosse proportion de communications locales, si le placement des ports des IPs est bien réalisé.

Sur la figure 6.10, nous pouvons voir la topologie de notre exemple. Les zones 1 et 2 seront considérées comme formant un NoC unique ou bien comme deux sub-NoC selon les cas d'études présentés ci-après. Pour des raisons de clarté, seules les communications inter-zones sont représentées.

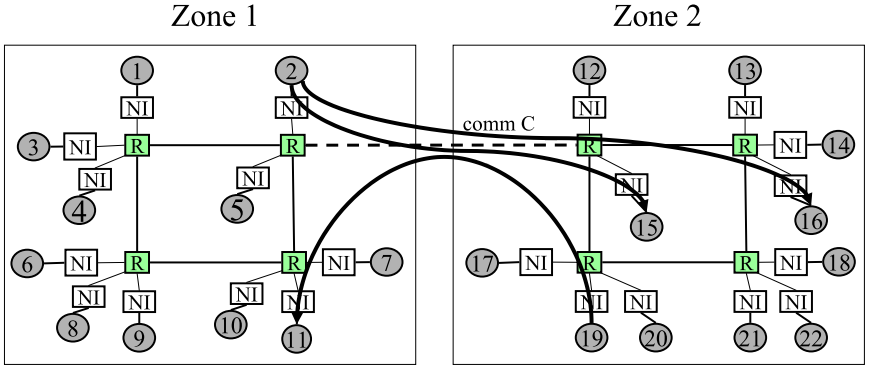


FIG. 6.10 – Topologie et communications inter-zones de l'application

Les caractéristiques générales du NoC sont ici : un lien ou phit a un largeur de 32 bits, la taille d'un flit est de un phit, et la taille d'un slot est de deux phits, enfin, l'entête du paquet a une largeur d'un phit.

6.5.2 Description des cas d'études

Nous allons étudier plusieurs cas reflétant différentes contraintes.

Cas 1 : l'implantation est celle d'un NoC unique classique, sans aucun problème de lien long, ni de décalage d'horloge. Ce cas correspond à un NoC classique, il nous servira de référence pour nos comparaisons.

Dans les cas suivants (de 2 à 5), nous supposons avoir un délai maximum de 50ns entre les deux zones.

Cas 2 : un NoC unique utilisant une interface de type routage temporel pour faire face au problème du lien long entre les deux zones.

Pour les 3 cas qui suivent, les zones 1 et 2 ont des domaines d'horloge différents (9 et 100 MHz respectivement). Ils sont intégrés en tant que sub-NoC1 et sub-NoC2. La taille de la table de TDMA du sub-NoC1 n'est plus de 6 slots mais de 4 slots ; les autres paramètres restent inchangés. Deux synchroniseurs sont introduits sur le lien entre les deux sub-NoCs. Ces trois cas correspondent aux différentes solutions d'implémentation présentées précédemment :
Cas 3 : deux sub-NoCs avec des horloges hétérogènes ; On utilise le contrôle de flux de bout-en-bout global. De plus, les TDMAs sont cohérents.
Cas 4 : deux sub-NoCs avec des horloges hétérogènes ; On utilise le contrôle de flux de bout-en-bout global. Les TDMAs ne sont pas cohérents.
Cas 5 : deux sub-NoCs avec des horloges hétérogènes ; On utilise le contrôle de flux de bout-en-bout local.

Les cas considérés sont résumés dans la table 6.2.

	Zone 1		Zone 2			
Cas	F MHz	$\|S\|$	F MHz	$\|S\|$	Commentaires	Solutions
1	100	6	100	6	Les tailles des tables TDMAs sont identiques. Le délai sur le lien est de moins d'un cycle.	Classique
2	100	6	100	6	Les tailles des tables TDMAs sont identiques. Le délai sur le lien est de 5 cycles.	Interfaces de routage temporel
3	9	4	100	6	Les tailles des tables TDMAs sont différentes. Le délai sur le lien est de 5 cycles.	Synchroniseurs avec contrôle de flux global et TDMAs cohérents
4	9	4	100	6	Les tailles des tables TDMAs sont différentes. Le délai sur le lien est de 5 cycles.	Synchroniseurs avec contrôle de flux global
5	9	4	100	6	Les tailles des tables TDMAs sont différentes. Le délai sur le lien est de 5 cycles.	Synchroniseurs avec contrôle de flux local

TAB. 6.2 – Les cas d'étude

6.5.3 Analyse des résultats

Le tableau 6.3 montre les coûts relatifs aux différents cas étudiés.

Dans le cas 1, la somme des profondeurs des FIFOs est due à la somme des FIFO de découplage et de *round trip buffer* dans les NIs. La latence pour la communication C de la figure 6.10 est de 200ns.

Le délai introduit sur le lien dans le cas 2 augmente la latence de 2 slots. En effet la durée normale aurait été de 1 slot, seulement ici comme il faut 5 cycles au lieu d'un

	cas 1	cas 2	cas 3	cas 4	cas 5
Nombre de FIFOs dans les interfaces	0	2	2	3	3
Somme des profondeurs des FIFOs	323	335	348	348	348
Augmentation de la taille des FIFOs pour les 3 communications inter-zones	0	12	18	18	18
Latence pour la communication notée C (ns)	200	240	1938	1938	1938

TAB. 6.3 – Résultats de latence et coût en taille de FIFO

pour traverser le lien, 3 slots sont alors nécessaires. L'augmentation de la taille des FIFOs a deux raisons :

- La FIFO dans les TRs (12 mots) : deux routeurs temporels sont ajoutés (un dans chaque sens). Chacun d'entre eux possède une FIFO. Et la profondeur de cette FIFO est constante (6 mots) et ne dépend pas des communications inter-zones. Remarque : le coût en FIFO entre deux routeurs était déjà de 2 mots pour chaque sens.
- La profondeur des FIFOs de *round trip buffer* dans les NIs dépend de la longueur des chemins des communications inter-zones (2 mots).

Dans les cas 3 à 5, la réduction de fréquence dans le sub-NoC1 implique une augmentation de latence des communications qui le traversent (+1578ns). L'augmentation de la latence est aussi due à la traversée des synchroniseurs pour les communications inter-zones.

Le cas 3 bénéficie de TDMAs cohérents entre les deux sub-NoCs, ainsi le synchroniseur utilise une FIFO unique.

Le cas 4 met en œuvre des TDMAs non cohérents, avec un contrôle de flux au niveau global. Il nécessite plus de FIFO (une FIFO par destination), cependant, la somme des profondeurs des FIFOs reste inchangée.

Le cas 5 est similaire au précédant si ce n'est qu'il utilise un contrôle de flux au niveau local.

Pour conclure, nous observons que la mise en œuvre de ces solutions a un coût acceptable dans ces quatre cas, comparé au cas de référence.

De plus, nous observons que la réduction de fréquence implique une augmentation de la surface du fait de l'augmentation des tailles des FIFOs pour s'adapter à l'augmentation de la latence des communications. Une étude précise est donc nécessaire pour voir si la diminution de la consommation dynamique est vraiment intéressante comparée à l'augmentation de la consommation statique due à l'augmentation de la surface.

6.6 Conclusion

Nous avons présenté des solutions originales pour garantir la QoS dans le contexte d'un NoCs constitué d'horloges avec des déphasages, ou des horloges de fréquence différentes (ou TDMA différents). Nous avons proposé une solution pour chacun de ces deux problèmes (le routage temporel et le synchroniseur de TDMA), incluant dans le dernier cas le concept de composition de sub-NoCs. Cette approche va dans le sens de la productivité pour la conception puisqu'elle est compatible avec la méthodologie

des NoCs synchrones classiques et peut être facilement intégrée dans le flot de conception. L'expérimentation nous montre que le coût de ces solutions est acceptable. De plus, nous avons présenté une technique pour traiter les instructions de routage au travers de plusieurs sous réseaux. Cette solution favorise la reconfiguration et permet l'extensibilité. Enfin, l'utilisation de sub-NoCs , avec des TDMA disjoints et indépendants, rend l'intégration du contrôleur de NoC plus facile pour introduire un contrôle dynamique des tensions et fréquences dans le but de réduire la consommation ainsi que pour gérer des aspects sécurité.

Chapitre 7

Expérimentations et résultats

7.1 Introduction

Ce chapitre présente premièrement une mise en œuvre réelle du NoC sur une plate-forme FPGA pour connecter 3 microprocesseurs MicroBlazes et leurs RAMs.

Ensuite, nous exposons les résultats obtenus pour plusieurs applications. Le choix des applications a été guidé par l'expérience des équipes des différents projets auquels nous avons participé et permet de valider nos méthodes dans les domaines variés que sont les domaines du traitement du signal, de l'image et des télécommunications. Nous verrons que les contraintes en terme de communication ne sont pas les mêmes pour ces différentes applications.

Enfin, nous avons réalisé des intégrations comportant des combinaisons de ces applications afin de montrer l'intérêt de traiter les exclusions mutuelles.

7.2 Intégration sur FPGA

7.2.1 Conditions du test

Afin de vérifier le bon fonctionnement du NoC sur une plate-forme matérielle, nous avons réalisé un exemple mettant en œuvre des communications entre 3 MicroBlazes et 3 mémoires. C'est un exemple suffisamment petit pour être intégré sur le FPGA dont nous disposons. Il nous permet d'observer les caractéristiques et les performances du NoC.

7.2.1.1 Environnement

Nous disposons du matériel et des logiciels suivants afin de réaliser cette expérience :

- Une plate-forme de prototypage Xilinx Virtex-II Pro FF1152 PROTO BOARD équipée d'un FPGA Xilinx Virtex-II Pro FF1152 VP50-5.
- Les logiciels Xilinx ISE 6.3 SP3 et Xilinx EDK 6.3 SP2.

7.2.1.2 Présentation

Le NoC que nous utilisons se compose des éléments suivants :

- 2 routeurs avec 3 ports réseau (R3ports) ;

– 2 routeurs avec 4 ports réseau (R4ports) ;
– 3 NIs avec 1 port et 2 ExtraNiChannels (NI2Ch) ;
– 3 NIs avec 1 port et 4 ExtraNiChannels (NI4Ch).

De plus, 3 *wrappers* maîtres et 3 *wrappers* esclaves sont utilisés pour connecter les trois bus OPB.

La figure 7.1 montre le NoC connecté aux trois bus OPB et les connections à créer entre les NIs. Nous testons à la fois des transactions entre composants maîtres et esclaves, mais aussi les échanges de messages entre des composants maîtres.

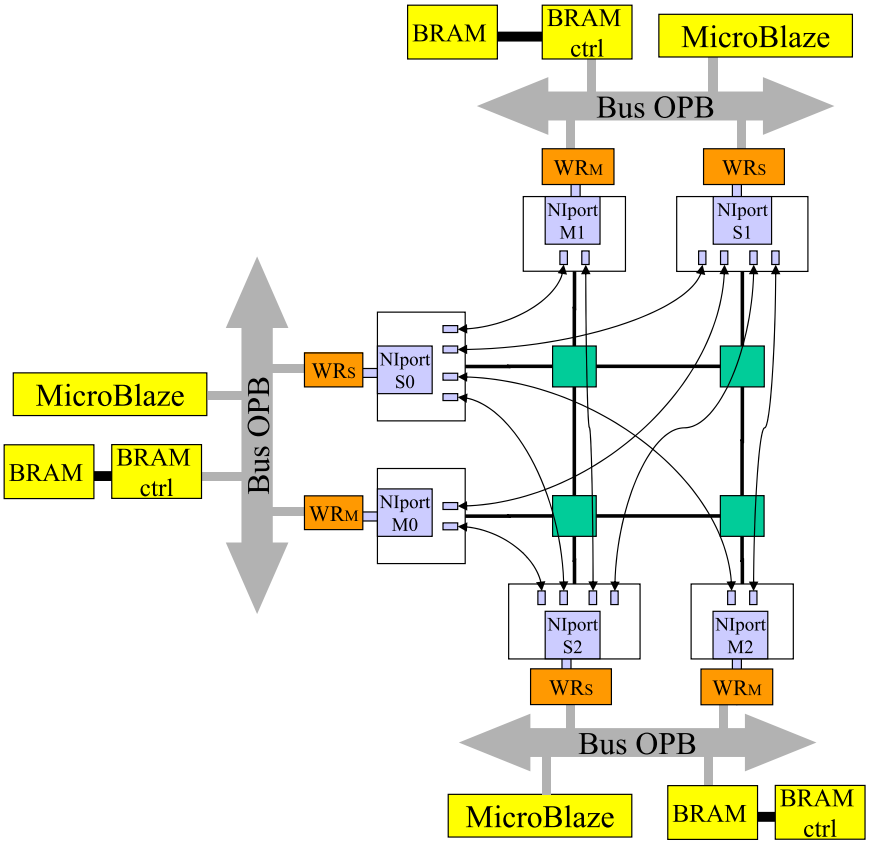


FIG. 7.1 – Le NoC et les connections à créer entre les NIs

Nous exécutons des programmes sur chacun des 3 MicroBlazes dans le but de les faire s'échanger des données au travers du NoC afin de vérifier son bon fonctionnement. De plus, ces programmes vérifient des règles de cohérence qui permettent de détecter l'ordre, la validité et la perte d'un message.

Un seul MicroBlaze peut communiquer via la liaison série de la plate-forme qui nous permet d'obtenir un affichage à l'écran. Ce processeur affiche à l'écran les informations concernant le fonctionnement des échanges.

Nous testons les cinq cas suivants :
– BE : Un canal virtuel configuré pour transporter du trafic *best effort* ;
– GT : Un canal virtuel configuré pour transporter du trafic garanti, il utilise une table TDMA de 16 slots ;
– GT&BE : Deux canaux virtuels. Le premier est un BE le second un GT ;

– BE&BE : Deux canaux virtuels BE. Le trafic sur le premier est plus prioritaire que sur le second ;
– BE8 : Un canal virtuel configuré pour transporter du trafic *best effort*. La taille des FIFOs dans les ports réseaux des routeurs et des NIs est de 8 au lieu de 2.

Nous utilisons une largeur de données de 32 bits, un routage *street-sign*, un arbitrage *round-robin* et un contrôle de flux de bout-en-bout entre les NIs. La taille des FIFOs dans les ExtraNIchannels est de 8. La taille des FIFOs dans les ports réseaux des routeurs et des NIs est de 2 mots, sauf dans le dernier cas ou elle est de 8.

7.2.2 Résultats de synthèse

Notre modèle de description en XML permet de spécifier le NoC voulu facilement et rapidement. Les codes VHDL des NoCs correspondants aux cinq cas présentés précédemment ont été automatiquement générés par notre outil en moins de 2 secondes. De plus, l'intégration d'un NoC dans l'environnement EDK est facilité. En effet, les fichiers VHDL ainsi que des fichiers spécifiques sont créés de manière à permettre à l'utilisateur d'ajouter le NoC en tant que simple bus par un simple copier-coller de ces fichiers dans la bibliothèque appropriée. Ceci met en évidence le gain en productivité obtenu par l'utilisation de notre outil de conception.

La synthèse du NoC complet et de ses composants considérés indépendamment nous donne la répartition en *slices* de l'utilisation du FPGA. La capacité totale du FPGA est de 23616 *slices*. Le tableau 7.1 nous montre cette répartition en *slices*, ainsi que la fréquence maximale en MHz.

Les *wrappers* conservent les mêmes caractéristiques dans ces différents cas. La taille du NoC est donnée sans les wrappers.

– Pour le *wrapper* maître W_S, la surface occupée est 1064 *slices* et la fréquence est de 105MHz.
– Pour le *wrapper* esclave W_S, la surface occupée est 90 *slices* et la fréquence est de 144MHz.

		NoC	NI2Ch	NI4Ch	R3ports	R4ports
Cas 1	Slices	7204	607	1167	430	645
BE	Freq.	94	120	118	136	136
Cas 2	Slices	7226	624	1195	430	643
GT	Freq.	84	120	118	136	136
Cas 3	Slices	9251	630	1181	851	1317
BE & GT	Freq.	85	93	90	118	118
Cas 4	Slices	9205	621	1170	851	1317
BE & BE	Freq.	84	93	90	118	118
Cas 5	Slices	7488	622	1203	443	662
BE8	Freq.	100	120	114	132	132

TAB. 7.1 – Synthèse du NoC et de ses composants

Nous pouvons voir que le NoC occupe entre 30 et 40 % de la surface du FPGA. Un routeur occupe entre 1 et 5% des *slices* du FPGA. Une NI occupe entre 2 et 4% des *slices* du FPGA. La surface occupée par le NoC reste donc raisonnable.

La fréquence du NoC dépasse les 80 MHz. Ce qui permet sur un lien une bande passante théorique de $32 * 80.10^6 = 2560.10^6 bits/s$ soit 320Mo/s.

A taille égale des FIFOs, nous pouvons voir que les NIs occupent une plus grande surface lorsqu'elles intègrent un VC GT qu'un VC BE. Cela est dû à la table des slots TDMA et au mécanisme d'ordonnancement.

Dans les cas 3 et 4, l'utilisation de deux canaux virtuels à un coût important en surface dans les routeurs, ils doublent presque de surface. En revanche les NIs ne voient leur surface augmenter que légèrement.

Dans le cas 5, l'augmentation légère (de 2 à 8) de la taille des FIFOs dans les ports réseau des routeurs et des NIs cause une augmentation de surface du NoC de 4%.

7.2.3 Résultats du test sur la plate-forme

Les tests sur la plate-forme ont été réalisés avec une horloge d'une fréquence de 50MHz.

Les placements routages des NoCs présentés précédemment se sont bien déroulés dans quatre des cas, en revanche nous avons rencontré un problème pour réaliser un NoC utilisant simultanément un canal virtuel GT et un canal virtuel BE. La synthèse se passe normalement avec ISE, mais lors de la phase de placement routage, l'outil indique un message d'erreur. Il s'agit d'une erreur interne de l'outil de placement routage dont Xilinx nous dit qu'elle sera corrigée dans les versions futures. Ce genre d'aléa est hélas courant et connu des concepteurs. En considèrent les éléments du NoC séparément, nous avons constaté que ce problème intervient uniquement dans les NIs et que lorsqu'elles utilisent des canaux virtuels GT et BE simultanément. Nous n'avons pas pu mener l'expérience sur la plate-forme pour le cas 3. Des simulations sous ModelSim nous ont tout de même montrés que notre code VHDL et le comportement obtenu sont corrects.

Pour les quatre autres cas, les tests des NoCs ont été effectués sur la plate-forme avec les 3 MicroBlazes et nous ont montrés que les données sont bien transmises et que le contrôle de flux de bout-en-bout fonctionne puisqu'aucune donnée n'est perdue.

L'utilisation du canal virtuel par un trafic garanti utilisant le TDMA a également été validée.

Nous avons vérifié que la priorité entre les canaux virtuels fonctionne dans le cas 4 qui utilise simultanément deux canaux virtuels BE. En effet le trafic transporté sur le premier canal virtuel est bien transmis en priorité par rapport au trafic transporté sur le second canal virtuel.

7.2.4 Conclusion

Ce test a permis de valider notre approche de façon complète, c'est à dire en intégrant le NoC en tant que IP, les *wrappers* pour le connecter à des bus OPB et la couche logicielle dans les MicroBlazes.

Nous avons observé que la surface du NoC est raisonnable (30 à 40% des *slices* du FPGA). De plus, dans cet exemple, nous avons mis en œuvre tous les types de communications (passage de message, transaction de lecture et d'écriture). Le NoC pourrait être réduit pour n'utiliser que le passage de message (pas de *wrapper* maître), ou au contraire que les transactions (*wrapper* esclave avec moitié moins de connections).

7.3 Traitement d'image

7.3.1 Présentation

L'application que nous considérons a été développée dans le cadre du projet Epicure [88] dans lequel l'équipe « Codesign » du laboratoire LESTER intervenait pour réaliser l'exploration de l'espace de conception avec l'outil Design Trotter.

Cette application est un système embarqué de caméra intelligente. Ce dispositif réalise la détection d'objets en mouvement dans des images capturées par une camera.

Ainsi, cette application intègre différentes tâches de traitement, incluant l'acquisition vidéo, le filtrage d'image, l'extraction des objets et du fond, le déplacement des objets et différents contrôles liés à l'affichage (figure 7.2).

Pour notre NoC, nous avons choisi d'utiliser une topologie en grille 2D de 3x4. Nous utilisons les ports situés sur les bords extérieurs de la grille. La figure 7.3 montre la topologie et le placement des composants avec les NIs. Les *wrappers* ne sont pas représentés.

L'application requière 19 transactions de lecture/écritures. La résolution de l'image est de 320x240 pixels et la contrainte applicative à respecter est une cadence de 25 images par secondes.

Par la technique de dérivation des contraintes, nous avons extrait depuis les contraintes de l'application, les contraintes de bande passante et de latence des communications (figure 7.4).

Certaines des communications n'ont jamais lieu en même temps. Ainsi, nous avons défini des communications mutuellement exclusives.

Notre objectif est ici de déterminer la configuration pour réaliser ces communications par un trafic GT. Nous désirons évaluer différentes configurations afin d'observer l'impact du contrôle de flux de bout-en-bout et de la prise en compte des exclusions mutuelles sur la taille de la table TDMA et le coût en FIFOs dans l'architecture.

7.3.2 Résultats

Ce problème complexe a été résolu à l'aide de notre outil de décision. Les résultats obtenus sont représentés dans le tableau 7.2.

Options choisies	Nombre de slots de la table TDMA	Somme de la profondeur des FIFOs dans les NIs
Avec contrôle de flux de bout-en-bout, Avec prise en compte des exclusions mutuelles	5	60
Avec contrôle de flux de bout-en-bout, Sans prise en compte des exclusions mutuelles	11	65
Sans contrôle de flux de bout-en-bout, Sans prise en compte des exclusions mutuelles	8	50
Sans contrôle de flux de bout-en-bout, Avec prise en compte des exclusions mutuelles	5	50

TAB. 7.2 – Résultats

Nous observons qu'en fonction des options choisies, l'outil est parvenu à des solutions avec des tailles de tables TDMA différentes. Nous observons que l'utilisation du contrôle de flux de bout-en-bout implique un coût supplémentaire en profondeur de

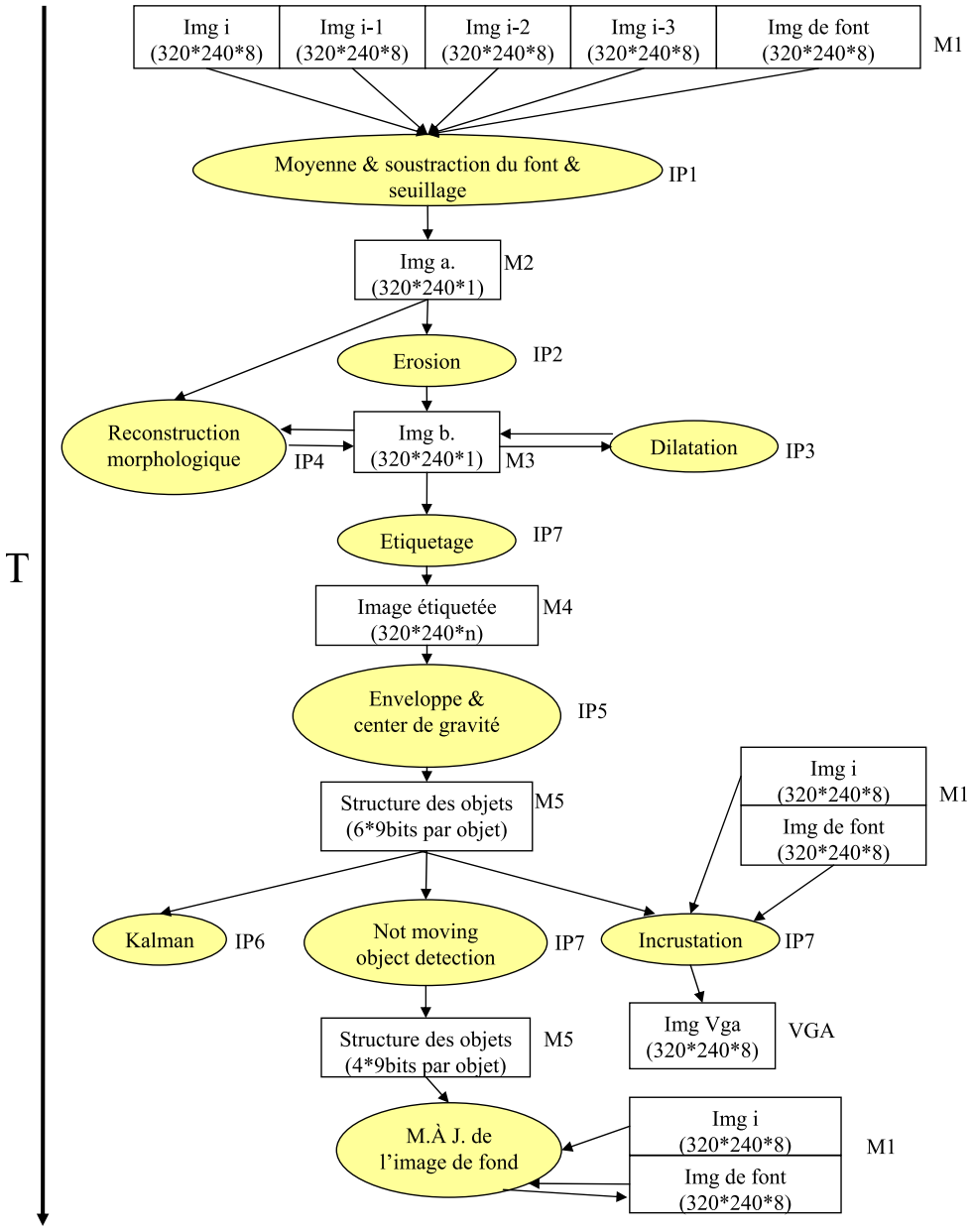


FIG. 7.2 – L'application caméra intelligente

FIFO. L'utilisation des exclusions mutuelles permet d'obtenir une solution avec une taille de table de TDMA réduite ce qui peut permettre de réduire le coût en FIFO.

7.3.3 Conclusion

Nous voyons qu'un NoC peut être utilisé pour satisfaire les besoins de communication de l'application. De plus, nous voyons que notre outil de conception automatique est parvenu à trouver des solutions pour les différentes configurations. Ces solutions offrent des tailles de tables de TDMA petites et des coûts en FIFO raisonnables.

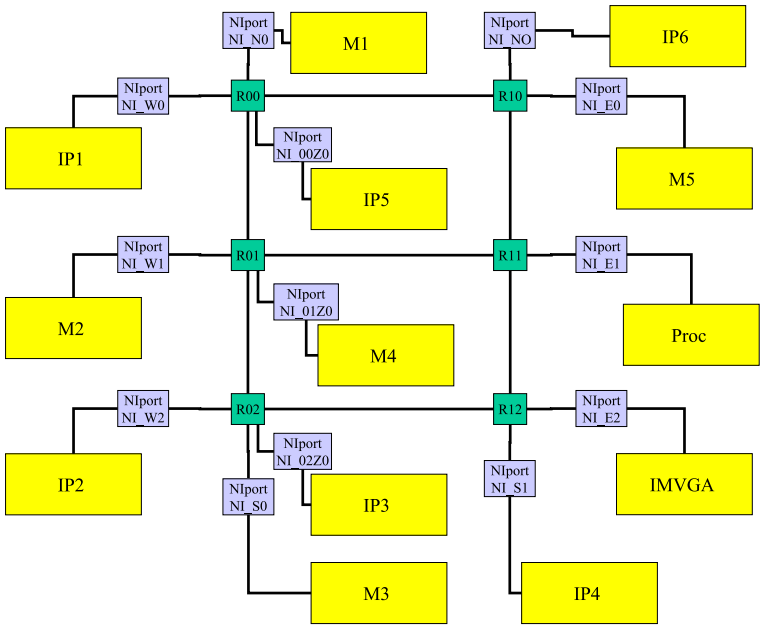


FIG. 7.3 – La topologie de l'application de traitement d'image

		READ			WRITE			
Initiator port	Target port	Bandwidth (MBytes/s)	BurstSize (Bytes)	Latency (nano sec)	Bandwidth (MBytes/s)	BurstSize (Bytes)	Latency (nano sec)	QoS
IP1	M1	15,36	5	40000000	0	0	40000000	gt
IP1	M2	0	0	40000000	0,24	4	40000000	gt
IP2	M2	0,24	4	40000000	0	0	40000000	gt
IP2	M3	0	0	40000000	0,24	4	40000000	gt
IP3	M3	0,24	4	40000000	0,24	4	40000000	gt
IP4	M2	0,24	4	40000000	0	0	40000000	gt
IP4	M3	0,24	4	40000000	0,24	4	40000000	gt
IP5	M4	1,92	4	40000000	0	0	40000000	gt
IP5	M5	0	0	40000000	0,0768	4	40000000	gt
IP6	M5	0,0768	4	40000000	0	0	40000000	gt
PROC	M3	0,24	4	40000000	0	0	40000000	gt
PROC	M4	0	0	40000000	19,2	8	40000000	gt
PROC	M5	0,0768	4	40000000	0,0768	4	40000000	gt
PROC	M1	3,84	4	40000000	1,92	8	40000000	gt
PROC	IMVGA	0	0	40000000	1,92	8	40000000	gt

FIG. 7.4 – Les contraintes de bande passante et de latence des communications

Dans le cadre du projet RaaR [89] au LESTER, cette application de traitement d'image est utilisée pour effectuer un partitionnement logiciel/matériel dynamique contrôlé par un RTOS. L'idée est de réguler automatiquement l'utilisation des ressources en fonction des caractéristiques de l'image (nombre d'objets, bruit, etc) sous contraintes de QoS et d'énergie.

Dans ce contexte, nous voyons qu'un NoC programmable est utile afin d'adapter les communications aux choix effectués en termes d'implantation logicielle/matérielle des différentes tâches et des paramètres de ces tâches.

7.4 Chaîne MC-CDMA MC-SS-MA

7.4.1 Présentation

Cette application est une chaîne MC-CDMA (*Multi-Carrier Code Division Multiple Access*)[90] d'un terminal mobile dans le cadre du futur standard de radio télécommunications de quatrième génération (4G). Son étude constitue une part du projet

européen 4MORE [1] dans lequel le laboratoire IETR est impliqué. La 4G nécessitera un système flexible et capable de supporter différents standards pour permettre l'évolution et la mise à jour du SoC. Ces contraintes font du NoC un bon candidat.

La contrainte de cadence est de 1 symbole OFDM toutes les 20.8µs. Une trame est composée de 32 symboles. La contrainte applicative est donc de tenir la cadence d'une trame tous les 665,6µs. Les figures 7.5 et 7.6 issues de [91], nous montrent les diagrammes du transmetteur (TX) et du récepteur (RX). Les contraintes des communications relatives à la partie transmission et réception nous ont été données et sont représentées dans les tableaux 7.3 et 7.4.

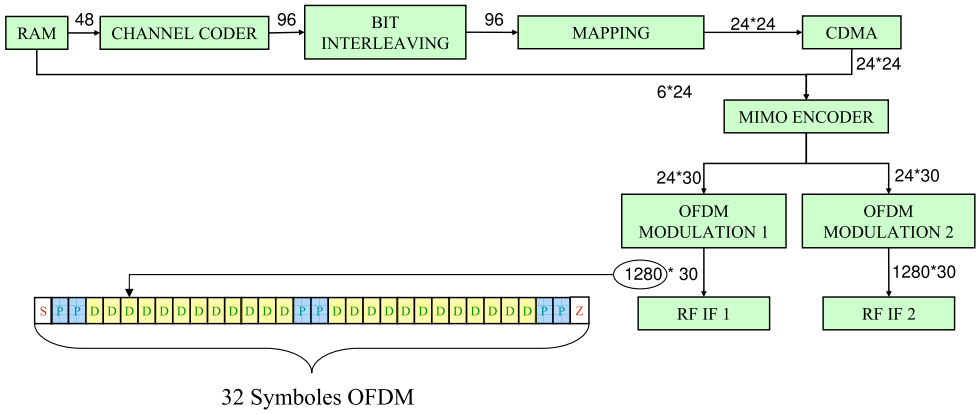


Fig. 7.5 – Diagramme du transmeteur MC-SS DMA

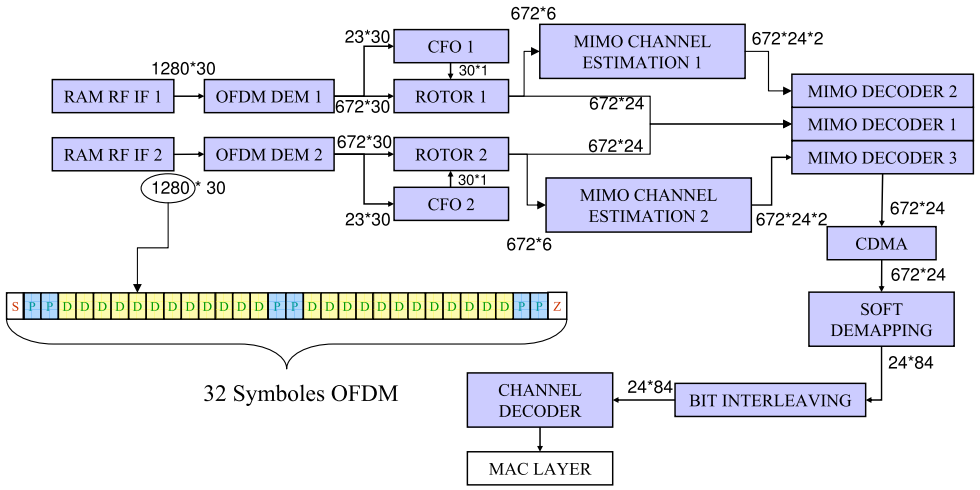


Fig. 7.6 – Diagramme du récepteur MC-CDMA

29 communications unidirectionnelles sont mises en jeu entre les différents éléments du système. Nous pouvons voir que les bandes passantes requises sont variées. Le trafic est prédictible, il est donc tout à fait adapté à l'utilisation du trafic garanti par TDMA.

Cependant le flot de données est relativement point à point et orienté. Nous avons donc utilisé un placement des composants permettant de tirer profit des liens dans chaque direction afin d'utiliser le NoC au mieux. Le port d'entrée et le port de sortie

de chaque composant ont été regroupés en *cluster* sur une même NI. Nous avons ainsi 28 clusters.

Nous n'avons pas cherché le placement optimal des composants, mais à valider notre flot dans le contexte d'une application de télécommunication qui présente des caractéristiques différentes de celles de l'application de traitement d'image.

Initiator port	Target port	BW(Bytes/s)	Latency (nano sec)
RAM 1	Channel coder	288462	665600
RAM 1	MIMO encoder	865385	665600
Channel coder	Bit interleaving	576923	665600
Bit interleaving	Mapping	576923	665600
Mapping	Spreading(CDMA)	3461538	665600
Spreading(CDMA)	MIMO encoder	3461538	665600
MIMO encoder	FFT 1	4326923	665600
MIMO encoder	FFT 2	4326923	665600
FFT 1	BB to RF 1	230769231	665600
FFT 2	BB to RF 2	230769231	665600

TAB. 7.3 – Les communications du transmetteur (TX)

Initiator port	Target port	BW(Bytes/s)	Latency (nano sec)
RAM 2 (RF to Baseband)	iFFT 1	230769231	665600
RAM 3 (RF to Baseband)	iFFT 2	230769231	665600
iFFT 1	CFO 1	4146635	665600
iFFT 2	CFO 2	4146635	665600
iFFT 1	ROTOR 1	121153846	665600
iFFT 2	ROTOR 2	121153846	665600
CFO 1	ROTOR 1	180288	665600
CFO 2	ROTOR 2	180288	665600
ROTOR 1	MIMO channel estimation 1	24230769	665600
ROTOR 2	MIMO channel estimation 2	24230769	665600
ROTOR 1	MIMO decoder 1	96923077	665600
ROTOR 2	MIMO decoder 1	96923077	665600
MIMO channel estimation 1	MIMO decoder 2	193846154	665600
MIMO channel estimation 2	MIMO decoder 3	193846154	665600
MIMO decoder 1	De-Spreading(CDMA)	121153846	665600
De-Spreading(CDMA)	Soft De-Mapping	12115385	665600
Soft De-Mapping	Bit De-interleaving	12115385	665600
Bit De-interleaving	Channel Decoder	12115385	665600
Channel Decoder	RAM 2	1514423	665600

TAB. 7.4 – Les communications du récepteur (RX)

7.4.2 Résultats

Nous avons choisi d'utiliser le trafic de classe GT et le contrôle de flux de bout-en-bout. Nous n'utilisons pas ici les mutuelles exclusions qui feront l'objet de la section 7.6. La topologie du NoC choisie est une grille 4x4 avec une connexion locale et nous utilisons également les bords extérieurs. Nous utilisons ainsi 16 routeurs et connectons 28 NIs. La largeur des mots sur les liens est de 32 bits et la fréquence est de 100MHz.

L'outil μSpider a trouvé une solution avec un taille de table TDMA de 6 slots de temps. La profondeur cumulée des FIFOs nécessaires est de 238 mots.

7.4.3 Conclusion

Notre solution utilise des TDMAs pour garantir les caractéristiques de bande passante et de latence des communications. Cette solution a un faible coût puisque les routeurs ne nécessitent pas de grandes tailles de FIFO pour traiter les conflits d'accès puisque les communications sont préordonnancées par le TDMA.

De plus, cette solution est moins sensible aux variations de comportement des éléments de traitement qu'une solution construite sur mesure qui s'appuierait uniquement sur des simulations pour s'assurer que les communications s'agencent correctement.

Dans le cadre de 4MORE, le NoC ANoC de l'architecture FAUST a été utilisé pour réaliser une conception extrêmement statique, sans TDMA, qui est basée sur les simulations et ne supporte donc aucun aléa. Nous pensons que notre approche est beaucoup plus sûre et flexible et surtout notre outil nous a permis d'obtenir une solution très rapidement comparée à l'approche manuelle par simulation de FAUST.

7.5 Turbo décodeur

7.5.1 Présentation

Les contraintes de cette application nous ont été fourni par l'ENST Bretagne dans le cadre du groupe R-PUCE. C'est une application de turbo-décodage utilisant une architecture multi-ASIP originale décrite dans [92].

Le turbo décodage permet de réduire le taux d'erreurs dans les transmissions même avec un faible rapport signal sur bruit.

Le turbo code utilisé est le DVB RCS. Le turbo décodeur est constitué de deux décodeurs qui s'échangent des informations (appelée extrinsèques) pour faire converger leurs décisions. Ces communications dépendent du schéma d'entrelacement retenu.

Le turbo décodeur utilisé ici est mis en œuvre sur une plate-forme multiprocesseur pour atteindre de hauts débits.

Notre objectif est de concevoir un turbo décodeur dont il est possible de changer le schéma d'entrelacement. Pour transporter les communications entre les processeurs, nous avons choisi d'utiliser un NoC pour ces capacités de reconfiguration.

Dans notre cas d'étude, une information extrinsèque est constituée de 72 bits (9 octets), de plus, chacun des décodeurs est intégré sur quatre processeurs, et chaque processeur a deux ports d'entrée et deux ports de sortie. La figure 7.7 montre notre architecture. Ainsi, l'architecture se compose de 8 processeurs et 16 ports d'entrée et 16 ports de sortie.

Les données du problème :

- Une trame = 1000 symboles = 2000 bits ;
- Fréquence des décodeurs (processeurs)= 100MHz ;
- Une information extrinsèque est constituée de : 64bits + 1bit de contrôle d'inversion + 6 bits d'adresse = 71 bits = environ 9 octets (72 bits) ;
- Il y a 4 processeurs dans chaque décodeur, soit un total de 8 processeurs.
- Chaque Processeur a 2 voies en entrées ia et ib et deux voies en sorties oa et ob ;
- L'entrelassement est formé de 64 périodes d'émissions. Nous notons la durée d'une période d'émission :Temiss. Temiss=10 cycles.

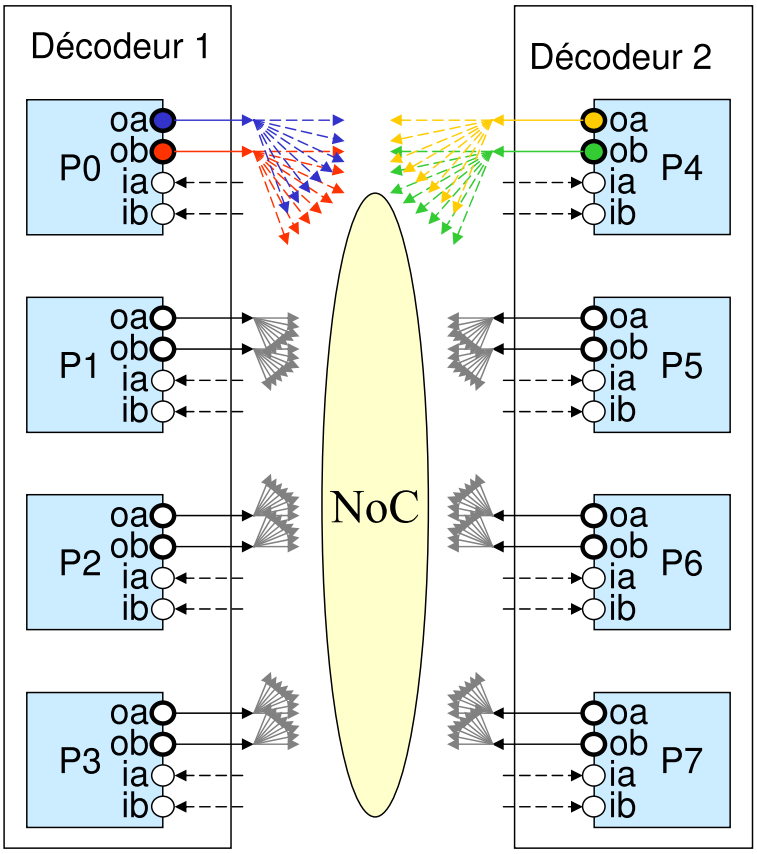


FIG. 7.7 – Architecture de la plate-forme pour le turbo décodage

– Durant chaque Temiss, les 16 voies d'émission (voies oa et ob des 8 processeurs) produisent une information extrinsèque et celle-ci doit être envoyée vers l'une des voies (ia ou ib) de l'un des 4 processeurs appartenant à l'autre décodeur. La destination de chaque information extrinsèque est définie pour chaque Temiss selon l'ordonnancement de l'entrelasseur ;
– La fréquence de production des informations extrinsèques par voie = $\frac{100.10^6}{10}$ = 10MHz ;
– La contrainte maximale de temps pour acheminer 1 information extrinsèque est de 3 Temiss, soit 30ns ;

Nos premières observations :

– Le débit par voie d'émission est de 90Mo/s ;
– 128 communications peuvent être identifiées. Alors que seulement 16 informations extrinsèques sont créées tous les Temiss ;
– Le débit total des échanges est de 16*90Mo/s=1,4Go/s. Une solution à base de bus est donc exclue ;
– Une entrée i peut recevoir jusqu'à 8 informations extrinsèques à la fois (ceci est très rare, la probabilité est de $1/8^8 = 1/16777216 = 59.10^{-9}$). Cela représente alors un débit de 8*90= 720Mo/s. En moyenne, une entrée reçoit une seule information extrinsèque par cycle d'émission, soit un débit de 90Mo/s. Une entrée peut aussi ne rien recevoir.

Nos premiers choix :
- les ports d'entrées ij et de sorties Oj sont regroupés en un seul port bidirectionnel ;
- Nous choisissons de ne pas faire de contrôle de flux de bout-en-bout. À l'émission, nous supposerons qu'il y a de la place à la réception pour recevoir les données.

Cette application présente des difficultés pour satisfaire sa communication. C'est probablement l'un des pires cas.
- La bande passante pire cas pouvant aboutir à une voie de réception est importante ;
- La latence maximale d'acheminement est faible ;
- La taille des messages échangés est faible, ce qui implique des petits paquets et donc un gaspillage important de la bande passante pour transmettre les nombreux entêtes de paquets ;
- 128 communications peuvent être identifiées, alors que seulement 16 communications nécessitent d'être transportées tous les Temiss. Il est donc difficile de pré-ordonnancer les communications.

7.5.2 Analyse

L'équipe du projet R-PUCE est composée de personnes de domaines différents. Cette pluridisciplinarité a été mise à profit durant ce projet. Les personnes travaillant dans le domaine des réseaux d'ordinateurs ont apporté des solutions pour qu'elles puissent être transposées au domaine des NoCs. Les compétences des personnes maîtrisant l'application ont permis d'imaginer des modes dégradés de l'application, ou des restrictions afin de réduire le coût de mise en œuvre de la solution. Enfin, nos compétences dans les NoCs ont permis de mettre en œuvre des solutions adaptées.

Nous avons ainsi proposé deux solutions faible coût, l'une utilisant le trafic BE avec un mode dégradé des performances de l'application et l'autre un trafic GT avec des restrictions sur l'entrelasseur utilisable.

7.5.3 Une première solution utilisant le *best effort*

Le trafic étant très varié et le nombre de communications ayant effectivement lieu étant faible par rapport à toutes les combinaisons possibles, il est difficile d'utiliser un pré-ordonnancement. Nous pouvons dans un premier temps voir quelle solution pourrait permettre l'utilisation d'une classe de trafic *best effort*. Les décodeurs s'échangent des informations leur permettant de converger vers le décodage des données. Nous notons que certaines informations apportent une correction importante et d'autres une correction insignifiante. L'absence de la transmission des communications apportant une faible correction peut, dans une certaine mesure, ne pas entraîner de dégradation significative sur les performances du turbo décodeur et seulement ralentir la convergence de ses décodeurs.

De part cette connaissance de l'application, nous pouvons faire le choix de transmettre en priorité les informations les plus pertinentes et en second lieu les informations qui peuvent être négligées. Nous pouvons alors utiliser un champ supplémentaire dans l'entête du paquet pour indiquer le niveau d'importance des données transportées et donc le niveau de priorité des paquets qui les transportent vis a vis des autres

paquets. Ainsi, dans les routeurs, nous intégrons des arbitres qui utilisent ce niveau de priorité pour allouer l'accès aux liens aux paquets les plus prioritaires.

La validation de cette solution nécessiterait des simulations qui n'ont pas été mises en œuvres. Nous notons que le niveau des priorités des paquets serait important au début, lorsque les deux décodeurs présente beaucoup de désaccord et se réduirait progressivement à mesure que les décodeurs convergent.

Ce travail continue, notamment pour la réalisation de la politique d'affectation des priorités.

7.5.4 Une solution utilisant le pré-ordonnancement par TDMA

La difficulté est de tirer parti de la connaissance des communications de l'application compte tenu de son schéma d'entrelacement pour pouvoir les pré-ordonnancer. Bien que le schéma d'entrelassement soit connu cela reste difficile car il contient 64 itérations (séquences).

De plus, nous devons dans une certaine mesure rester génériques afin de pouvoir intégrer différents schémas d'entralassement.

Ainsi, afin de ne pas surdimensionner l'architecture de communication du système, nous avons défini la restriction suivante :

Un port d'entrée ne pourra pas recevoir plus de cinq informations extrinsèques durant une période de 3 itérations consécutives. Ainsi, nous rendons les communications mutuellement exclusives sauf celles qui ont lieu en même temps ou durant des itérations voisines dans l'entrelasseur. La figure 7.8 image ces propos. Par exemple P5b ne pourra pas recevoir plus de 5 informations extrinsèques durant les périodes d'envoi I-1, I et I+1. Il en va de même pour tous les autres.

Cette restriction est compatible avec un grand nombre d'entrelasseur.

Ce n'est pas une solution ad hoc pour un entrelasseur considéré. Une solution conçue spécialement pour un entrelasseur serait évidement moins coûteuse, mais n'offrirait pas la flexibilité que nous recherchons.

Nous choisissons d'utiliser une topologie est une grille 2D de 4x4, avec des routeurs de 4 ports et une fréquence de 200MHz. La largeur d'un mot est de 24 bits.

7.5.5 Résultats

Notre outil a trouvé une solution avec une taille de la table de TDMA de 10 slots de temps. Il a déterminé un chemin pour chacune des communications. La latence la plus grande pour acheminer une information extrinsèque est de 0,20µs et est donc en accord avec la contrainte de temps spécifiée qui est de 0,30µs. Le coût en mots dans les FIFOs est d'un total de 416 mots.

2 slots de temps ont été réservés pour chaque communication. Comme un slot contient 2 mots et que la largeur de l'entête d'un paquet est de 1 mot, chaque communication bénéficie d'une portion de la bande passante du lien qui est de 1,5 slots sur les 10 slots de la table.

La bande passante est donc de 24*200 000 * 1,5/10= 720Mbits/s, soit 90Mo/s. Ce qui est exactement la bande passante nécessaire.

Un récepteur peut donc au plus recevoir 5 communications. La latence est de 0,20µs, ainsi une communication peut se retrouver en collision avec un communication

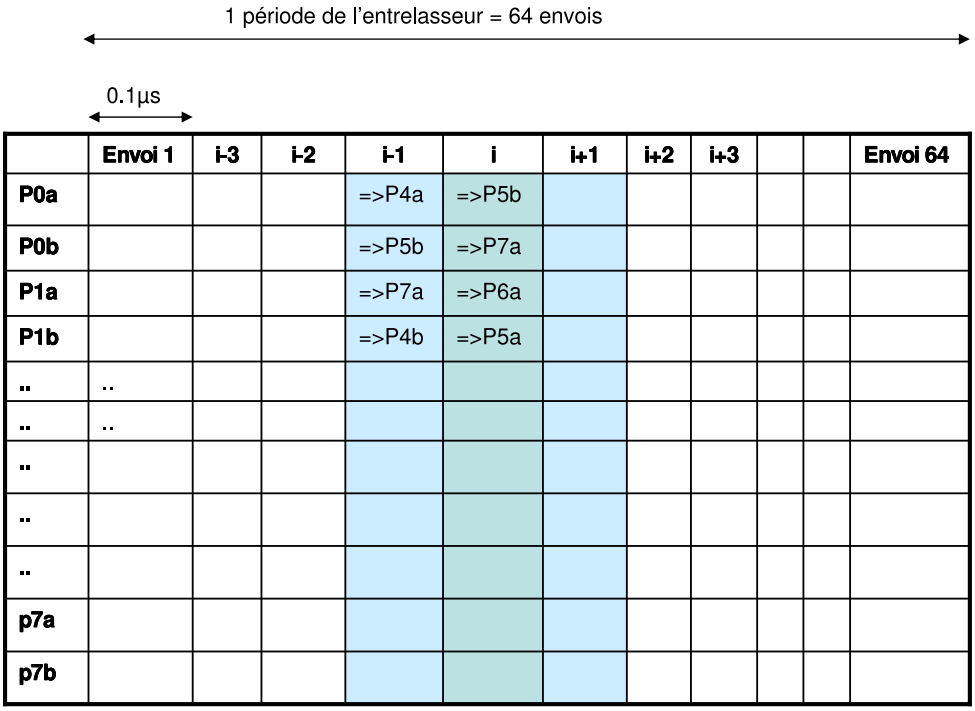


FIG. 7.8 – L'hypothèse permettant l'exclusions entre des communications

émise juste avant elle ou pendant ou juste après elle. C'est pourquoi nous limitons la possibilité d'envoi de 5 communications non mutuellement exclusive à 3 envois consécutif de l'entrelasseur.

Une largeur de 24 bits pour la largeur du mot transporté sur un lien est suffisante grâce à l'utilisation de notre hypothèse qui nous a permis d'utiliser les exclusions mutuelles, alors qu'une largeur de 72 bits aurait été nécessaire sans cette hypothèse.

7.5.6 Conclusion

La conception du NoC dépend des spécificités de l'application et de la nature même des données transportées. Un NoC totalement générique aurait conduit à un surdimensionnement injustifié. Par une bonne connaissance de l'application, nous avons déterminé deux types de compromis. Le premier est un compromis entre la complexité et la Qos de l'application. Le second est un compromis entre complexité et la généricité.

Nous avons ainsi proposé deux solutions, l'une utilisant le trafic BE et un mode dégradé des performances de l'application et l'autre un trafic GT avec des restrictions sur l'entrelasseur utilisable.

Nous avons donc su mener la conception du NoC en tirant partie de la connaissance des spécificités de l'application et en déterminant les compromis possible. Ceci est un exemple typique qui montre comment un outil de CAO peut avec des spécifications formelles des contraintes, aider le concepteur à trouver une solution à faible coût en lui permettant d'évaluer rapidement différents compromis.

Ceci ouvre de nouvelles perspectives pour généraliser cette approche.

7.6 Combinaisons des applications

7.6.1 Présentation

Nous cherchons à valider la technique de recherche de chemin spatio-temporel et l'intérêt de l'exclusion mutuelle des communications. Nous utilisons ici l'application traitement d'image et l'application de télécommunication avec ses deux parties, transmission (TX) et réception (RX). Toutes les communications utilisent la classe de trafic GT.

Quand les deux applications sont connectées simultanément, un total de 63 ports d'IPs, celles-ci sont regroupés en 25 *clusters*. Les IPs connectées à des NIports d'une même NI forme un cluster. Chaque cluster est connecté à une NI. Il y a donc 25 NIs. Le NoC spécifié à une topologie en grille 5x5. Les routeurs ont 5 ports avec des mots de 32 bits de large. la technique de routage est le *street-sign*. La fréquence du NoC est de 100MHz.

Nous étudions plusieurs cas, et pour chacun nous considérons l'impact de l'utilisation du contrôle de flux de bout-en-bout. Le contrôle de flux de bout-en-bout empêche que les données ne soient perdues à la réception par faute de place. Notons que le choix de ne pas utiliser ce contrôle de flux est envisageable dans les domaines des télécommunications et du traitement d'image en fonction du compromis qui est fait entre le coût et la QoS désirée.

7.6.2 Résultats

Le tableau 7.5 montre nos différents cas d'étude, qui correspondent à un NoC dédié à l'une ou les deux applications à la fois.

Dans chaque cas nous observons la taille de la table TDMA (en slots), la latence moyenne de transport sur chaque chemin (nombre de routeurs traversés) et la somme de la profondeur de toutes les FIFOs dans les NIs (en nombre de mots). Les résultats sont détaillés dans le tableau 7.6.

Le terme « ET » signifie que les applications fonctionnent en même temps, le « OU » indique qu'elles sont fortement exclusives et qu'elles ne fonctionnent donc pas en même temps, et le terme LME (pour *Light Mutual Exclusion*) signifies qu'elles sont faiblement exclusives.

Cas	spécifications applicatives
0	TX OU RX
1	TX ET RX
2	(TX ET RX) avec LME
3	traitement d'image
4	traitement d'image avec LME
5	traitement d'image ET TX ET RX
6	(traitement d'image ET TX ET RX)) avec LME
7	traitement d'image OU (TX ET RX)
8	(traitement d'image OU (TX AND RX)) avec LME
9	traitement d'image OU TX OU RX
10	(traitement d'image OU TX OU RX) avec LME

TAB. 7.5 – Les cas expérimentés

	Avec contrôle de flux de bout-en-bout			Sans contrôle de flux de bout-en-bout		
Cas	Taille de la table TDMA	Latence moyenne	Somme de la taille des FIFOs	Taille de la table TDMA	Latence moyenne	Somme de la taille des FIFOs
0	5	3,3	203	2	4,14	50
1	6	4,59	247	2	4	58
2	5	3,3	242	2	4,14	58
3	10	4,33	65	8	4,36	50
4	5	4,33	60	5	4,28	50
5	21	4,4	927	13	4,5	280
6	16	4,5	490	6	4,16	160
7	10	4,34	314	8	4,2	182
8	5	4,16	275	6	4,2	146
9	10	4,34	307	8	4,2	168
10	5	4,16	257	5	4,12	136

TAB. 7.6 – Les résultats

7.6.3 Analyse

Notre analyse est organisée en 5 points.

Premièrement, nous observons que notre algorithme de recherche des chemins parvient à trouver une solution, même avec des applications complexes.

Le second point concerne le coût du contrôle de flux de bout-en-bout. Le contrôle de flux de bout-en-bout nécessite le renvoi des crédits des consommateurs vers les producteurs. Dans le cas des transactions de lecture, le retour de crédit peut être intégré dans l'entête du paquet des requêtes suivantes. En revanche, dans le cas d'une transaction d'écriture, cela requière l'ajout d'une communication dans le sens opposé. De plus, ces communications occupent des slots de temps supplémentaires pour être transmis, ce qui provoque une augmentation de la taille de la table TDMA, et donc des FIFOs de découplage. De plus, des FIFOs sont nécessaires pour masquer le temps entre l'envoi des données depuis le producteur et le retour des crédits. Ceci cause une augmentation de la taille des FIFOs et donc du coût en surface. Nous observons que l'augmentation du coût en FIFO causé par l'utilisation du contrôle de flux de bout-en-bout est beaucoup plus forte pour l'application de télécommunication. Ceci s'explique par le fait que cette application n'utilise que des transactions d'écriture. Le retour de crédit utilisé dans le contrôle de flux de bout-en-bout ajoute une communication de retour qui double le nombre des communications. De plus, ces communications nécessitent des slots de temps pour être transmis, ce qui cause une augmentation de la taille de la table TDMA. Le résultat est donc une augmentation de la taille de la table TDMA et de la profondeur des FIFOs. L'application de traitement d'image quant à elle ne souffre pas trop de l'utilisation du contrôle de flux de bout-en-bout car une large part de ses transactions sont des lectures.

Le troisième point est le découpage de la table TDMA en slots. Dans le cas 8, qui permet de faire fonctionner soit l'une ou l'autre des applications (ME), nous pouvons voir que la taille de la table est supérieure aux tailles de celles des cas ou les applications fonctionnent seules (cas 2 et 4). La raison est que cette taille de la table doit offrir à toutes les communications un nombre entier de slots. La subdivision de la table qui accommode les deux applications peut donc être plus grande que chacune individuellement.

Le quatrième point est celui de l'amélioration apportée par l'exploitation des exclusions mutuelles. Les réservations des slots de temps peuvent être partagées par les communications mutuellement exclusives, ainsi la table de TDMA requise est plus petite et nous obtenons une réduction de la taille des FIFOs de découplage. Enfin, cette réservation multiple des slots permet d'augmenter le nombre de slots laissés disponibles ce qui favorise la découverte et l'utilisation de chemins plus courts, ce qui permet de plus la réduction des profondeurs des FIFOs qui masquent le temps de retour des crédits.

Enfin, nous pouvons voir que dans le cas 1 (avec contrôle de flux de bout-en-bout) qui met en œuvre l'application MC-CDMA, le coût en mots dans les FIFOs est supérieur au résultat obtenu dans la section 7.4. Ceci s'explique par le fait que la topologie est ici plus large (une grille 5x5 au lieu de 4x4). Les paquets parcourent donc un chemin plus long et les FIFOs qui masquent le temps de retour des crédits peuvent donc être plus profondes.

7.6.4 Conclusion

L'utilisation du contrôle de flux de bout-en-bout a un coût plus important dans le cas des transactions d'écriture que dans le cas des transaction de lecture. Son utilisation peut ne pas être justifiée dans le cas de certaines applications qui, soit acceptent des pertes de données, soit offrent la garantie que les données seront consommées avant l'arrivée des suivantes.

Notre technique d'allocation des chemins pour la classe de trafic garanti en bande passante trouve une solution, y compris dans le cas d'applications extrêmement complexes et hétérogènes. Elle tire avantageusement partie de l'exploitation de la connaissance des exclusions mutuelles entre les communications pour réduire la taille de la table TDMA et réduire la profondeur des FIFOs dans les interfaces réseau.

Cette expérience nous montre que la classe de trafic GT peut être utilisée avec un coût raisonnable en FIFOs. De plus, nous prenons en considération le fait que le GT ne requière que des FIFOs de taille minimale (2 mots) dans les routeurs puisqu'il ne peut pas y avoir de contention entre les paquets contrairement à la classe de trafic *best effort* et que les tailles des FIFOs dans les NIs peuvent être définies précisément. Nous pouvons conclure que le trafic GT par TDMA est une solution réaliste.

Cette capacité à satisfaire plusieurs applications sur un même NoC nous amène à la réflexion qu'un système d'exploitation pourrait surveiller et contrôler ces différentes applications afin d'adapter le NoC aux variations des besoins.

7.7 Conclusion

Ces expérimentations ont montré que l'architecture de NoC que nous proposons fonctionne, ce qui est bien sûr un préalable. D'autre part, nous avons vu que notre flot de conception permet de déterminer efficacement les paramètres du NoC.

La technique d'allocation des chemins que nous proposons parvient à trouver des solutions dans le cas d'applications complexes.

La définition des communications mutuellement exclusives permet la réutilisation des slots du TDMA et ainsi de trouver des solutions avec des tables de TDMA de taille minimale, ce qui peut permettre de réduire le coût en FIFO dans l'architecture.

Nous avons montré qu'il y a une grande complexité à déterminer les paramètres d'un NoC pour une application. Un outil de CAO est nécessaire pour aider le concepteur à concevoir un NoC. Il est ainsi possible d'explorer rapidement différentes solutions.

Nous avons résolu des cas très différents, issus des domaines du traitement du signal, de l'image et des télécommunications, ce qui prouve que nous disposons d'un outil généraliste.

Chapitre 8

Conclusion et Perspectives

8.1 Conclusion

Nous avons proposé dans ce mémoire une architecture de NoC et un flot de conception pour les réseaux sur puce dans le cadre d'applications possédant des contraintes de temps réel.

Nous avons proposé un modèle de NoC comportant de nombreux paramètres génériques pour nous offrir une grande liberté de mise en œuvre et ainsi disposer des aptitudes requises pour satisfaire le plus grand nombre d'exigences. Il est ainsi capable de manipuler différents types de trafic, dont un trafic garanti par TDMA.

L'espace de conception des NoCs est vaste et il est nécessaire de disposer de méthodes pour réaliser un NoC adapté aux besoins de l'application.

Le premier point consiste à prendre connaissance et décrire les besoins en communication d'une application. Or, les contraintes de l'application sont exprimées de façon globale, et les contraintes à satisfaire individuellement par chaque communication ne sont pas exprimées. Ainsi, nous avons proposé une méthode pour extraire les contraintes de communication depuis une description des tâches de l'application. Celle-ci repose sur le respect de règles qui assurent la bonne initialisation et le bon fonctionnement en cadence de l'application. Ces règles permettent d'extraire les contraintes de bande passante et de latence que les communications devront satisfaire.

Pour obtenir un ordonnancement TDMA correcte, il faut allouer des chemins à chacune des communications en respectant leur contrainte de bande passante et de latence, et en assurant qu'elles ne peuvent jamais se rencontrer. Ce problème est complexe et nécessite une approche heuristique. Nous avons proposé de représenter l'espace d'exploration par la topologie à laquelle nous ajoutons une dimension temporelle rebouclée que nous appelons topologie spatio-temporelle. Nous avons proposé une méthode heuristique afin de réaliser cette exploration. Celle-ci repose sur une technique d'allocation des chemins de façon concertée entre les communications qui doivent être placées afin d'offrir le plus de chance de succès pour permettre à chacune de trouver une solution qui lui convienne.

De plus, nous avons proposé d'identifier les communications qui ne peuvent pas se produire en même temps (mutuellement exclusives) de façon à permettre de mutualiser leurs réservations de slots de temps dans la topologie spatio-temporelle.

Cette méthodologie a été intégrée dans un outil afin de permettre d'automatiser et de valider son fonctionnement. Cet outil nous a permis de confronter notre méthodologie à des applications des domaines du traitement du signal, de l'image et des télécommunications.

Nous avons présenté un ensemble de résultats obtenus avec ces applications, qui montrent l'intérêt de tirer profit des communications mutuellement exclusives. Ainsi, la taille de la table TDMA nécessaire peut être réduite et nous pouvons limiter la profondeur de FIFO nécessaire. Ces expérimentations ont permis de démontrer l'aptitude de notre outil à trouver une solution.

Nous avons vu par les expérimentations qu'il peut être intéressant, voir nécessaire, de tenir compte des spécificités de l'application pour choisir la politique de transport des communications. Ainsi, nous nous sommes aperçus dans les cas de l'application turbo décodeur, que les messages n'avaient pas tous la même criticité en fonction de l'apport qu'ils réalisent et que certaines communications pouvaient être plus facilement négligées que d'autres en cas de conflit d'accès.

En marge de ce flot de conception, nous nous sommes intéressés à deux sujets qui sont l'aspect sécurisation du NoC et le problème des horloges dans les circuits de grande taille.

Nous avons présenté une solution pour vérifier à destination la provenance d'un paquet. Cette vérification ne se fait pas selon un identifiant ou une clef, mais selon les instructions de chemin que le paquet a utilisé pour atteindre cette destination. Cette clef est unique et sûre dans l'hypothèse que l'intégrité du NoC est garantie. L'intérêt de cette technique réside dans la simplicité de sa mise en œuvre. De plus, nous avons montré qu'il est également possible d'utiliser les instructions aller pour obtenir les instructions retour réorganisées dans le bon ordre (sans que ces instructions nécessitent d'avoir la même largeur en bits).

Pour pallier le problème de l'horloge qui ne peut plus être considérée comme synchrone sur l'ensemble d'un circuit de grande taille, nous avons proposé une division du NoC en sous parties (sub-NoC) fonctionnant chacune de façon synchrone mais ayant entre elles des horloges qui sont mésochrones (même horloges mais déphasées). Nous avons proposé une adaptation appelée routeur temporel pour permettre à un sous NoC qui utilise un TDMA (qui nécessite une notion commune d'horloge), d'intégrer un routeur se trouvant dans un autre domaine d'horloge. De plus, nous avons proposé des synchroniseurs qui permettent d'interconnecter des sub-NoCs possédant des TDMA distincts. Nous avons montré qu'il existait deux façons de mettre en œuvre ces derniers, soit en conservant le contrôle de flux au niveau du NoC global, soit en dépaquetant et rempaquetant les messages entre les sub-NoCs et en réalisant donc un contrôle de flux local à chaque sub-NoC. Nous avons montré les avantages de chacune de ces approches.

8.2 Perspectives

Bien que la méthodologie proposée conduise à des résultats de bonne qualité sur les exemples traités, plusieurs améliorations et extensions peuvent être apportées comme nous l'avons mentionné à plusieurs reprises dans ce document.

D'un point de vu développement et généricité, il serait intéressant de développer de nouveaux *wrappers* pour permettre de connecter au NoC des composants utilisant d'autres protocoles que celui du bus OPB.

Il serait intéressant d'élaborer un modèle pour permettre de donner au concepteur des indications sur le NoC spécifié (surface, consommation), sans qu'une synthèse soit nécessaire.

Si nous désirons intégrer notre algorithme de recherche des chemins afin de réaliser des reconfigurations en ligne, il est nécessaire d'accélérer son temps d'exécution en réalisant certaines approximations. Nous avons déjà spécifié les heuristiques possibles.

La recherche de chemin est réalisé uniquement dans un NoC ou sub-NoC synchrone. Des travaux pourraient être menés pour étendre la recherche des chemins dans le cadre d'un NoC constitué de plusieurs sub-NoCs fonctionnant chacun avec leur domaine TDMA. Il faudrait alors choisir par quels sub-NoCs intermédiaires il est le plus intéressant de traverser pour atteindre un autre sub-NoC.

L'aspect sécurité dans les NoCs n'est pas encore un sujet de recherche très étudié. La sécurité deviendra cependant un aspect important dans les systèmes multifonctions qui intégrerons un NoC. En plus de détecter des attaques ou des erreurs, il sera nécessaire d'offrir des contre-réactions au NoC.

Il est également nécessaire de permettre la convergence entre le système d'exploitation et la gestion du NoC afin de tirer parti de toutes les capacités de reconfiguration et d'adaptation du système.

Enfin, l'ajout de la synthèse de code SystemC en complément du code VHDL RTL actuellement généré est en cours de développement. Il permettra une simulation plus facile pour l'interaction entre le NoC et des programmes destinés à fonctionner sur des processeurs connectés à celui-ci. Cela permettra d'explorer plus facilement les aspects réaction de contre-attaque et convergence avec l'OS cités précédemment.

Annexe

Publications personnelles

Publication dans une revue internationale

– S. Evain, J-Ph. Diguet and D. Houzet, “NoC Design Flow for TDMA and QoS Management in a GALS Context”, EURASIP Journal on Embedded Systems, Volume 2006, Hindawi Publishing Corporation, 2006, accepté.

Publications dans des conférences internationales

– S. Evain, J-Ph. Diguet, Milad El Khodary and D. Houzet, “Automated derivation of NoC Communication Specifications from Application Constraints”, IEEE SIPS 2006, Workshop on Signal Processing Systems, Banff, AB, Canada, October 2-4, 2006.
– S. Evain, J-Ph. Diguet and D. Houzet, “µSpider NoC Road Map”, DATE 06 Workshops, Future Interconnects and Networks on Chip Workshops, March 10, 2006.
– S. Evain and J-Ph. Diguet, “From NoC Security Analysis To Design Solutions”, IEEE SIPS 2005, Workshop on Signal Processing Systems, Athens, Greece, November 2-4, 2005.
– S. Evain, J-Ph. Diguet and D. Houzet, “µSpider : a CAD Tool for efficient NoC design”, IEEE NORCHIP 2004, Oslo, NORWAY, November 8-9, 2004.
– S. Evain, J-Ph. Diguet and D. Houzet, “A CAD Tool for efficient NoC design”, IEEE ISPACS 2004, International Symposium on Intelligent Signal Processing and Communication Systems, Seoul, Korea, November 18-19, 2004.

Brevet

– Inventeurs : S. Evain et J-Ph. Diguet, “Routeur et réseau de routage”. Titulaire : Centre National de la Recherche Scientifique. Brevet FR 05/53280, déposé le 28 octobre 2005.

Table des figures

Liste des tableaux

Bibliographie

[1] « IST 4MORE project ». European project 2004-2007, http ://4more.av.it.pt/.

[2] S. Dutta, R. Jensen et A. Rieckmann, « Viper : A Multiprocessor SOC for Advanced Set-Top Box and Digital TV Systems ». *IEEE Design and Test of Computers*, vol. 18, pages 21–31, Sept-Oct 2001.

[3] « INTERCONNECT ». Rapport, *The International Technology Roadmap for Semiconductors :2005*, 2005.

[4] L. Benini et G. De Micheli, « Networks on Chip : A New Paradigm for Systems on Chip Design ». In *Proceedings of the conference on Design, automation and test in Europe*, page 418, IEEE Computer Society, 2002.

[5] W. J. Dally, « Interconnect limited VLSI architecture ». In *International Interconnect Technology Conference Proceedings*, pages 15–17, 1999.

[6] « SIA. 1997. National technology roadmap for semiconductors ». Rapport, *Semiconductor Industry Association*, 1997.

[7] L. Benini et G. De Micheli, « Networks on Chips : A New SoC Paradigm ». *Computer*, vol. 35, n°1, pages 70–78, 2002.

[8] W. J. Dally et B. Towles, « Route Packets, Not Wires : On-Chip Interconnection Networks ». In *Design Automation Conference*, pages 684–689, Jun 2001.

[9] « ARM - AMBA ». http ://www.arm.com/armtech/AMBA/.

[10] « IBM - CoreConnect ». http ://www.chips.ibm.com/products/coreconnect/.

[11] H. Dubois, *Analyse des systèmes multiprocesseurs : Application à la mise en oeuvre sous contraintes d'algorithmes de traitement d'images*. Thèse de Doctorat, Université de Rennes I, ENSSAT LASTI, Lannion, 1991.

[12] P. Guerrier et A. Greiner, « A generic architecture for on-chip packet-switched interconnections ». In *Proceedings of the conference on Design, automation and test in Europe*, pages 250–256, ACM Press, 2000.

[13] Arteris, « A comparison of Network-on-Chip and Busses ». http ://www.arteris.com, 2005.

[14] V. Bangalore et N. Nayak Sujir, « Performance Enhancement of On-Chip communication Architecture using Multicast ». *On-chip communication Networks Research paper ,Spring 2002*, 2002.

[15] E. Bolotin, A. Morgenshtein, I. Cidon, R. Ginosar et A. Kolodny, « Automatic Hardware-Efficient SoC Integration by QoS Network on Chip ». In *ICECS-2004.*, 2004.

[16] T. Bjerregaard et S. Mahadevan, « A Survey of Research and Practices of Network-on-Chip ». *ACM Computing Surveys (CSUR)*, vol. 38, page 1, 2006.

[17] E. Rijpkema, K.G.W. Goossens et A. Radulescu, « Trade Offs in the Design of a Router with Both Guaranteed and Best-Effort Services for Networks on Chip ». In *Design, Automation and Test in Europe Conference and Exhibition (DATE'03)*, (Munich, Germany), March 03 - 07 2003.

[18] L. M. Ni et P. K. McKinley, « A Survey of Wormhole Routing Techniques in Direct Networks ». *Computer*, vol. 26, n°2, pages 62–76, 1993.

[19] W. J. Dally et H. Aoki, « Deadlock-Free Adaptive Routing in Multicomputer Networks Using Virtual Channels ». *IEEE Trans. Parallel Distrib. Syst.*, vol. 4, n°4, pages 466–475, 1993.

[20] W. J. Dally, « Virtual-channel flow control ». In *Proceedings of the 17th annual international symposium on Computer Architecture*, pages 60–68, ACM Press, 1990.

[21] M. Millberg, E. Nilsson, R. Thid et A. Jantsch, « Guaranteed Bandwidth Using Looped Containers in Temporally Disjoint Networks within the Nostrum Network on Chip ». In *Design, Automation and Test in Europe Conference and Exhibition Volume II (DATE'04)*, vol. 2, (Paris, France), page 20890, February 16 - 20 2004.

[22] K. Goossens, J. Dielissen, J. van Meerbergen, P. Poplavkoy, A. Radulescu, E. Rijpkema, E. Waterlander et P. Wielage, « Guaranteeing the quality of services in networks on chip ». *Networks on chip*, vol. Chapter 4, pages 61–82, 2003.

[23] A. Leroy, P. Marchal, A. Shickova, F. Catthoor, F. Robert et D. Verkest, « Spatial division multiplexing : a novel approach for guaranteed throughput on NoCs ». 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 81–86, ACM Press, 2005.

[24] J. Bainbridge et S. Furber, « Chain : A Delay-Insensitive Chip Area Interconnect ». *IEEE Micro*, vol. 22, n°5, pages 16–23, 2002.

[25] F. Steenhof, H. Duque, B. Nilsson, K. Goossens et R. P. Llopis Peset Llopis, « Networks on Chips for High-End Consumer-Electronics TV System Architectures ». In *Proc. Design, Automation and Test in Europe Conference and Exhibition (DATE)*, mar 2006.

[26] S. Evain, J-Ph. Diguet et D .Houzet, « μSpider NoC Road Map ». In *Work. Future Interconnects and Networks on Chip Workshops (DATE06)*, (Munich, Germany), mar 2006.

[27] J-P. Sansonnet, « Le Transputer d'Inmos. Architecture des Machines Parallèles ».

[28] L. Benini et G. De Micheli, « Powering networks on chips : energy-efficient and reliable interconnect design for SoCs ». In *Proceedings of the 14th international symposium on Systems synthesis*, pages 33–38, ACM Press, 2001.

[29] « The network simulator - ns-2 ». http ://nsnam.isi.edu/nsnam/index.php.

[30] Y.R. Sun, S. Kumar et A. Jantsch, « Simulation and Evaluation for a Network on Chip Architecture using NS-2 ». In *Proc. NORCHIP 2002, Copenhagen.*, 2002.

[31] F. Karim, A. Nguyen et S. Dey, « An Interconnect Architecture for Networking Systems on Chips ». *Computer*, vol. 22, n°5, pages 36–45, 2002.

[32] E. Bolotin, I. Cidon, R. Ginosar et A. Kolodny, « QNoC : QoS architecture and design process for network on chip ». *Journal of Systems Architecture*, 2003.

[33] J. Dielissen, A. Radulescu, K. Goossens et E. Rijpkema, « Concepts and Implementation of the Philips Network-on-Chip ». In *IP-Based SOC Design*, (Grenoble France), nov 2003.

[34] H. Charlery et A. Greiner, « Systèmes intégrés : Un micro-réseau d'interconnexion à commutation de paquets respectant la norme VCI ». *Troisième Colloque CAO de circuits et systèmes intégrés*, mai 2002.

[35] M. Dall'Osso, G. Biccari, L. Giovannini, D. Bertozzi et L. Benini, « Xpipes : a Latency Insensitive Parameterized Network-on-Chip Architecture for Multi-Processor SoCs ». In *IEEE International Conference on Computer Design (ICCD'03)*, pages 536–539, 2003.

[36] K. Goossens, J. Dielissen, O. P. Gangwal, S. Gonzalez Pestana, A. Radulescu et E. Rijpkema, « A Design Flow for Application-Specific Networks on Chip with Guaranteed Performance to Accelerate SOC Design and Verification ». In *Proc. Design, Automation and Test in Europe Conference and Exhibition (DATE)*, march 2005.

[37] D. Bertozzi, A. Jalabert, S. Murali, R. Tamhankar, S. Stergiou, L. Benini et G. De Micheli, « NoC Synthesis Flow for Customized Domain Specific Multiprocessor Systems-on-Chip ». *IEEE Trans. Parallel Distrib. Syst.*, vol. 16, n°2, pages 113–129, 2005.

[38] S. Murali et G. De Micheli, « SUNMAP : a tool for automatic topology selection and generation for NoCs ». In *DAC '04 : Proceedings of the 41st annual conference on Design automation*, pages 914–919, ACM Press, 2004.

[39] S. Murali, L. Benini et G. De Micheli, « Mapping and Physical Planning of Networks-on-Chip Architectures with Quality-of-Service Guarantees ». In *Design Automation Conference, 2005*, vol. 1, (Shangai), pages 27–32, Januray 2005.

[40] A. Jalabert, S. Murali, L. Benini et G. De Micheli, « xpipesCompiler : A Tool for Instantiating Application Specific Networks on Chip ». In *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition Volume II (DATE'04)*, page 20884, IEEE Computer Society, 2004.

[41] P. Guerrier, *Un réseau d'interconnexion pour systèmes intégrés.* Thèse de Doctorat, Université Pierre et Marie Curie LIP6, 2000.

[42] A. Andriahantenaina, H. Charlery, A. Greiner, L. Mortiez et C. Albenes Zeferino, « SPIN : A Scalable, Packet Switched, On-Chip Micro-Network ». In *Design, Automation and Test in Europe Conference and Exhibition (DATE'03 Designers' Forum)*, page 20070, 2003.

[43] A. Greiner, « The SPIN & DSPIN networks on chip ». Journée Informatique Embarquée : Du matériel au Logiciel, 2005.

[44] A. Radulescu, J. Dielissen, K. Goossens, E. Rijpkema et Paul Wielage, « An Efficient On-Chip NI Offering Guaranteed Services, Shared-Memory Abstraction, and Flexible Network Configuration ». In *TCAD*, 2004.

[45] A. Hansson, K. Goossens et A. Rădulescu, « 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, september 2005.

[46] S. Murali, M. Coenen, A. Radulescu, K. Goossens et G. De Micheli, « A Methodology for Mapping Multiple Use-Cases on to Networks on Chip ». In *Design, Automation and Test in Europe Conference and Exhibition (DATE)*, page 118, mar 2006.

[47] E. Bolotin, I. Cidon, R. Ginosar et A. Kolodny, « Cost considerations in Network on Chip ». In *Special issue on Networks on Chip, Integration - The VLSI Journal, 2003*, 2003.

[48] D. Rostislav, V. Vishnyakov, E. Friedman et R.Ginosar, « An Asynchronous Router for Multiple Service Levels Networks on Chip ». In *ASYNC 05 11th IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC'05)*, vol. 0, pages 44–53, 2005.

[49] Arteris, « Networks on Chip for Managing On-Chip Communications ». SOC-central, May 8 2006.

[50] P. Martin, « Design of a Virtual Component Neutral Network-on-Chip Transaction Layer ». In *DATE '05 : Proceedings of the conference on Design, Automation and Test in Europe*, (Washington, DC, USA), pages 336–337, IEEE Computer Society, 2005.

[51] Arteris, « Arteris and CoWare Announce Integration of Arteris Network on Chip (NoC) Technology into CoWare Platform Architect ESL Design Environment ». http ://www.arteris.com, July 17 2006.

[52] « STNoC : Building a New System-on-Chip Paradigm ». http ://www.st.com/stonline/press/news/back2005/b9014t.htm, December 2005.

[53] L. Bononi et N. 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.

[54] « Arteris Announces STMicroelectronics Use of NoC for Next Generation Wireless Infrastructure Platform ». http ://www.arteris.com, March 15 2006.

[55] T. Bjerregaard, *The MANGO Clockless Network-on-Chip : Concepts and Implementation*. Thèse de Doctorat, Technical University of Denmark, 2005.

[56] E. Beigné, 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 11th IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC'05)*, pages 54–63, 2005.

[57] Y. Durand, C. Bernard et D. Lattard, « FAUST : On-chip distributed architecture for a 4G baseband modem SoC ». In *Design & Reuse IP-SoC*, (Grenoble, France), Dec. 2005.

[58] F. Clermidy, D. Varreau et D. Lattard, « A NoC-based communication framework for seamless IP integration in complex systems ». In *Design & Reuse IP-SoC, Grenoble, France*, décembre 2005.

[59] S. Fujita, K. Abe1, K. Nomura1 et T. Lee, « Novel Performance of Three-dimensional (3D) On-chip Crossbar Bus using non- Silicon Transistors ». DATE 06 Workshop, Future Interconnects and Networks on Chip, Munich, Germany, 2006.

[60] O. Connor et F. Gaffiot, « Advanced Research in on-chip optical interconnects ». *Lower Power Electronics and Design*, 2004.

[61] D. Zhao, S. Upadhyaya et M. Margala, « Design of a wireless test control network with radio-on-chip technology for nanometer system-on-a-chip ». *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 25, pages 1411– 1418, July 2006.

[62] S. Evain, J-Ph. Diguet et D. Houzet, « A Generic CAD Tool for Efficient NoC Design ». In *IEEE Int. Symp. on Intelligent Signal Processing and Communication Systems (ISPACS)*, (Seoul, Korea), pages 728–733, nov 2004.

[63] T. Marescaux, J-Y. Mignolet, A. Bartic, W. Moffat, D. Verkest, S. Vernalde et R. Lauwereins, « Networks on Chip as Hardware Components of an OS for Reconfigurable Systems ». In *Intl. Conf. on Field Programmable Logic and Applications (FPL)*, (Lisbon, Portugal), pages 595–605, September 2003.

[64] Y. W. Lu, G. K. Yeh et J. B. Burr Spa, « A 15mW 1.6Gb/s Wormhole Data Router for 2-D Meshes ». Symp. on VLSI Circuits, 1996., 1996.

[65] J. A. Williams, N. W. Bergmann et X. Xie, « FIFO Communication Models in Operating Systems for Reconfigurable Computing ». In *FCCM '05 : Proceedings of the 13th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM'05)*, (Washington, DC, USA), pages 277–278, IEEE Computer Society, 2005.

[66] J. A. Williams et N. W. Bergmann, « Programmable parallel coprocessor architectures for reconfigurable system-on-chip ». In *Field-Programmable Technology*, pages 193– 200, 6-8 Dec 2004.

[67] M. Coenen, S. Murali, A. Rădulescu, K. Goossens et G. De Micheli, « A buffer-sizing Algorithm for Networks on Chip using TDMA and credit-based end-to-end Flow Control ». In *Int'l Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS)*, (Seoul, Korea), octobre 2006.

[68] O. P. Gangwal, A. Radulescu, K. Goossens, S. Gonzalez Pestana et E. Rijpkema, « Building Predictable Systems on Chip : An Analysis of Guaranteed Communication in the Æthereal Network on Chip ». In *Dynamic and Robust Streaming In And Between Connected Consumer-Electronics Devices* (Peter van der Stok (ed.)), « 1 »., Kluwer, 2005.

[69] S. Murali, M. Coenen, A. Rădulescu, K. Goossens et G De Micheli, « Mapping and Configuration Methods for Multi-Use-Case Networks on Chips ». In

Proc. Design Automation Conference. Asia and South Pacific (ASP-DAC), jan 2006.

[70] C. Marcon, M. Kreutz, N. Calazans et A. Susin, « Models for Embedded Application Mapping onto NoCs : Timing Analysis ». In *RSP'2005 International Workshop on Rapid System Prototyping*, vol. 00, (Montreal, Canada), pages 17–23, June 8-10, 2005 2005.

[71] S. Evain, J-Ph. Diguet et D .Houzet, « μSpider : a CAD Tool for efficient NoC design ». In *IEEE NORCHIP*, (Oslo, NORWAY), pages 218–221, nov 2004.

[72] D. Andreasson et S. Kumar, « Slack-time aware routing in NoC systems ». In *International Symposium Circuits and Systems (ISCAS)*, pages 2353–2356, 2005.

[73] S. Evain et J-Ph. Diguet, « From NoC Security Analysis To Design Solutions ». In *IEEE Work. on Signal Processing Systems (SIPS)*, (Athens, Greece), pages 166–171, oct 2005.

[74] D. Park, C. Nicopoulos, J. Kim, N. Vijaykrishnan et C. R. Das, « Exploring Fault-Tolerant Network-on-Chip Architectures ». In *Proceedings of the 2006 International Conference on Dependable Systems and Networks (DSN'06)*, pages 93–104, June 2006.

[75] C. H. Gebotys et R. J. Gebotys, « A framework for security on NoC technologies ». In *VLSI, 2003. Proceedings. IEEE Computer Society Annual Symposium*, pages 113 – 117, 20-21 Feb. 2003.

[76] K. Goossens, P. Wielage, A. Peeters et J. van Meerbergen, « Networks on Silicon : Combining Best-effort and Guaranteed Services ». In *Proceedings of Design Automation and Test Conference in Europe*, pages 423–425, March 2002.

[77] J. Mitola., « The software radio architecture ». *IEEE Communications Magazine*, vol. 33, pages 26–38, May 1995.

[78] F-X. Standaert, L van Oldeneel tot Oldenzeel, D. Samyde et J-J. Quisquater, « Power Analysis of FPGAs : How Practical is the Attack ? ». In *FPL*, pages 701–711, 2003.

[79] T. Marescaux, A. Bartic, D. Verkest, S. Vernalde et R. Lauwereins, « Interconnection Networks Enable Fine-Grain Dynamic Multi-Tasking on FPGAs ». In *FPL'02*, pages 795–805, Sep 2002.

[80] L. Bossuet, W. Burleson et G. Gogniat, « Dynamically Configurable Security for SRAM FPGA Bitstreams ». In *IEEE Reconfigurable Architectures Workshop, RAW, Workshop of IEEE IPDPS,*, (Santa Fé, New Mexico, USA), april 2004.

[81] T. Martin, M. Hsiao, D. Ha et J. Krishnaswami, « Denial-of-Service Attacks on Battery-powered Mobile Computers ». In *IEEE Pervasive Computing Conference*, (Orlando, Florida), pages 309–318, March 2004.

[82] S. Evain, J-Ph. Diguet et D .Houzet, « NoC design flow for TDMA and QoS Management in a GALS context ». *EURASIP Journal on Embedded Systems*, vol. 2006, 2006.

[83] D. Wiklund, « Mesochronous clocking and communication in on-chip networks ». In *Proc. Swedish system-on-chip conference (SSoCC´03)*, (Eskilstuna), April 8-9 2003.

[84] A. Lines, « Nexus : an asynchronous crossbar interconnect for synchronous system-on-chip designs ». In *Proc. 11th Symposium on High Performance Interconnects*, pages 2–9, 20-22 Aug 2003.

[85] T. Bjerregaard et J. Sparsø, « A Scheduling Discipline for Latency and Bandwidth Guarantees in Asynchronous Network-on-Chip ». In *ASYNC 05 11th IEEE International Symposium on Asynchronous Circuits and Systems*, pages 34–43, 2005.

[86] D. Hillman, « Using Mobilize Power Management IP for Dynamic and Static Power Reduction in SoC at 130 nm ». In *Design, Automation and Test in Europe, 2005. Proceedings*, September 30 2005.

[87] L. P. Carloni et A. L. Sangiovanni-Vincentelli, « Coping with Latency in SOC Design ». *computer IEEE Micro*, vol. 22, n°5, pages 24–35, 2002.

[88] J-Ph. Diguet, G. Gogniat, J. L. Philippe, Y. Le Moullec, S. Bilavarn, C. Gamrat, K. Ben Chehida, M. Auguin, X. Fornari et P. Kajfasz, « EPICURE : A partitioning and co-design framework for reconfigurable computing ». *Journal of Microprocessors and Microsystems*, vol. 30, pages 367–387, 2006.

[89] Y. Eustache, J-Ph. Diguet et M. Elkhodary, « RTOS-based Hardware Software Communications and Configuration Management in the context of a Smart Camera ». In *Engineering of Reconfigurable Systems and Algorithms (ERSA'06)*, 2006.

[90] A. Chouly, A. Brajal et S. Jourdan, « Orthogonal multicarrier techniques applied to direct sequence spread spectrum CDMA systems ». In *GLOBECOM'93*, pages 1723–1728, 1993.

[91] J. Delorme, D. Houzet, R. Lemaire, et D. Lattard, « Proposition of a benchmark for evaluation of cores mapping onto noc architectures ». In *ReCoSoC'05 conference*, (Montpellier, France), June 2005.

[92] O. Muller, A. Baghdadi et M. Jezequel, « ASIP-Based Multiprocessor SoC Design for Simple and Double Binary Turbo Decoding ». In *IEEE/ACM Design, Automation and Test in europe*, (Munich, Germany), March 6-10 2006.

Acronymes et Abréviations

AHB Advanced High-speed Bus

ALG Asynchronous Latency Guarantee

ANoC Asynchronous Network-on-Chip

ASIC Application Specific Integrated Circuit

AXI Advanced eXtensible Interface

BE Best-Effort

CAO Conception assistée par ordinateur

CCM Central Configuration Module

DMA Direct Memory Access

DPA Differential power analysis

DTD Document Type Definition

FAUST Flexible Architecture of Unified System for Telecom

FIFO First In First Out

FLIT FLow control unIT

FPGA Field Programmable Gate Array

FSL Fast Simplex Link

GALS Globally asynchronous, locally synchronous

GS Guaranteed Services

GT Guaranteed Throughput

IPs Intellectual Property

LME Light Mutual Exclusion

MC-CDMA Multi-Carrier Code Division Multiple Access

MC-SS-MA Multi-Carrier Spread Spectrum Multiple Access

ME Mutual Exclusion

MIMO Multiple-Input Multiple-Output

NA Network Adapter

NI Network Interface

NoC Network on Chip

NS2 Network Simulator

NTTP NoC Transaction and Transport Protocol

OCP Open Core Protocol

OPB On-chip Peripheral Bus

OS Operating System

OSI Open Systems Interconnection

PHIT PHysical unIT

QDI Quasi-Delay-Insensitive

QoS Quality of Service

R-PUCE Réseau embarqué sur PUCE

RoC Radio-on-Chip

RTL Register Transfer Level

RTOS Real-time operating system

SCP Self Complemented Path

SDM Spatial Division Multiplexing

SoC System on Chip

SPA Source Path Authentication

TBP Trusted Boomerang Path

TC Time Coder

TDMA Time Division Multiple Access

TLM Transaction level modeling

TR Time-routing

UMARS Unified MApping, Routing and Slot allocation

VC Virtual Channel

VCI Virtual Component Interface

VHDL Very High Speed Integrated Circuit Hardware Description Language

XML eXtensible Markup Language